Philipps Universität Marburg

Grundlagen der Wirtschaftsinformatik

Teil 4

Prof. Dr. Alpar

Sommersemester 2019

13. Phasenmodelle in der Softwareentwicklung

Definition: Softwareentwicklung

Die Softwareentwicklung umfasst alle Aktivitäten der Softwareplanung, der Softwaredefinition, des Softwareentwurfs und der Softwareumsetzung unter Berücksichtigung von Kundenwünschen und geforderten Qualitätseigenschaften (in Anlehnung an (Balzert 2009)).

Qualitätsanforderungen an IS

- Änderbarkeit
- Allgemeingültigkeit
- Benutzerfreundlichkeit
- Effizienz
- Funktionsabdeckung
- Kompatibilität
- Portabilität
- Produktivität
- Richtigkeit
- Robustheit
- Sicherheit
- Testbarkeit

Stakeholdergruppen - Netzdiagramm

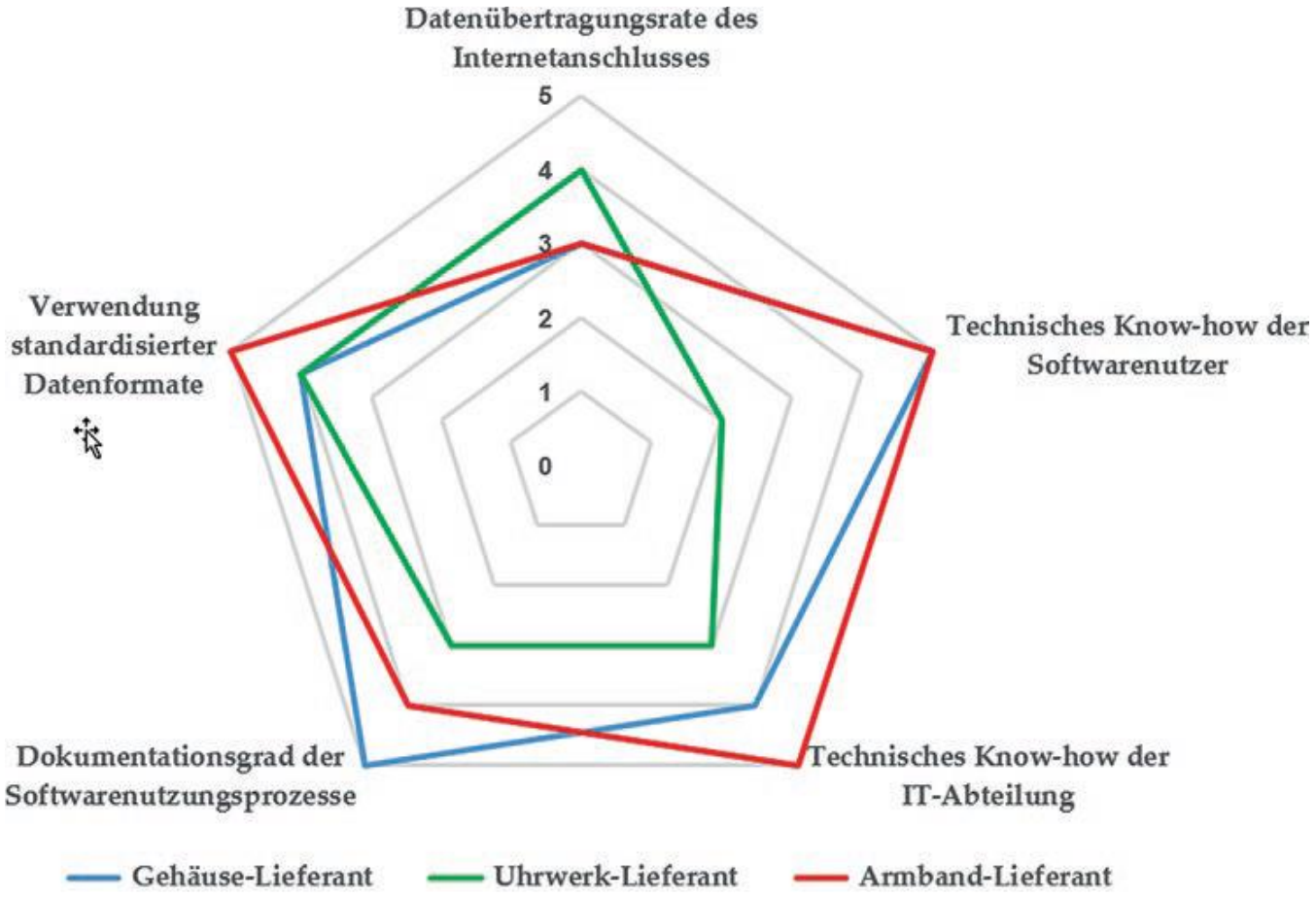

Abb. 13-7: Beschreibung von Stakeholdergruppen mit Hilfe eines Netzdiagramms im Rahmen des Uhren-Beispiels

Grundlegende Entwicklungsstrategien

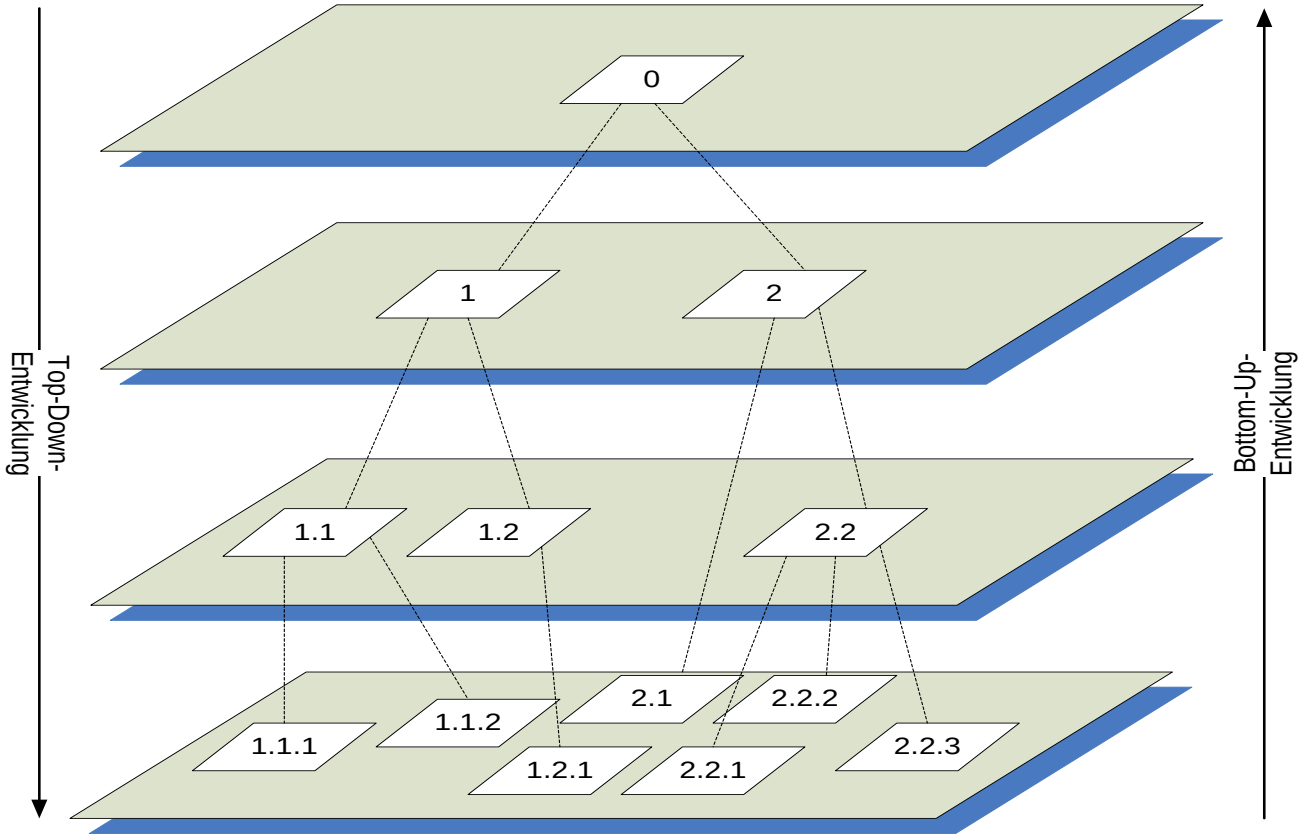

Abb. 13-1: Grundlegende Entwicklungsstrategien
[Schönthaler/Neméth 1990, S. 17]

Definitionen

Methode

Eine Methode enthält Aussagen über die Aktivitäten des Vorhabens und deren Reihenfolge, über die hierbei verwendeten Techniken, über die beteiligten Akteure bzw. Rollen sowie über die Repräsentation der Ergebnisse.

Vorgehensmodell

Ein Vorgehensmodell enthält die zur Erreichung eines bestimmten Ergebnisses notwendigen Aktivitäten, und stellt diese in einer sachlogischen Reihenfolge dar.

Lebenszyklus einer Software

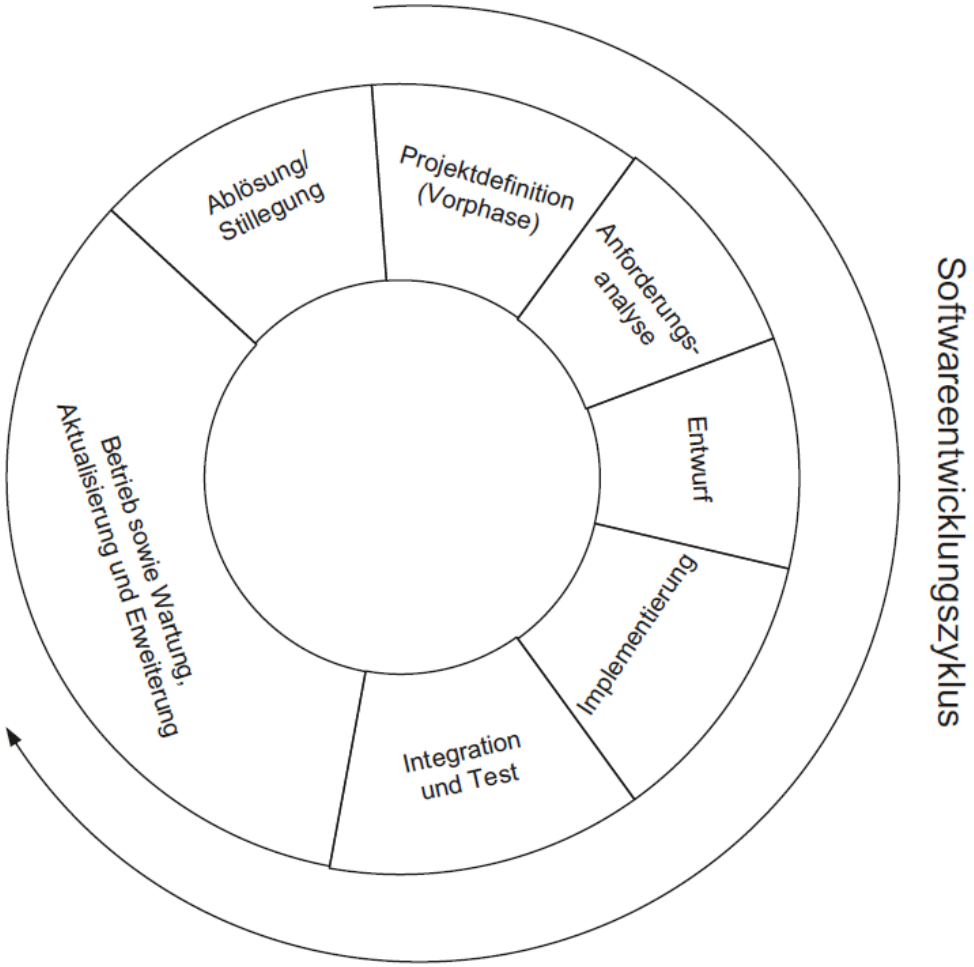

Abb. 13-3: Lebenszyklus einer Software

Wasserfallmodell

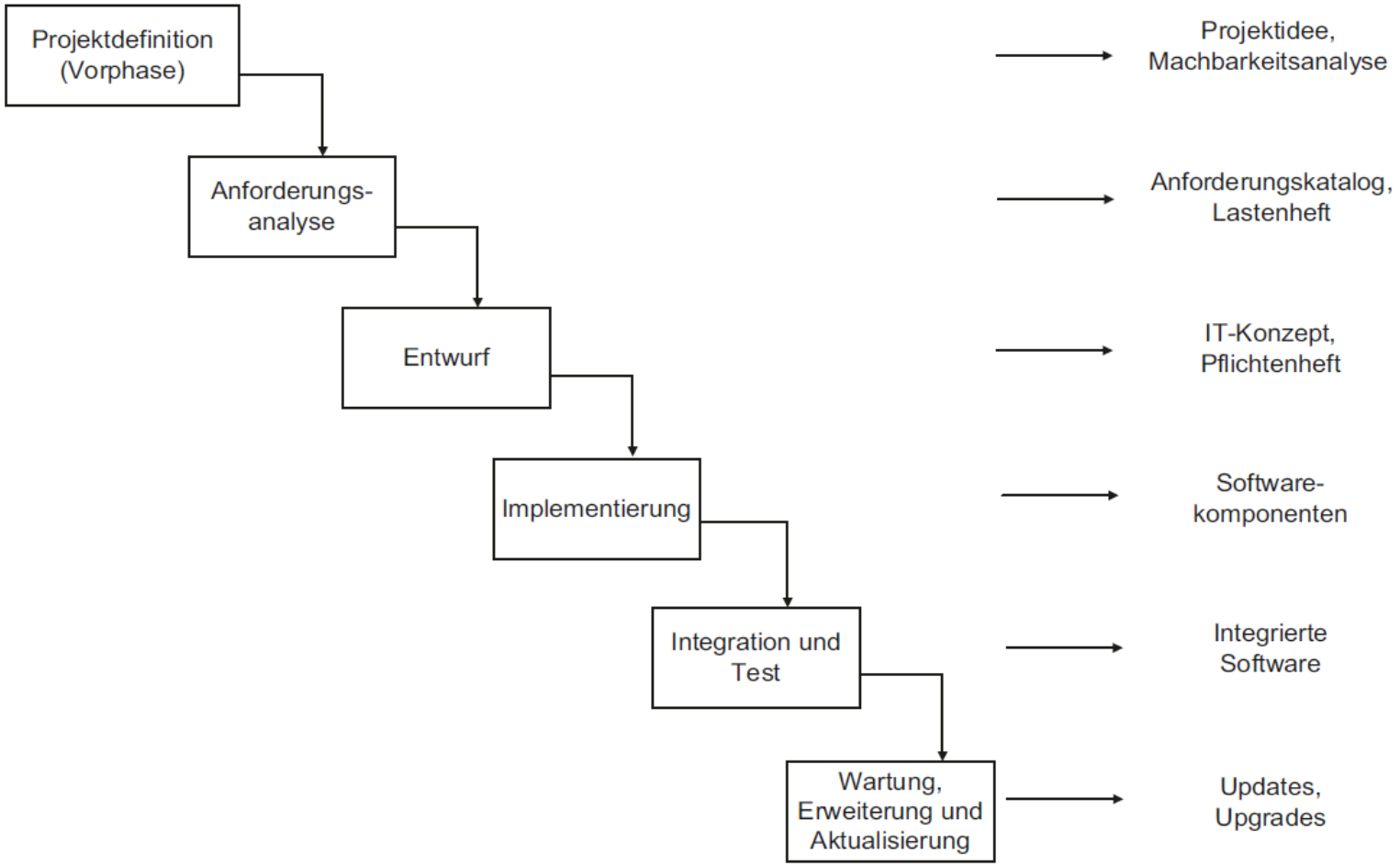

Abb. 13-4: Softwareentwicklungsphasen und ihre Ergebnisse am Beispiel des klassischen Wasserfallmodells

Inhalte eines Lastenhefts

- Ausgangssituation und Zielsetzung,
- Überblick über das zu gestaltende IS-System sowie über die Teilsysteme (IS-Architektur),
- funktionale und nichtfunktionale Anforderungen,
- Anforderungen an den Auftragnehmer (z. B. Zertifizierungen),
- Anforderungen an das Projekt- und Qualitätsmanagement des Auftragnehmers,
- Rahmenbedingungen für die Softwaregestaltung, und
- Abnahmekriterien.

Anforderungen

Eine Anforderung ist eine Aussage über eine Eigenschaft oder Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen.

• *Funktionale Anforderungen* adressieren die funktionalen Verhaltensweisen und Strukturen einer Software inklusive der relevanten Systemzustände. Zusätzlich umfassen sie auch die Inputs (z. B. Daten, manuelle Eingaben, Ereignisse) sowie die sich aus dem funktionalen Verhalten der Software ergebenden Outputs (z. B. Daten, Reaktionen des Systems).
• *Nichtfunktionale Anforderungen* beziehen sich auf die Qualitätsattribute der gewünschten Funktionen (z. B. Zuverlässigkeit, Ressourcenauslastung Nutzerfreundlichkeit), die Anforderungen an die Gesamtanwendung (z. B. Notfallmaßnahmen, Mitarbeiterqualifikationen), die Vorgaben für die Softwareerstellung (z. B. Projektorganisation, verwendete Hilfsmittel) sowie Anforderungen an die Prüfung, Einführung und den Betrieb der Software.

Abwicklung des Projekts

Typ	Funktional	Nichtfunktional	Einschränkung (der Lösung)	Domain Requirements
Inhalt	Nutzungsanforderung	Leistungsanforderung	Funktionalität	
Scope	Geschäftsanforderung	Benutzeranforderung	Technische Anforderung	
Detaillierungsgrad	Systemanforderung	Subsystemanforderung	Softwareanforderung	
Bezugs-Objekttyp	Produktanforderungen	Prozessanforderungen		
Form	Text	Schablone	Tabelle	Grafik	Modell
Formalität	Informell	Semiformal	Formal	
Quelle	Nutzer	Kunde	Vertrieb	Lieferant	
Reife	Initial	Abgestimmt	Verabschiedet	Zurückgewiesen
Verbindlichkeit	Wunsch	Feste Absicht	Vorschlag	Kommentar	Pflicht
Priorisierung	Wichtig, dringend	Wichtig, nicht dringend	Nicht wichtig, nicht dringend	

Abb. 13-8: Klassifikationskriterien für Anforderungen (Partsch 2010)

Vorgehen im Requirements Engineering

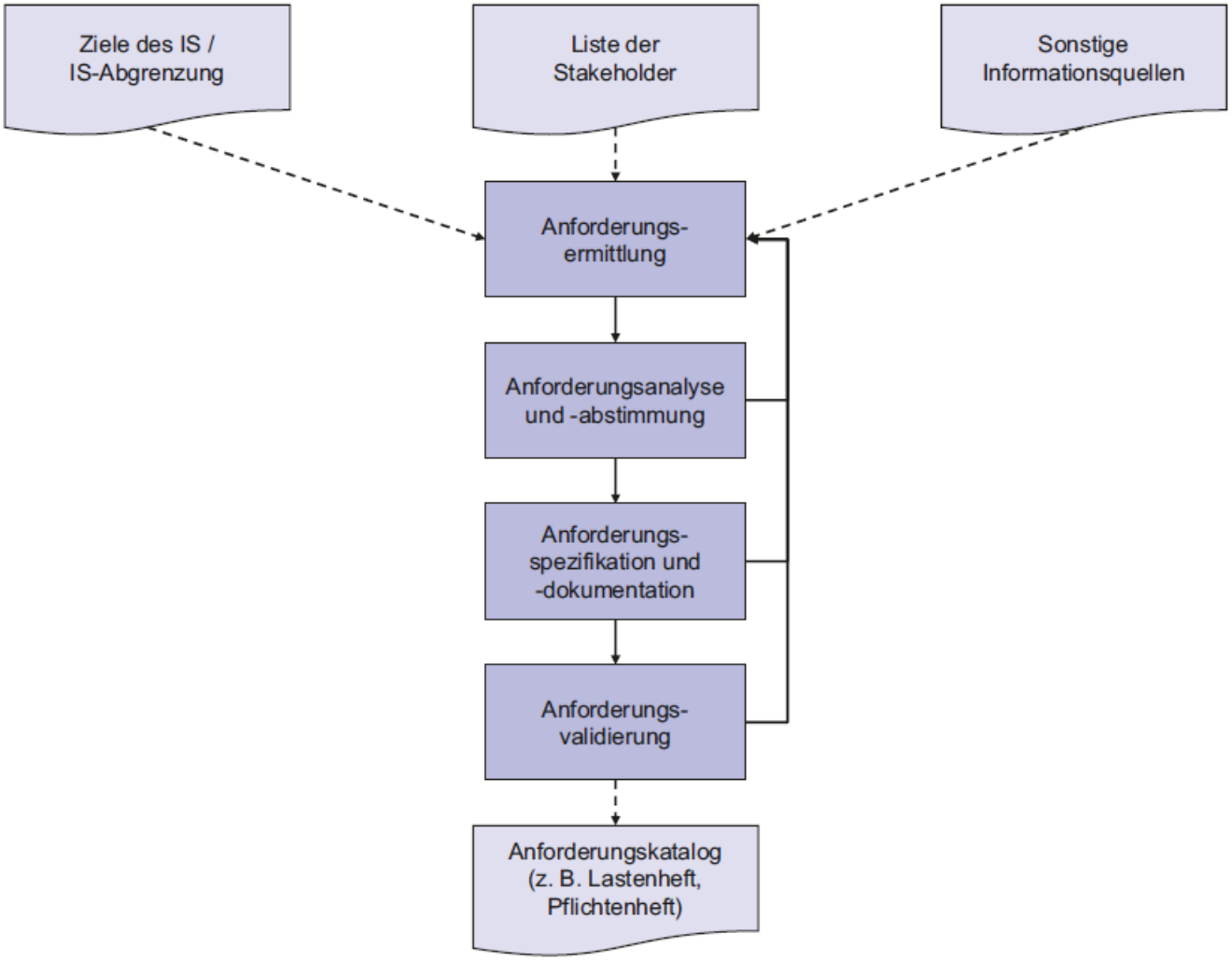

Abb: 13-9: Vorgehen im Requirements Engineering

Informations- und Anforderungsermittlung

Quellen der Informationen:

- Stakeholder
- Organisationshandbücher
- Dokumentation des Alt-Systems
- Verwendete Formulare und Berichte

Techniken zur Ermittlung der Anforderungen:

- Interview
- Schriftliche Befragung
- Workshop
- Beobachtung
- Dokumenten-Review
- Tagebuchtechnik

Iteratives vs. lineares Modell

Eine Iteration bezeichnet eine einzelne Durchführung eines bestimmten vorgegebenen Vorgehens. Bezogen auf die Softwareerstellung bringt eine iterative Vorgehensweise periodisch wiederkehrende, ähnliche Aufgaben der Softwareentwicklung mit sich.

Das iterative, inkrementelle Phasenmodell versucht, die Nachteile des linearen Modells zu umgehen, indem:

- nur in der Projektdefinitionsphase das System in seiner Gesamtheit betrachtet wird,
- danach jedoch Teilsysteme (Inkremente) gebildet und jeweils getrennt nach dem Phasenmodell weiterentwickelt werden. Die Entwicklung des Gesamtsystems erfolgt dann komponentenweise.

Iteratives, inkrementelles Phasenmodell

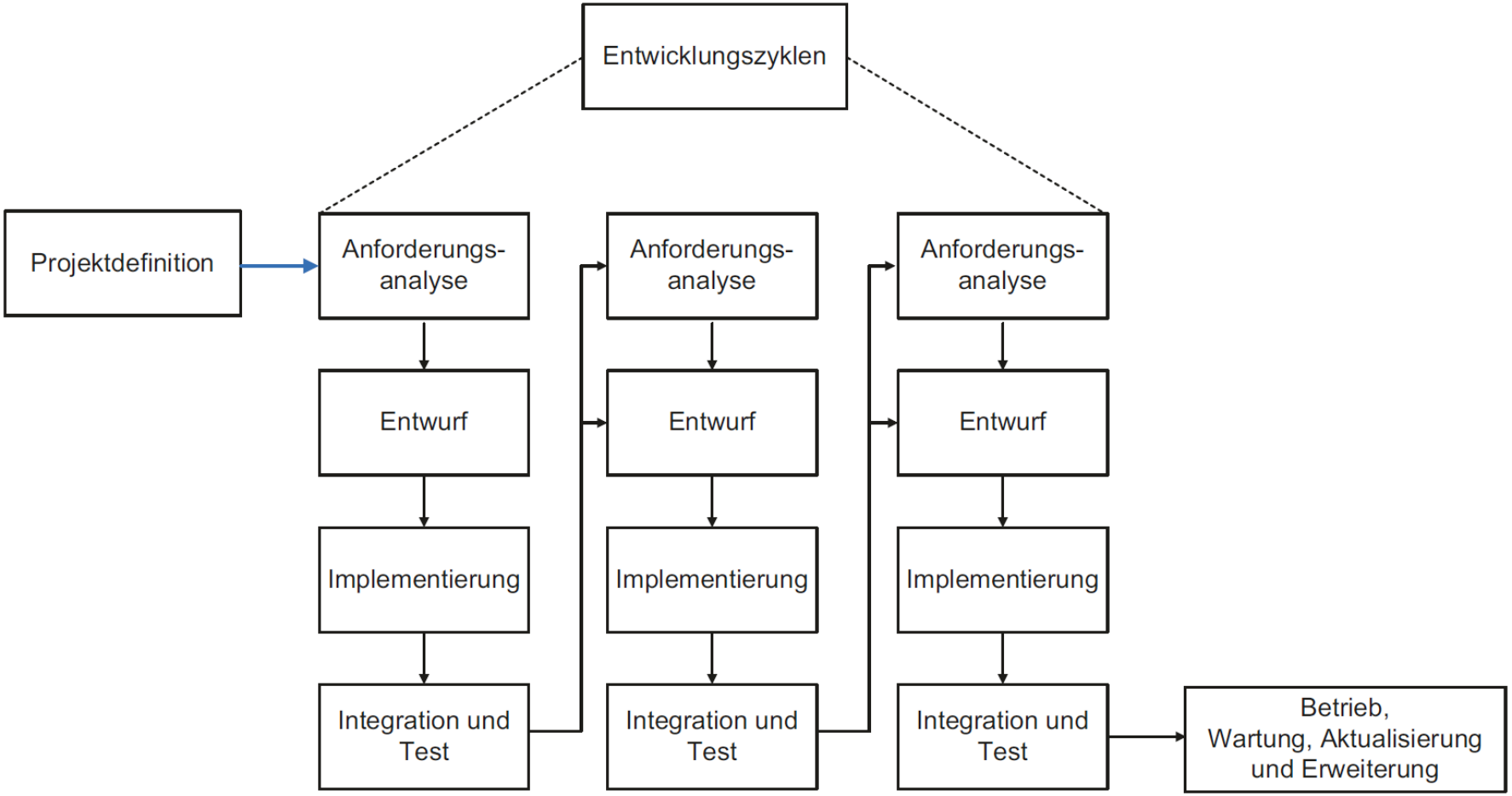

Abb. 13-10: Iteratives, inkrementelles Phasenmodell

Definitionen

Verifikation

Die Verifikation einer Systemkomponente (z. B. eines Softwareprodukts) bezeichnet die Überprüfung der Übereinstimmung zwischen der Systemkomponente und ihrer Spezifikation.

Validation

Die Validation einer Systemkomponente (z. B. eines Softwareprodukts) bezeichnet die Überprüfung der Eignung bzw. des Wertes einer Systemkomponente bezogen auf ihren Einsatzzweck.

V-Modell

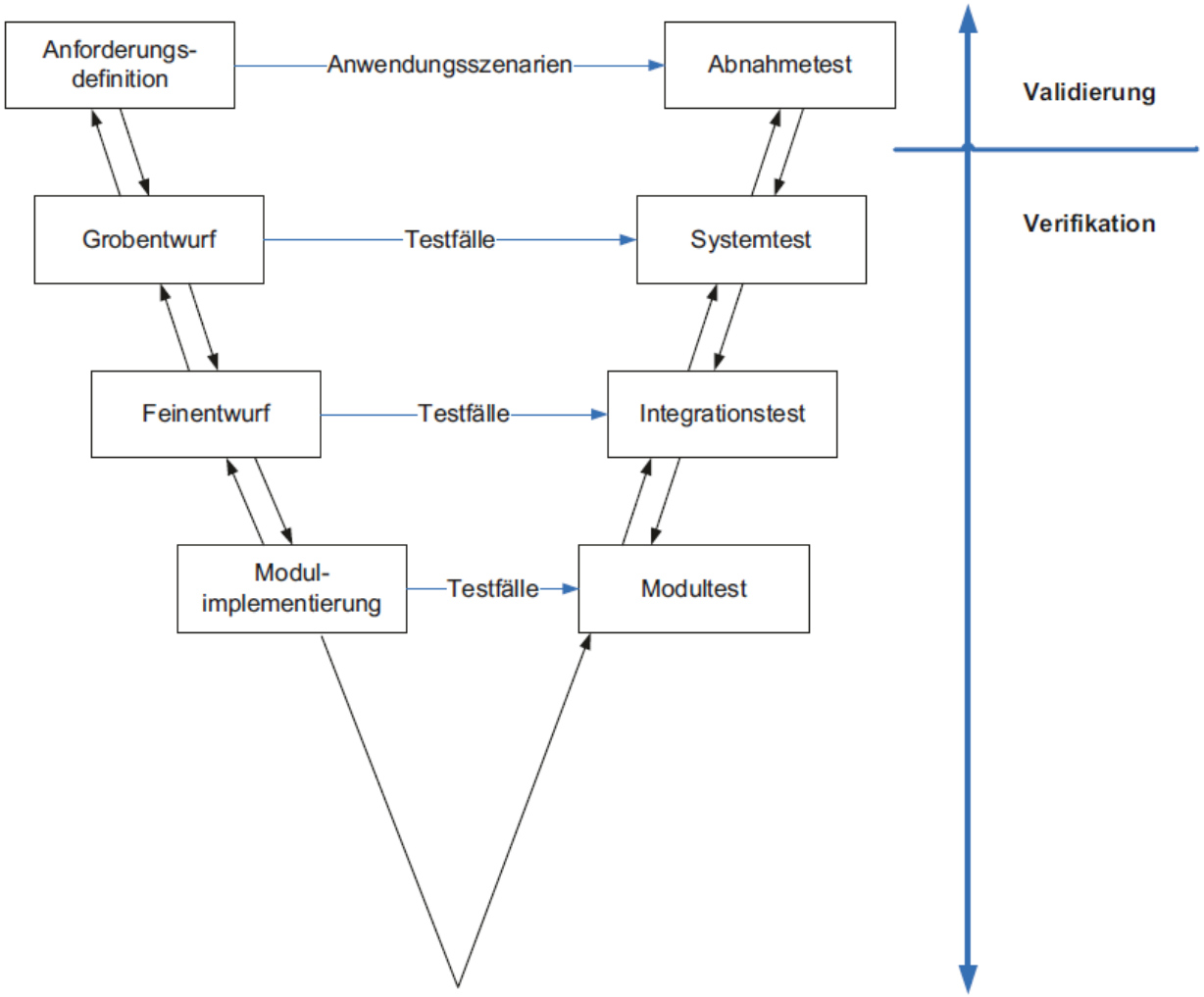

Abb. 13-11: V-Modell nach Boehm (Balzert 1996)

Unterstützung einer Softwareentwicklungsmethode

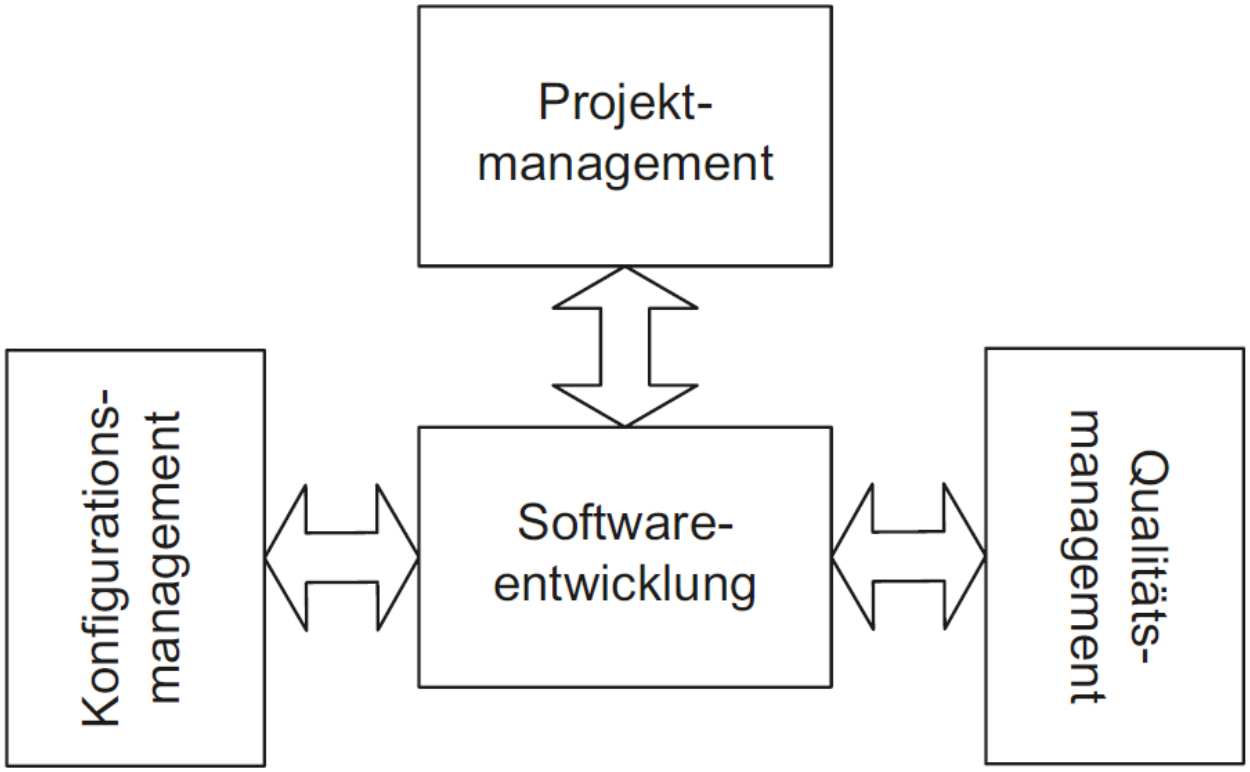

Abb. 13-2: Phasenübergreifende Komponenten
zur Unterstützung einer Softwareentwicklungsmethode

Capability Maturity Model Integration (CMMI)

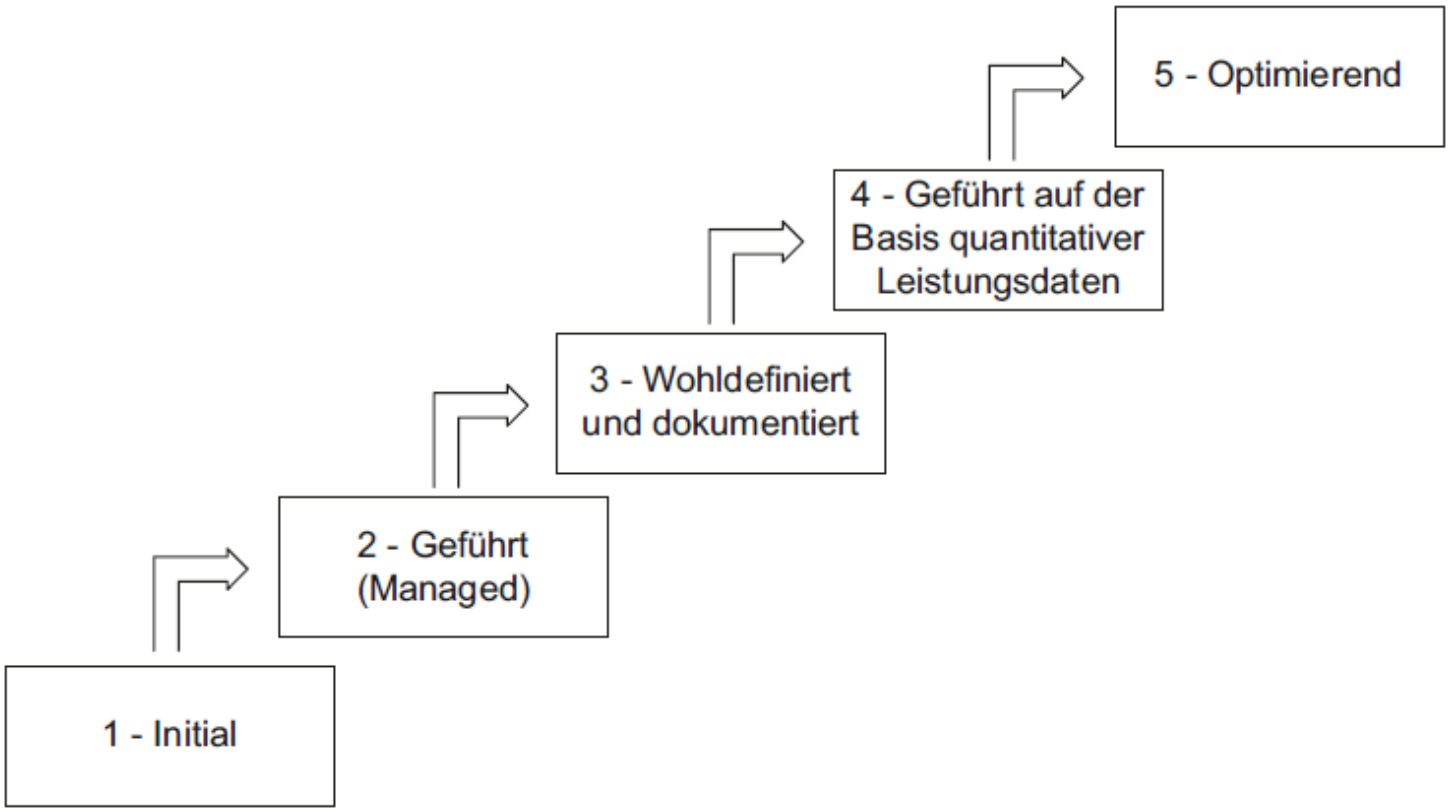

Abb. 13-5: Reifegradstufen gemäß CMMI

Probleme einer umfangreichen Dokumentation in klassischen Modellen

- Die Anforderungen müssen komplett vor Beginn der Entwicklung vorhanden sein.
- Der Auftraggeber hat Probleme, sämtliche Anforderungen konkret zu formulieren.
- Einige Wünsche der Anwender ergeben sich erst nach der ersten Nutzung.
- Von der Spezifikation der Anforderungen bis zum ersten Release vergehen oft Monate oder Jahre.

Prototyp

Ein Prototyp einer Software ist eine frühe ausführbare Version, die bereits die relevanten grundlegenden Merkmale des späteren Produkts aufweist. Prototyping bezeichnet die Vorgehensweise, Prototypen solange iterativ zu entwickeln, zu bewerten und zu verfeinern, bis das endgültige Produkt entstanden ist.

Exploratives Prototyping

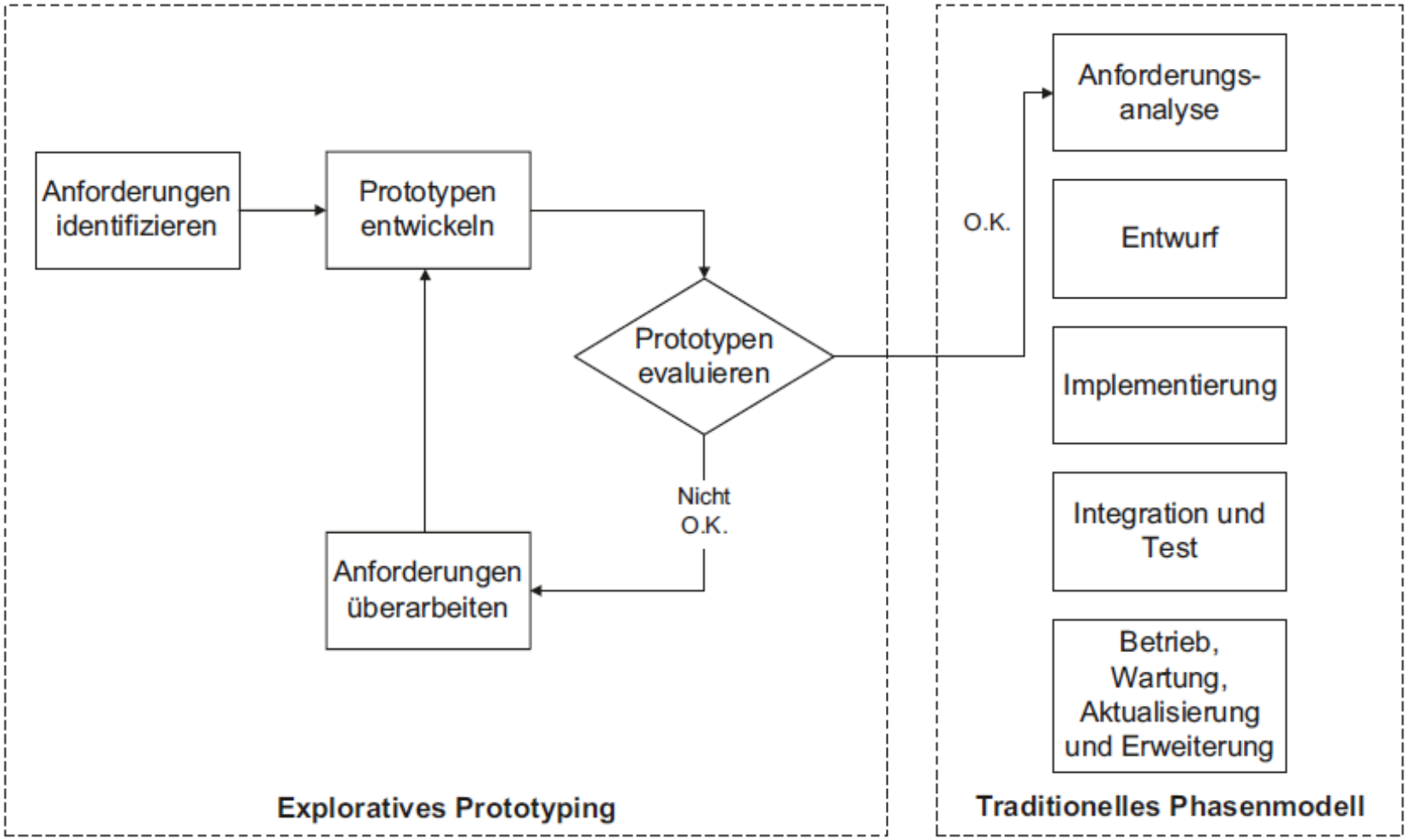

Abb. 13-12: Exploratives Prototyping

Arten des Prototyping

- **Exploratives Prototyping**

  Sukzessive Klärung fachlicher Anforderungen durch wiederholte, schnelle Erzeugung (und Verwerfung) von Prototypen in einer Testumgebung.

- **Experimentelles Prototyping**

  Nachweis der Realisierbarkeit eines Entwurfs vor Einstieg in die Implementierungsphase.

- **Evolutionäres Prototyping**

  Laufende Anpassung des sich in Entwicklung befindlichen Systems.

Grundprinzipien der Agilen Systementwicklung

- Individuen und Interaktion werden über Prozesse und Werkzeuge gestellt,
- eine lauffähige Software wird über eine umfassende Dokumentation gestellt,
- die Zusammenarbeit mit Kunden wird über Vertragsverhandlungen gestellt und
- die Reaktion auf Veränderung wird über Befolgung eines Plans gestellt.

Rollen in Scrum

- *Der **Product Owner**:* Er legt mit dem Team das Ziel und die Rahmenbedingungen der Projektaufgabe fest. Er priorisiert regelmäßig die Funktionalitäten / Anforderungen im Produkt-Backlog und legt dadurch fest, welche Systemteile ihm am Wichtigsten sind. Weiterhin bestimmt er Auslieferungsdatum und Inhalt. Er akzeptiert die Arbeitsergebnisse bzw. weist sie zurück.

- Der ***Scrum Master***: Er ist verantwortlich für die Einhaltung der Scrum Philosophie und Methoden. Er sollte Projekthindernisse entfernen und sicherstellen, dass das Team produktiv zusammen arbeiten kann.

- Das ***Team*** besteht typischerweise aus fünf bis zehn Personen. Alle Teammitglieder sollten Vollzeitmitglieder sein. Das Team arbeitet selbst organisiert über eine Anzahl von Entwicklungszyklen, den sogenannten ***Sprints*** und hat das Recht und die Pflicht, selbst zu entscheiden, wie viele Funktionen/Anforderungen nach dem nächsten Sprint gemäß Sprint-Backlog erreicht sein müssen.

Scrum-Entwicklungszyklus

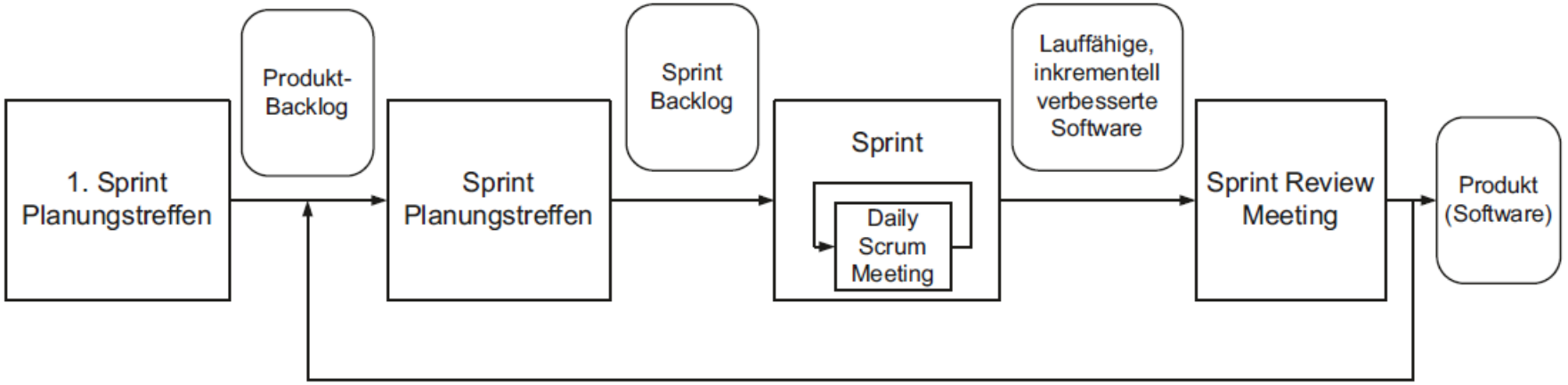

Abb. 13-13: Scrum-Entwicklungszyklus

DevOps im Überblick

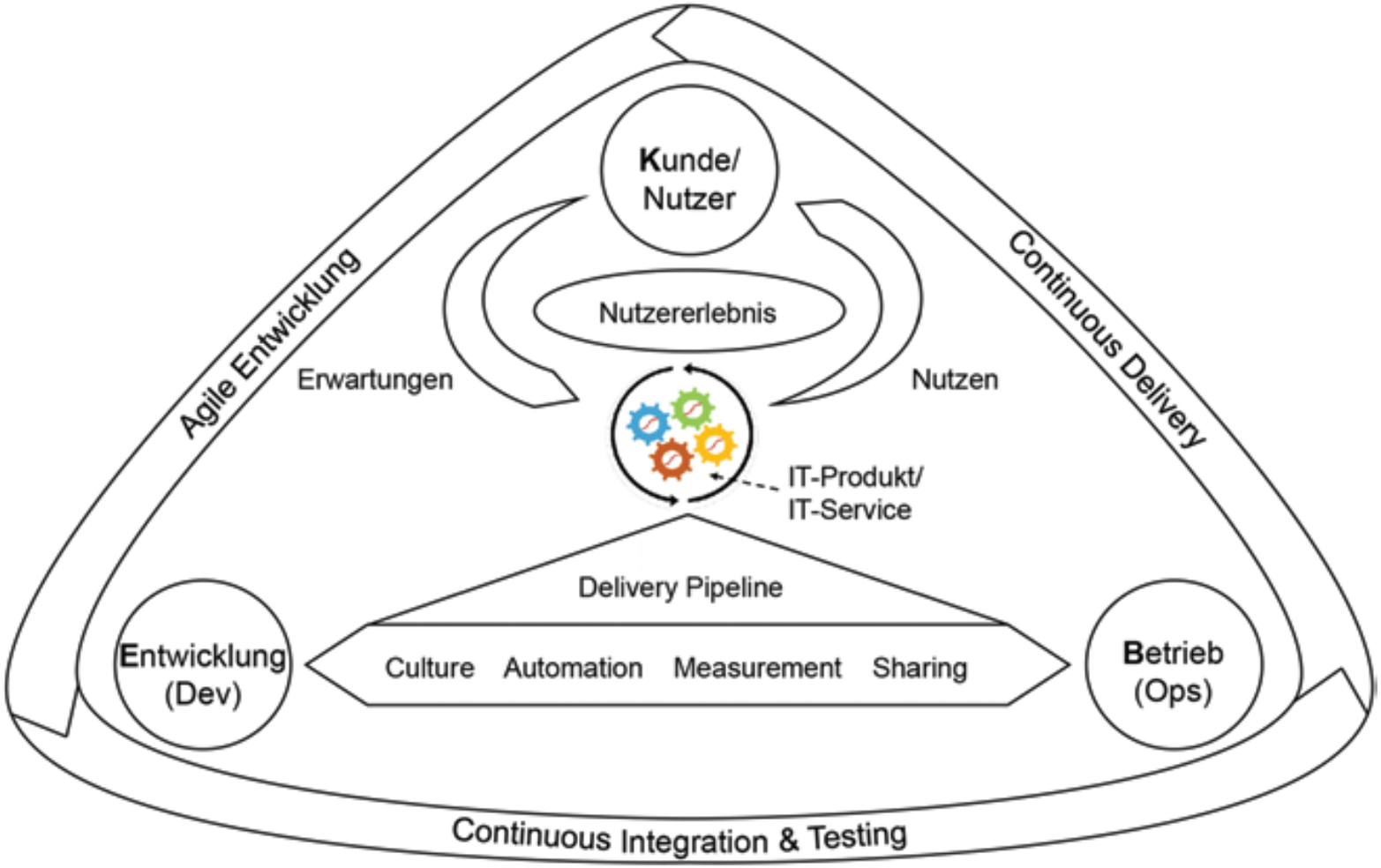

Abb. 13-14: DevOps im Überblick (Alt et al. 2017, S. 28)

Definitionen

Projekt

Ein Projekt ist ein Vorhaben, das im Wesentlichen durch die Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist, wie z. B. Zielvorgabe, zeitliche, finanzielle, personelle oder andere Begrenzungen, Abgrenzung gegenüber anderen Vorhaben und projektspezifische Organisation (DIN 69901-5 2009).

Projektmanagement

Projektmanagement bezeichnet die Gesamtheit von Führungsaufgaben, -organisationen, -techniken und -mitteln für die Abwicklung sowohl aller Projekte als auch eines einzelnen Projekts (DIN 69901-5 2009).

Phasen eines Projekts

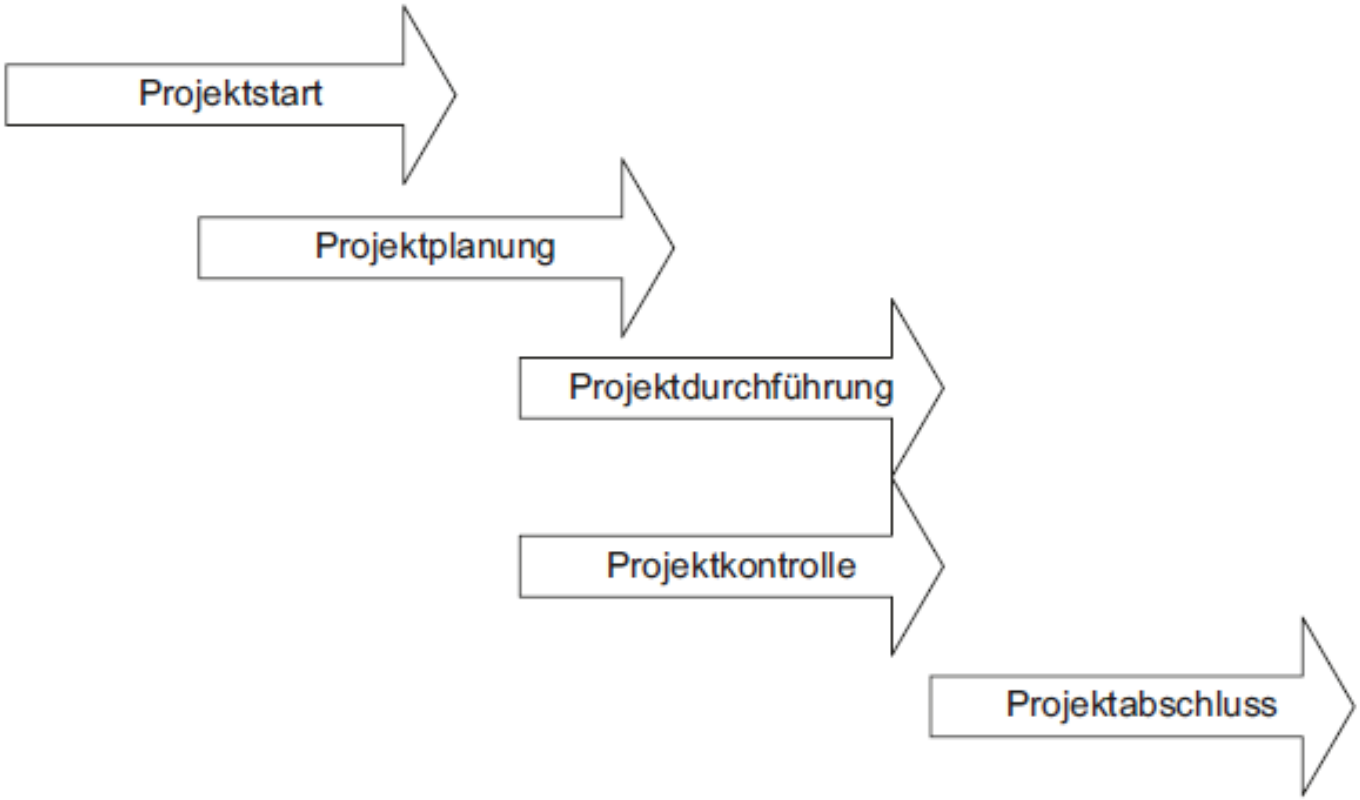

Abb. 13-16: Phasen eines Projekts

Projektorganisation

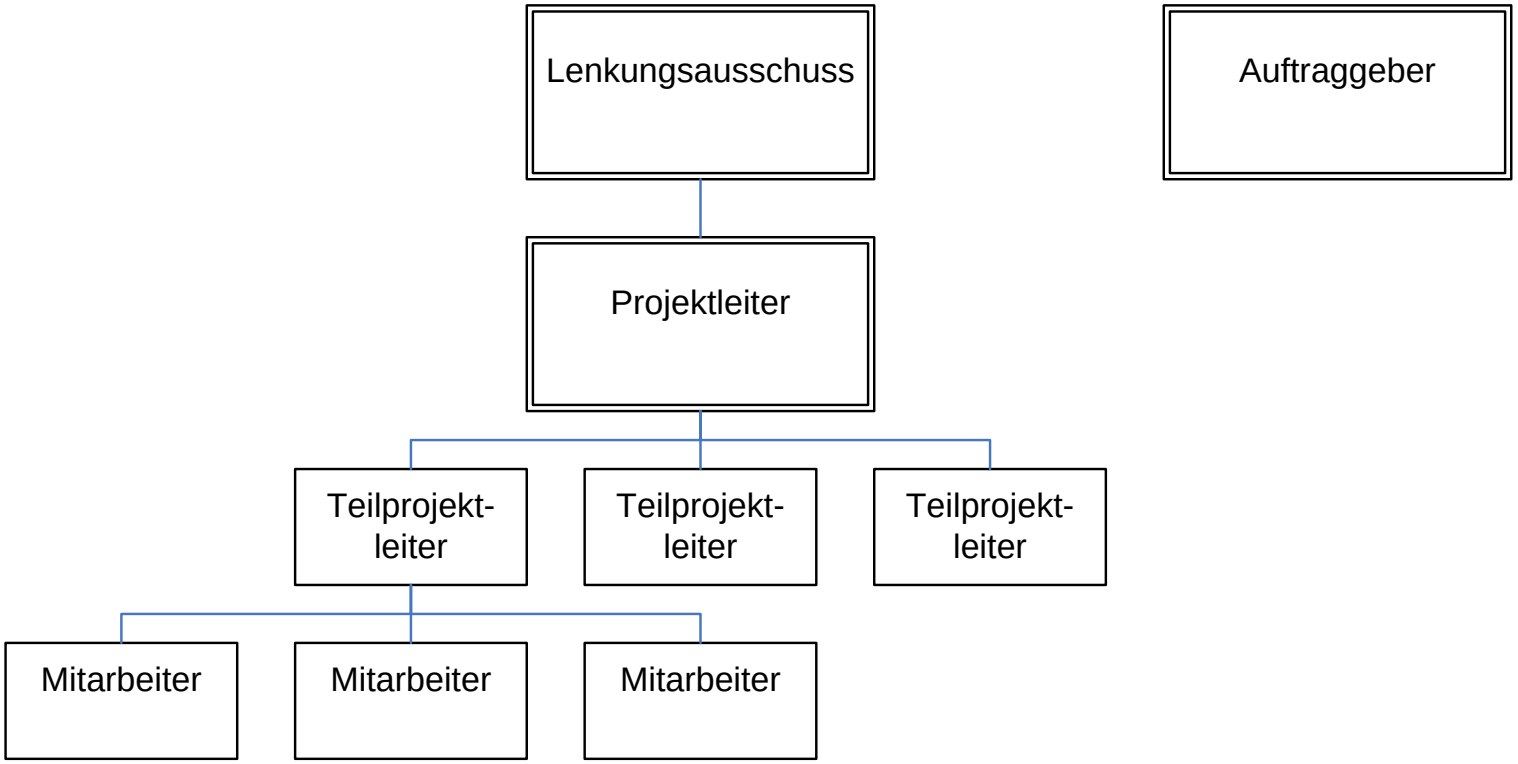

Abb. 13-17: Projektorganisation

Projektstart

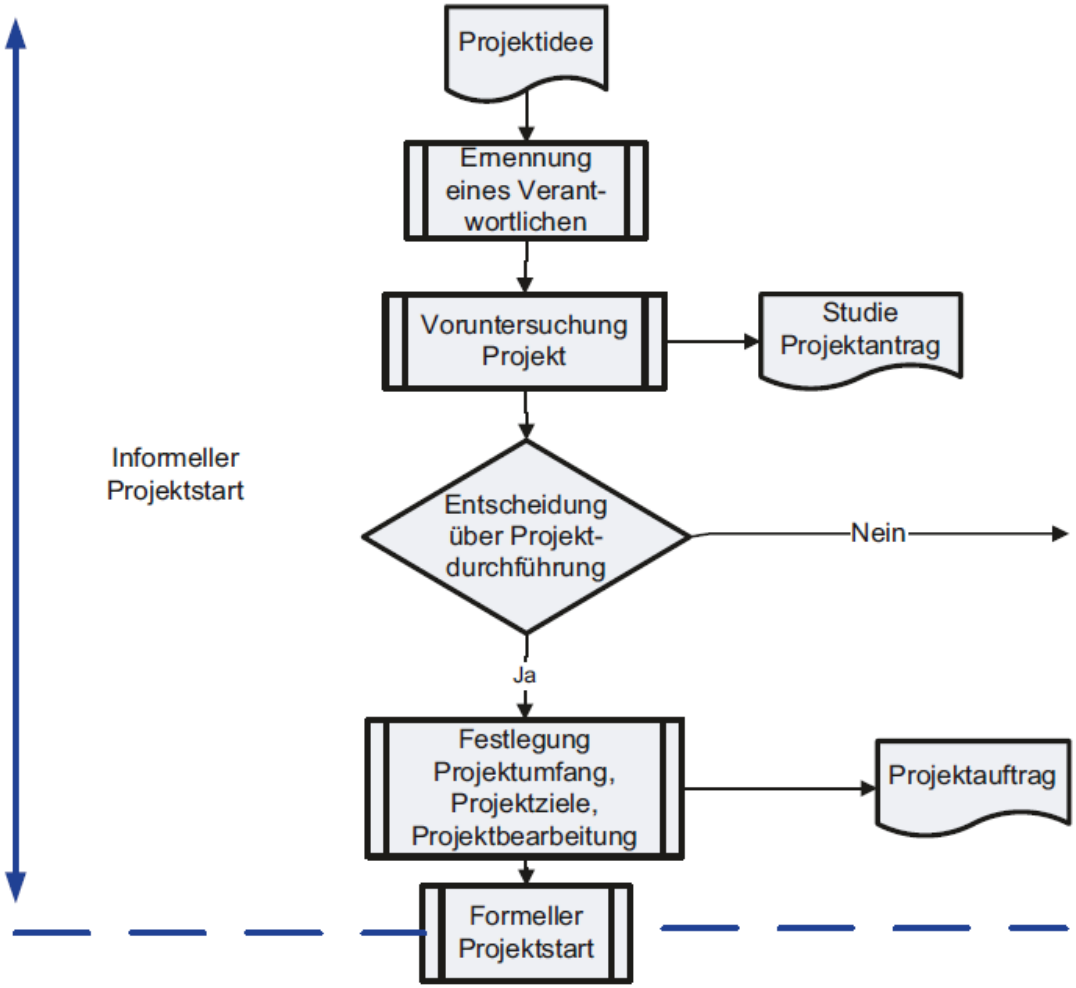

Abb. 13-18: Informeller – Formeller Projektstart

Schritte der Projektplanung

- Projektstrukturierung
- Meilensteinplanung
- Einsatzmittelbedarfsplanung
- Ablauf- und Terminplanung
- Kostenplanung
- Planoptimierung
- Risikomanagementplanung

Beispiel eines produktorientierten Projektstrukturplans (PSP)

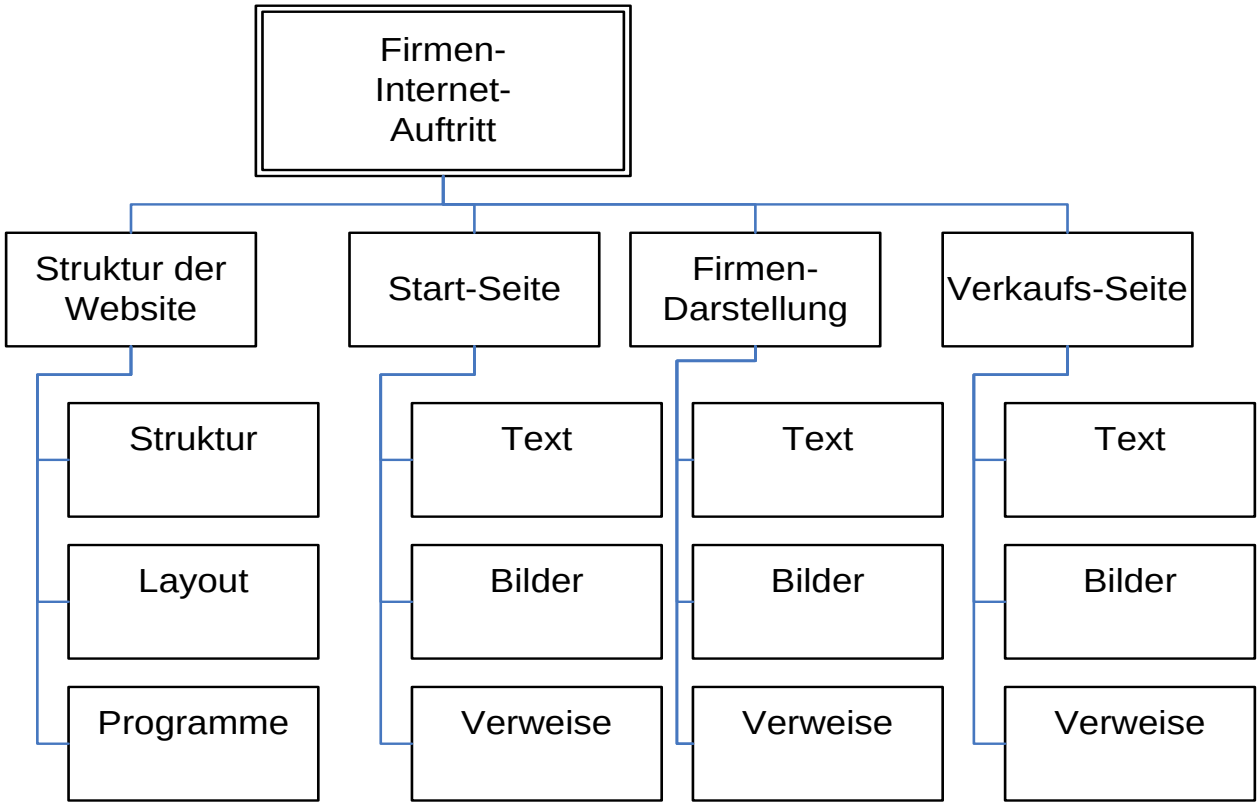

Abb. 13-19: Beispiel eines produktorientierten PSP

Beispiel eines prozessorientierten PSP

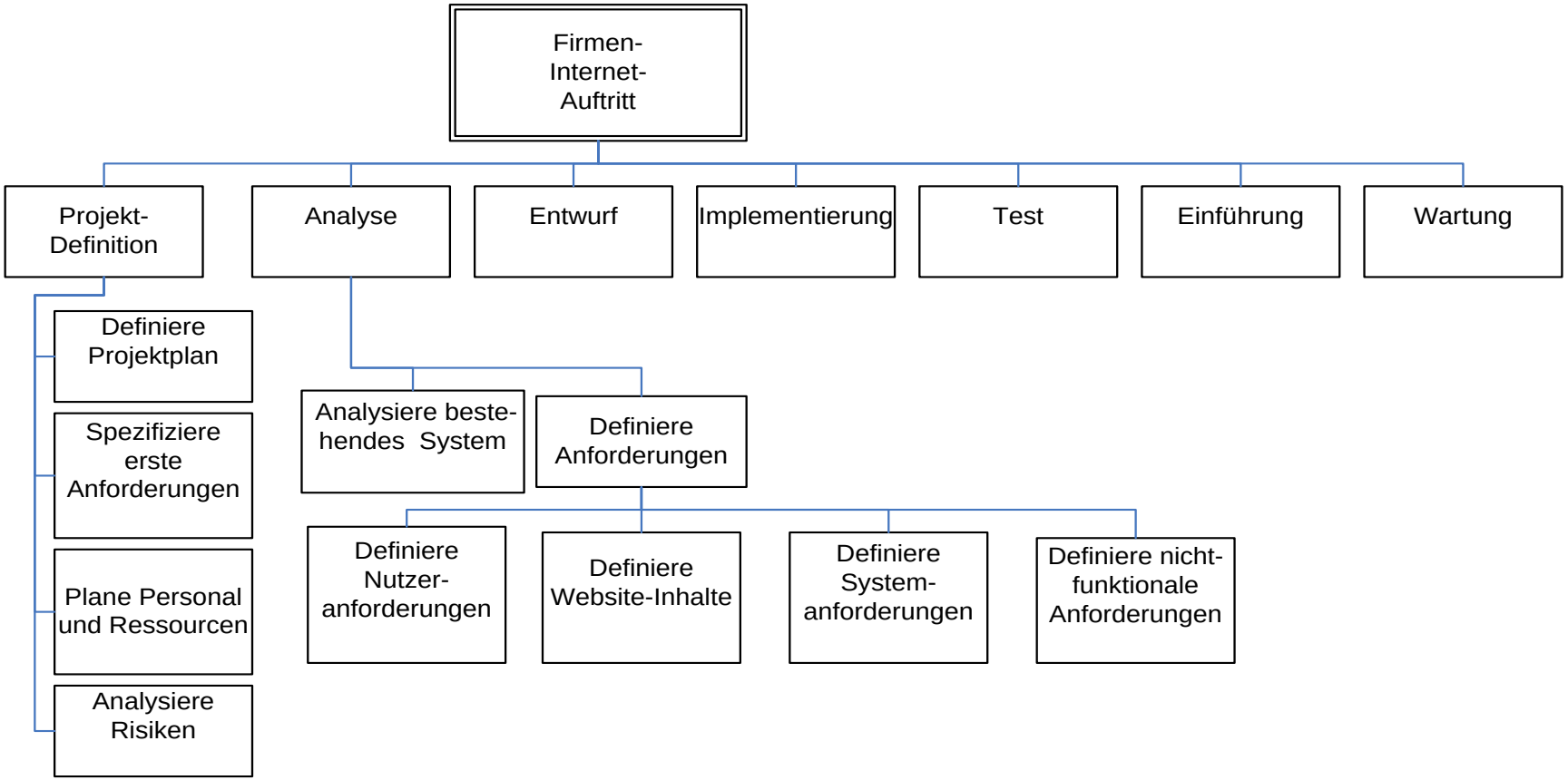

Abb. 13-20: Beispiel eines prozessorientierten PSP

Tabelle zur Aufwandsschätzung

Arbeitspakete	Bearbeiter	Personaleinsatz	Aufwand	Dauer
Definiere Projektplan	Fr. Herzog	50%	4 PH	8 H
Spezifiziere erste Anforderungen	Hr. Kasse Fr. Herzog Hr. Zog	100% 100% 100%	90 PH	30 H
Plane Personal und Ressourcen	Fr. Herzog	50 %	4 PH	8 H
Analysiere Risiken	Hr. Kasse	20 %	10 PH	50 H

Abb. 13-21: Tabelle zur Aufwandsschätzung

Bei der Aufwandsschätzung zu beachten

- Aufwandsschätzungen zu neuen Themen fallen erfahrungsgemäß zu niedrig aus.
- Zu niedrige Schätzungen sind auch für Mitarbeitende typisch, die nur selten bewusst Aufwände schätzen.
- Viele Mitarbeiter trennen Aufwand und Dauer nicht scharf. Der Aufwand hängt vom zu erbringenden Arbeitsinhalt ab, ist also nicht direkt beeinflussbar. Die Dauer kann dagegen durch mehr oder weniger intensives Arbeiten an einem Arbeitspaket beeinflusst werden.
- Auch Projektmanagement verursacht Aufwand. Dieser wird jedoch häufig nicht in die Planung mit einbezogen. Analog verhält es sich mit Aufwänden für die Qualitätssicherung.
- Es sollten erfahrene Mitarbeiter zur Aufwandsschätzung hinzugezogen werden.

Balkendiagramm

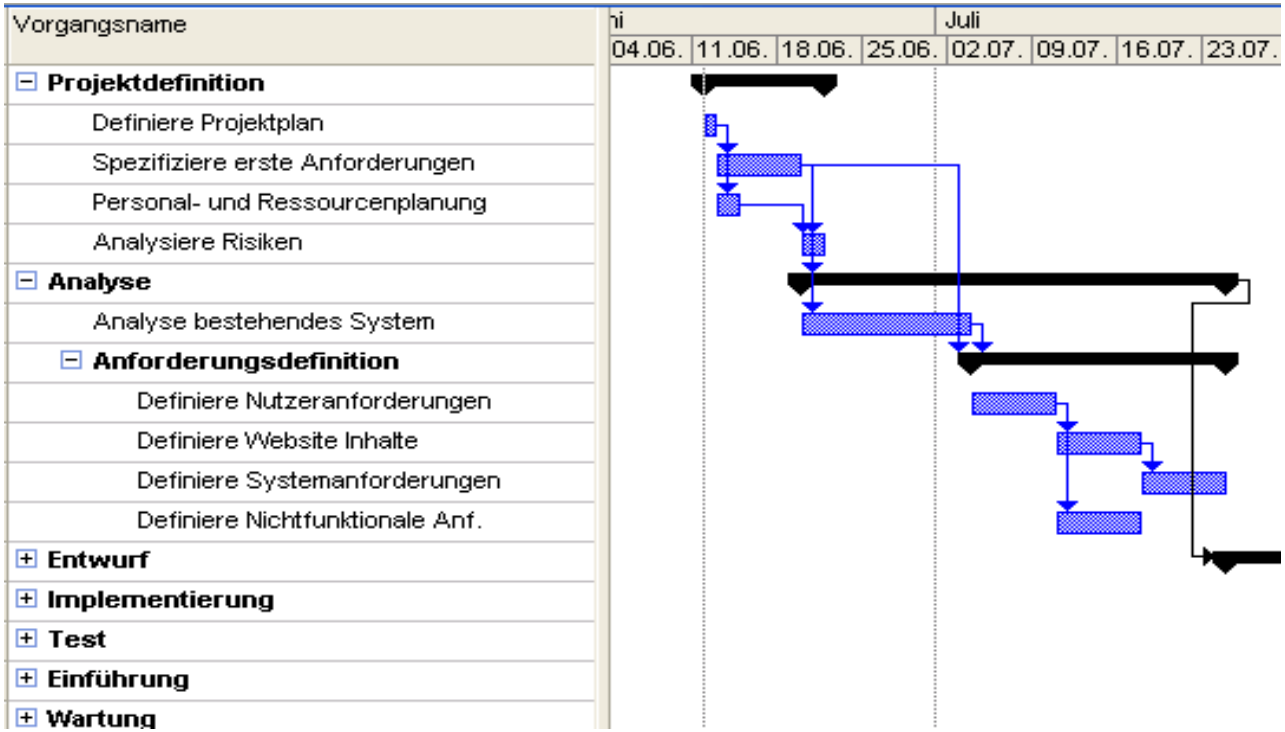

Abb. 13-22: Balkendiagramm

Netzplan

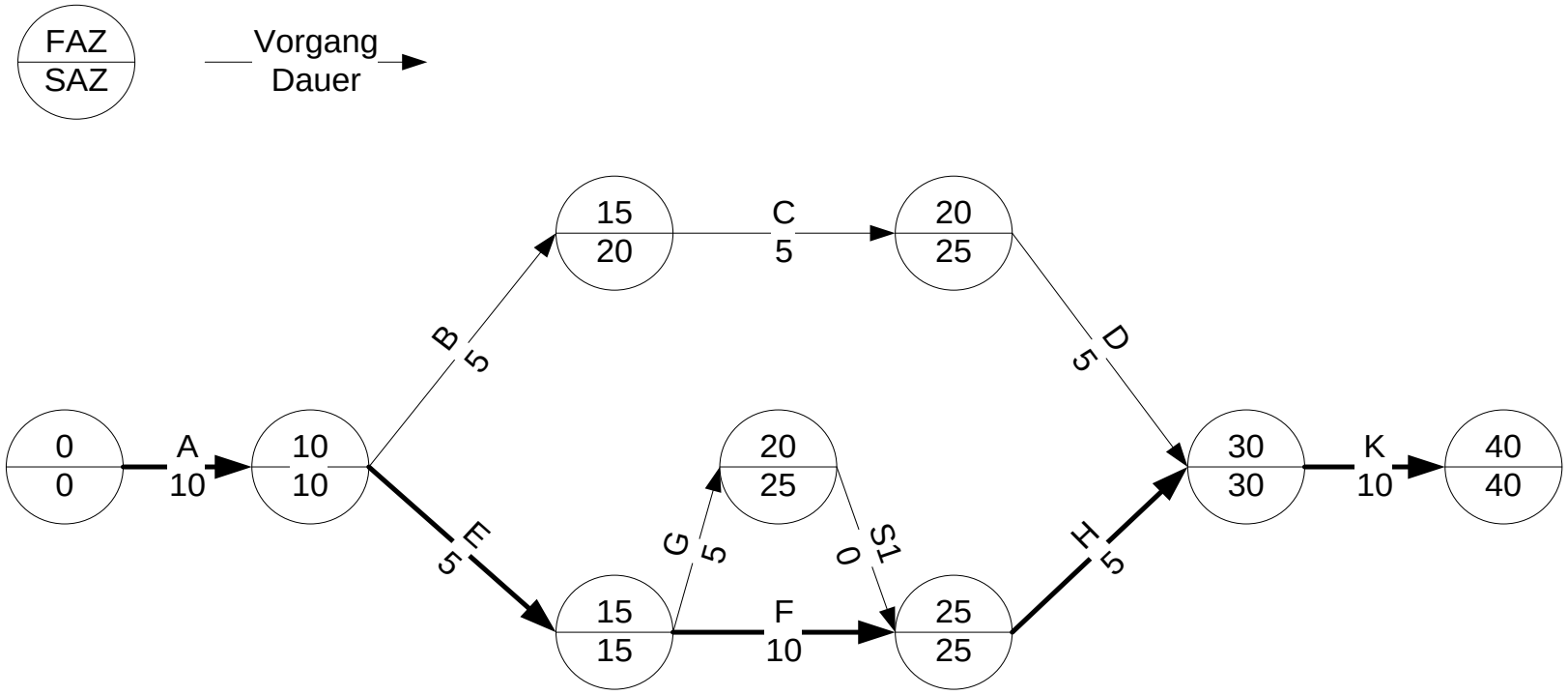

Abb. 13-23: Netzplan

FAZ: Frühster möglicher Startzeitpunkt
SAZ: Spätester möglicher Startzeitpunkt

Zielgrößen der Projektplanung

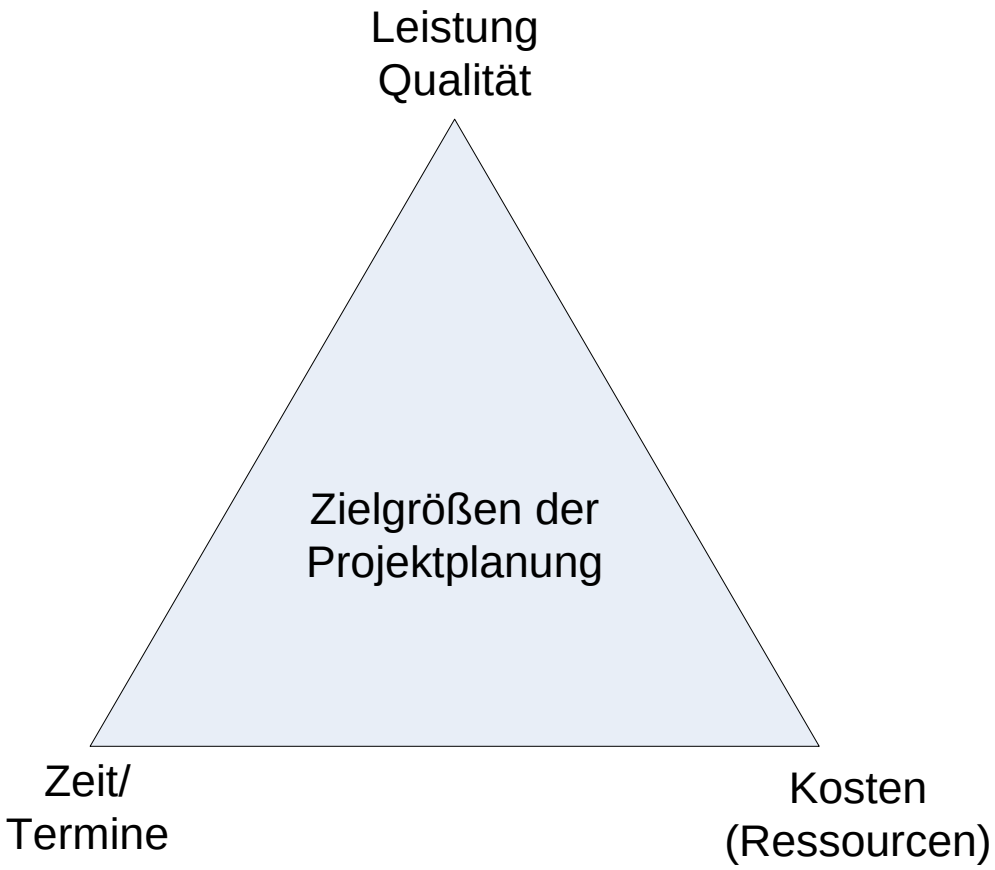

Abb. 13-24: Zielgrößen der Projektplanung

Definitionen

Risiko

Ein Risiko ist ein unbestimmtes Ereignis, dessen Eintreten eine Gefahr für den Projekterfolg, d. h. für die zu erbringenden Leistungen, die Terminplanung und/oder die geplanten Einsatzmittel, darstellt. Ein Risiko hat eine Eintrittswahrscheinlichkeit sowie eine Auswirkung auf das Projekt bzw. das Unternehmen.

Risikomanagement

Die Aufgabe des Risikomanagements ist es, Faktoren, die eine Gefahr für den Projekterfolg (die im Projektauftrag definierte Leistung in geplanter Zeit mit den geplanten Ressourcen im vorgegebenen Budget zu erbringen) darstellen, zu identifizieren, zu bewerten und entsprechende Gegenmaßnahmen vorzubereiten bzw. einzuleiten.

Grundaufgaben des Risikomanagement-Prozesses

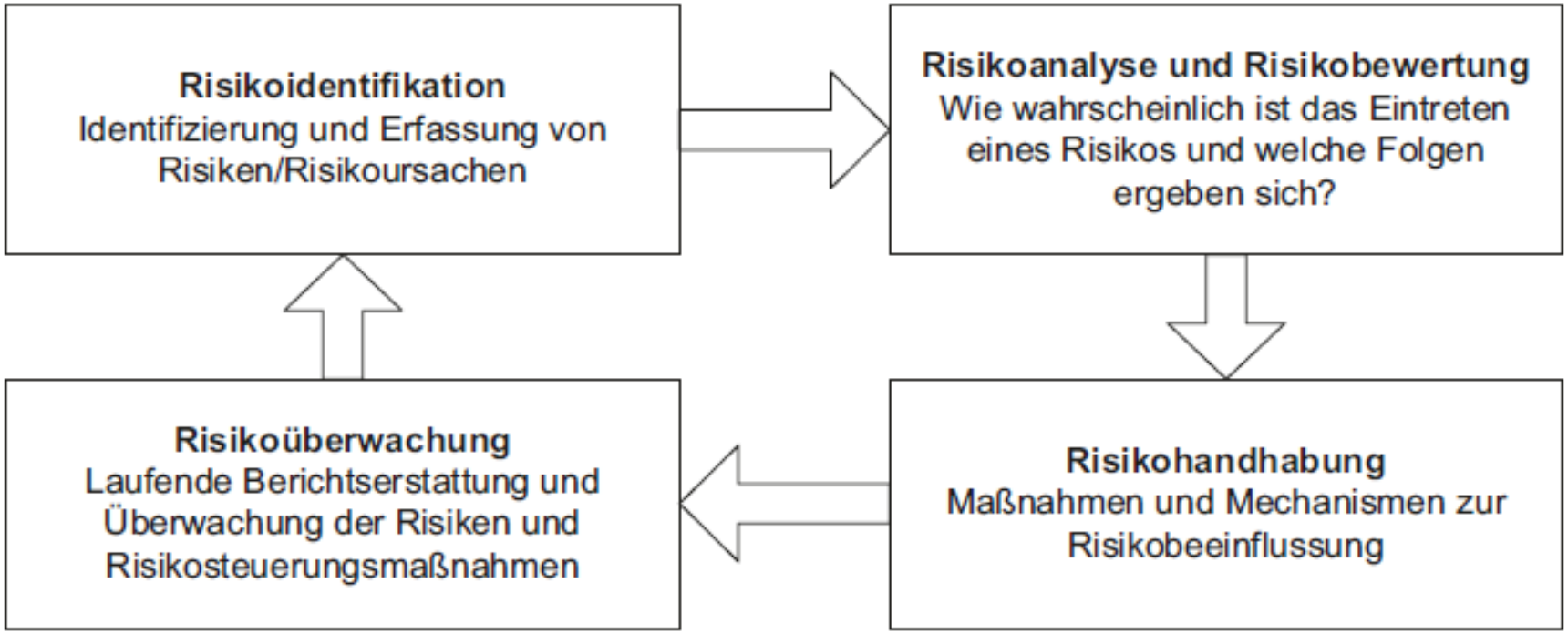

Abb. 13-25: Grundaufgaben des Risikomanagement-Prozesses [Wallmüller 2004]

Definition: Qualität

Gemäß der Norm DIN EN ISO 9000:2015 wird Qualität als derjenige Grad bezeichnet, in dem ein Satz inhärenter Merkmale eines Objekts Anforderungen erfüllt.

Qualitätsmerkmale

- **Unmittelbare Gebrauchsfähigkeit:**
  - Benutzerebene: benutzerfreundlich, zuverlässig, korrekt, robust
  - Administratorenebene: Integrität, Datensicherheit, Wiederherstellbarkeit, Effizienz
- **Zukünftige Gebrauchsfähigkeit:**
  - Allgemeingültigkeit der Problemlösung
  - Wiederverwendbarkeit
  - Portabilität
  - Wartungsfreundlichkeit

Konfigurationsmanagement

Das *Konfigurationsmanagement* (KM) ist „eine Managementdisziplin, die organisatorische und verhaltensmäßige Regeln auf den Produktlebenslauf einer Konfigurationseinheit von ihrer Entwicklung über Herstellung und Betreuung anwendet.“ [DIN 10007 2003].

Eine Konfigurationseinheit meint eine beliebige Kombination aus Hardware, Software oder Dienstleistung im Zusammenhang mit einem IS.

Teil 4, Kapitel 14: Individualentwicklung von Software

Definition: Objekt

Ein Objekt ist eine eindeutig identifizierbare Instanz einer Klasse. Objekte sind Aggregate aus lokalen Daten und lokalen Operationen, deren interne Strukturen nach außen nicht transparent sind. Lediglich als öffentlich deklarierte Operationen können über eine entsprechende Schnittstelle ausgelöst werden und auf lokale Daten und lokale Operationen zugreifen. Objekte kommunizieren miteinander durch Austausch („das Senden“) von Nachrichten. Diese Nachrichten entsprechen dem Aufruf der als öffentlich deklarierten Operationen.

Klasse mit Objekten

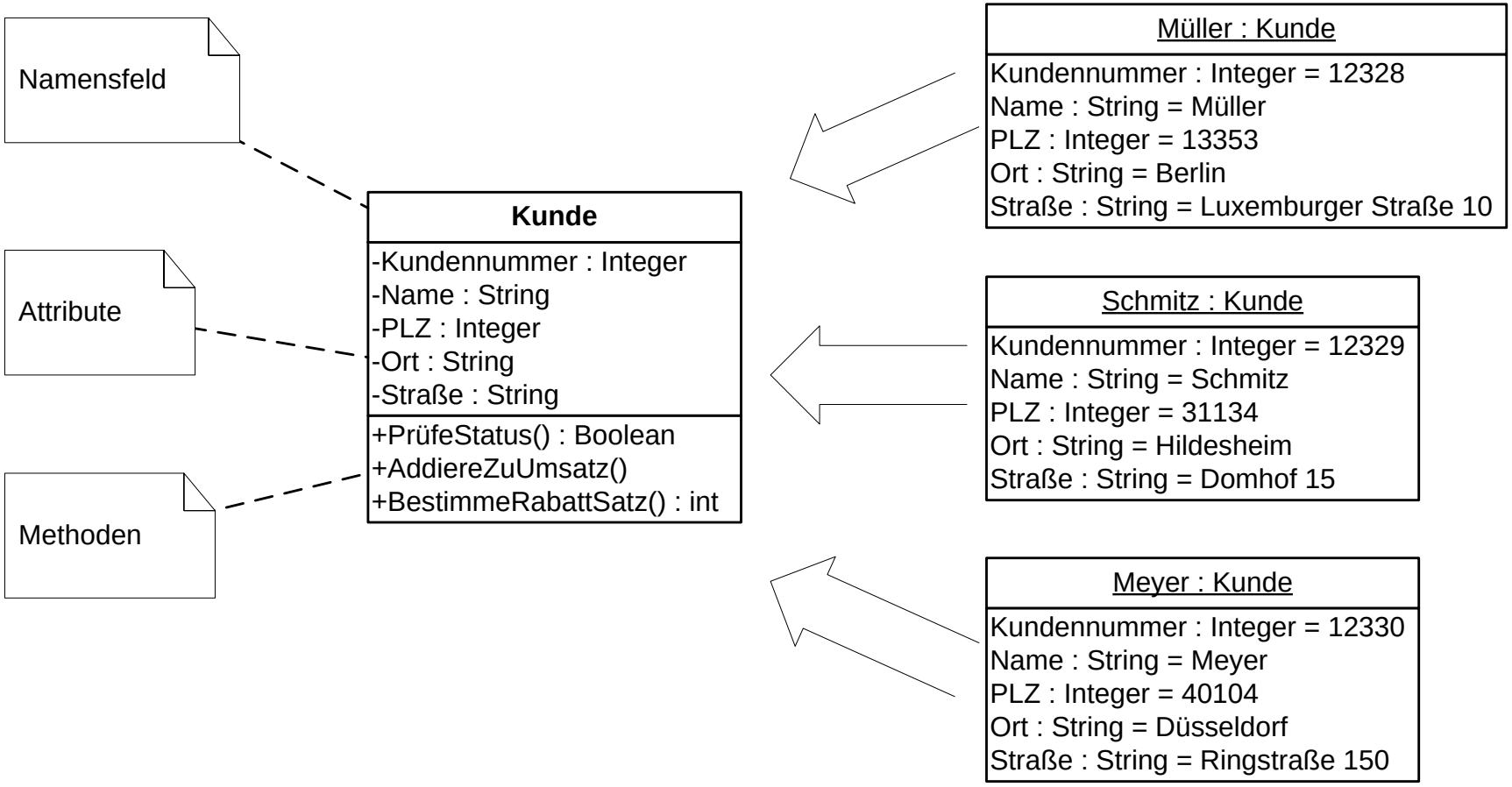

Abb. 14-1: Klasse Kunde mit den Objekten Meyer, Schmitz und Müller

Prinzipien der Objektorientierung

- Kapselung
- Vererbung
- Wiederverwendung

Auftragsmanagement-System

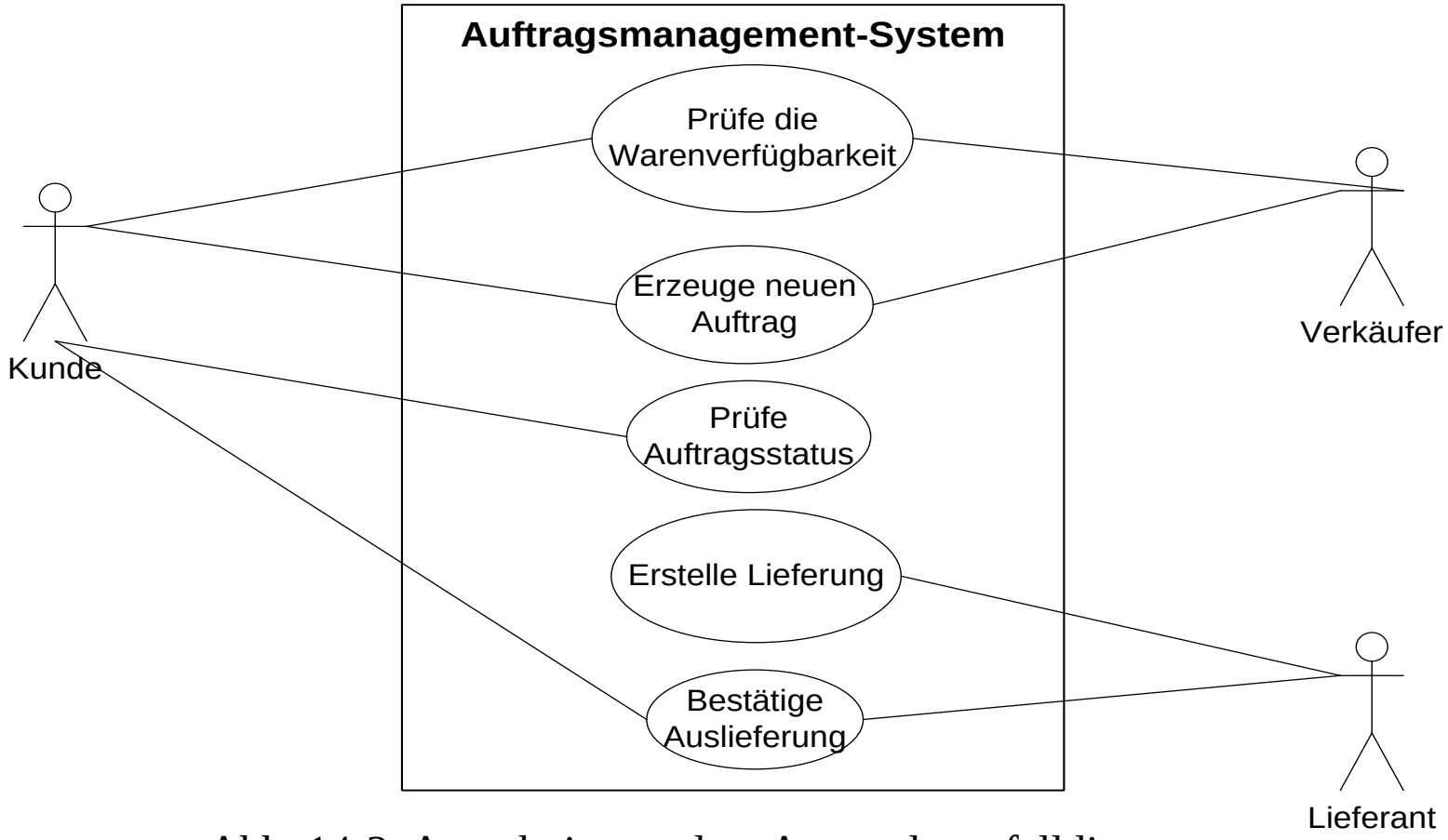

Abb. 14-2: Ausschnitt aus dem Anwendungsfalldiagramm „Auftragsmanagement-System“

Formulare zur Beschreibung von Anwendungsfällen

- Verweis auf den Anwendungsfall im Diagramm
- Verweis auf Anforderungen im Fachkonzept
- Vorbedingung zur Durchführung des Anwendungsfalls
- Nachbedingung: Was für Auswirkungen hatte der Anwendungsfall auf das System?
- Akteure
- Auslösendes Ereignis
- Kurzbeschreibung
- Ablauf, ggf. im Dialog Akteur - System

Sechs Schritte zum Klassendiagramm

- Klassen identifizieren,
- Assoziationen identifizieren,
- Attribute identifizieren,
- Vererbungsstrukturen identifizieren,
- Assoziationen verfeinern und
- Attribute spezifizieren.

Klassendiagramm mit Vererbung

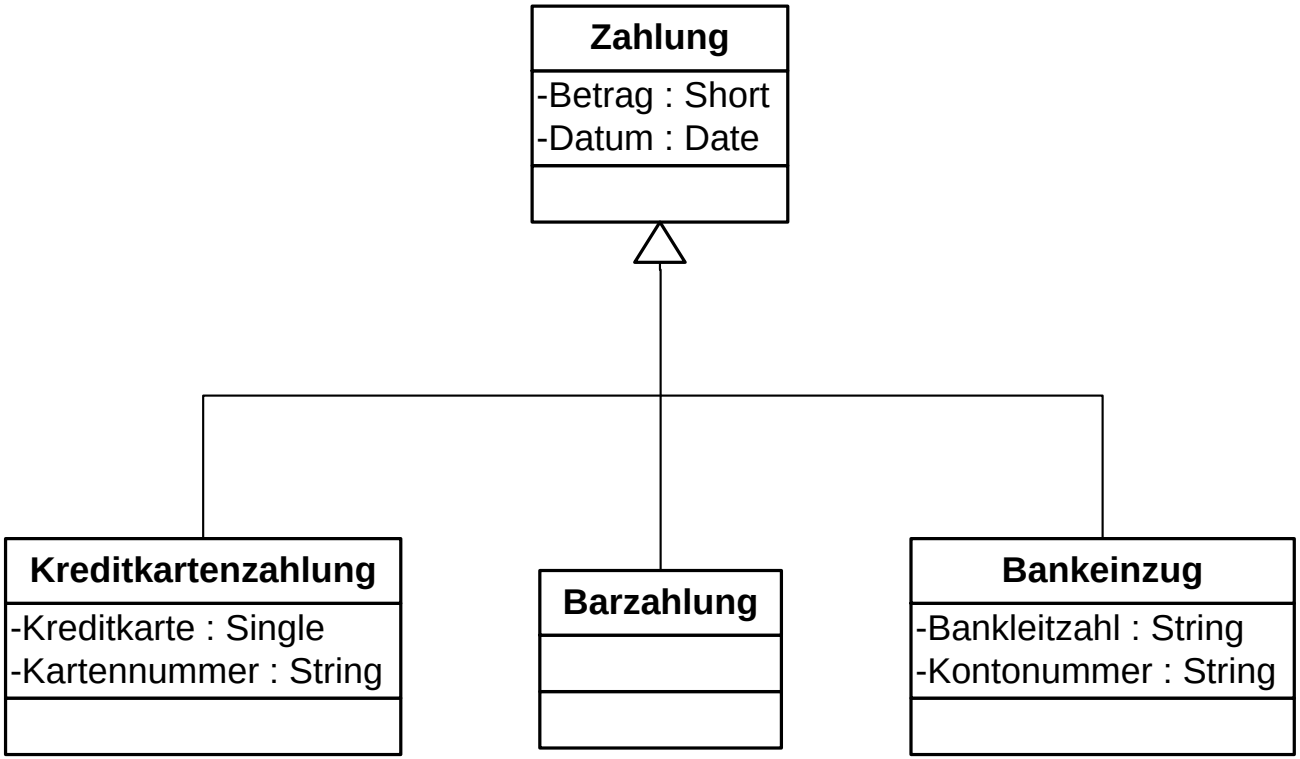

Abb. 14-3: Klassendiagramm mit Vererbung

Komposition und Assoziation

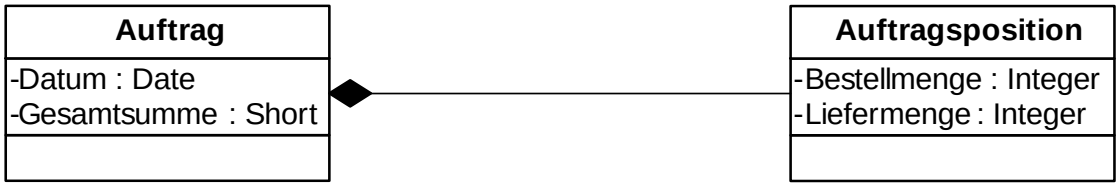

Abb. 14-4: Komposition

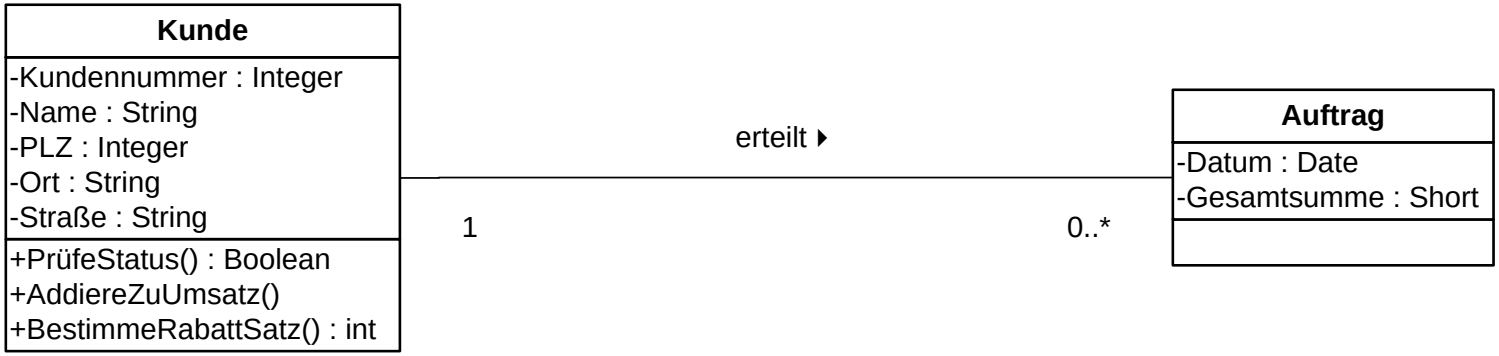

Abb. 14-5: Assoziation

Klassendiagramm

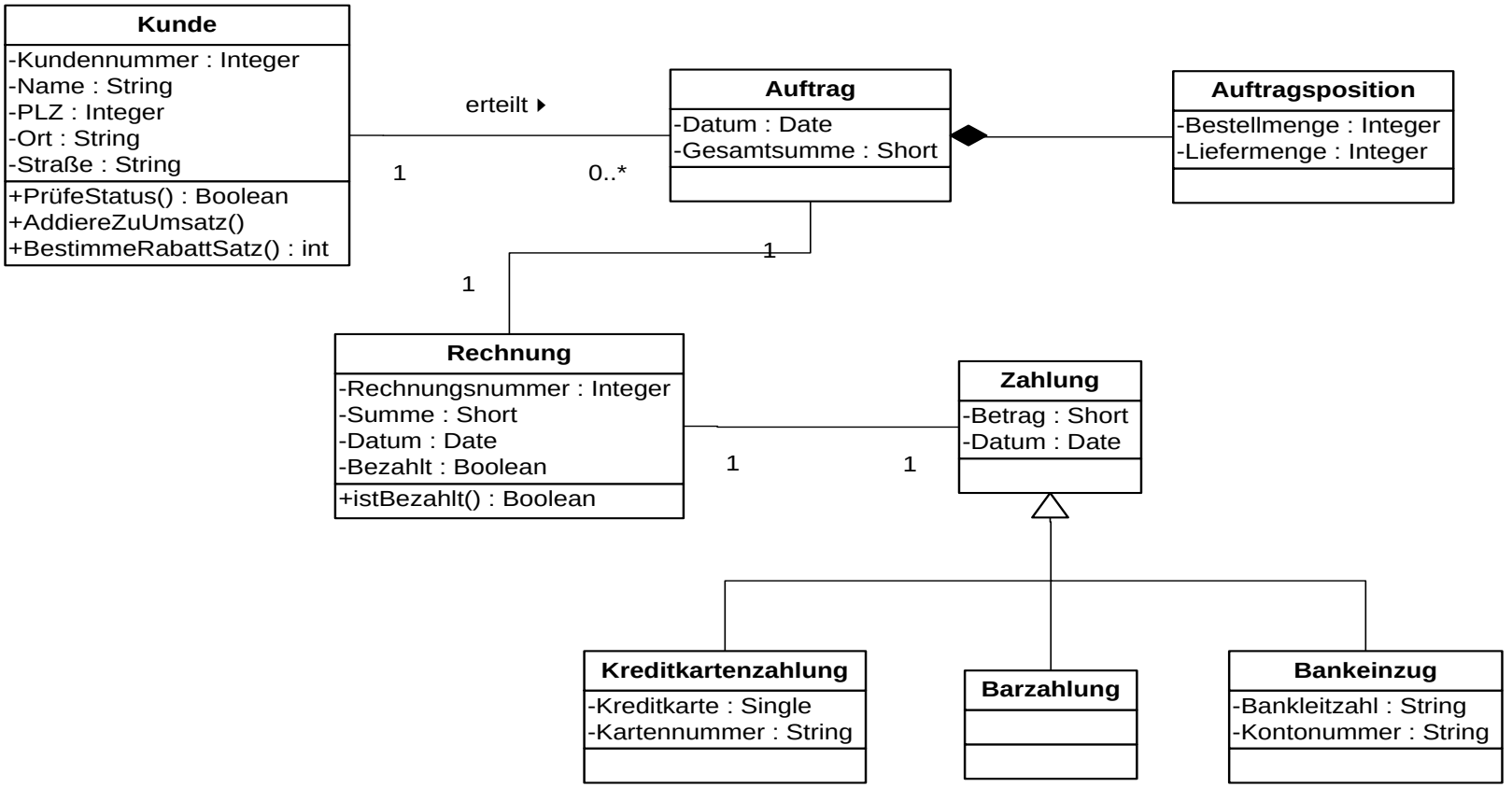

Abb. 14-6: Vereinfachtes Klassendiagramm

Sequenzdiagramm

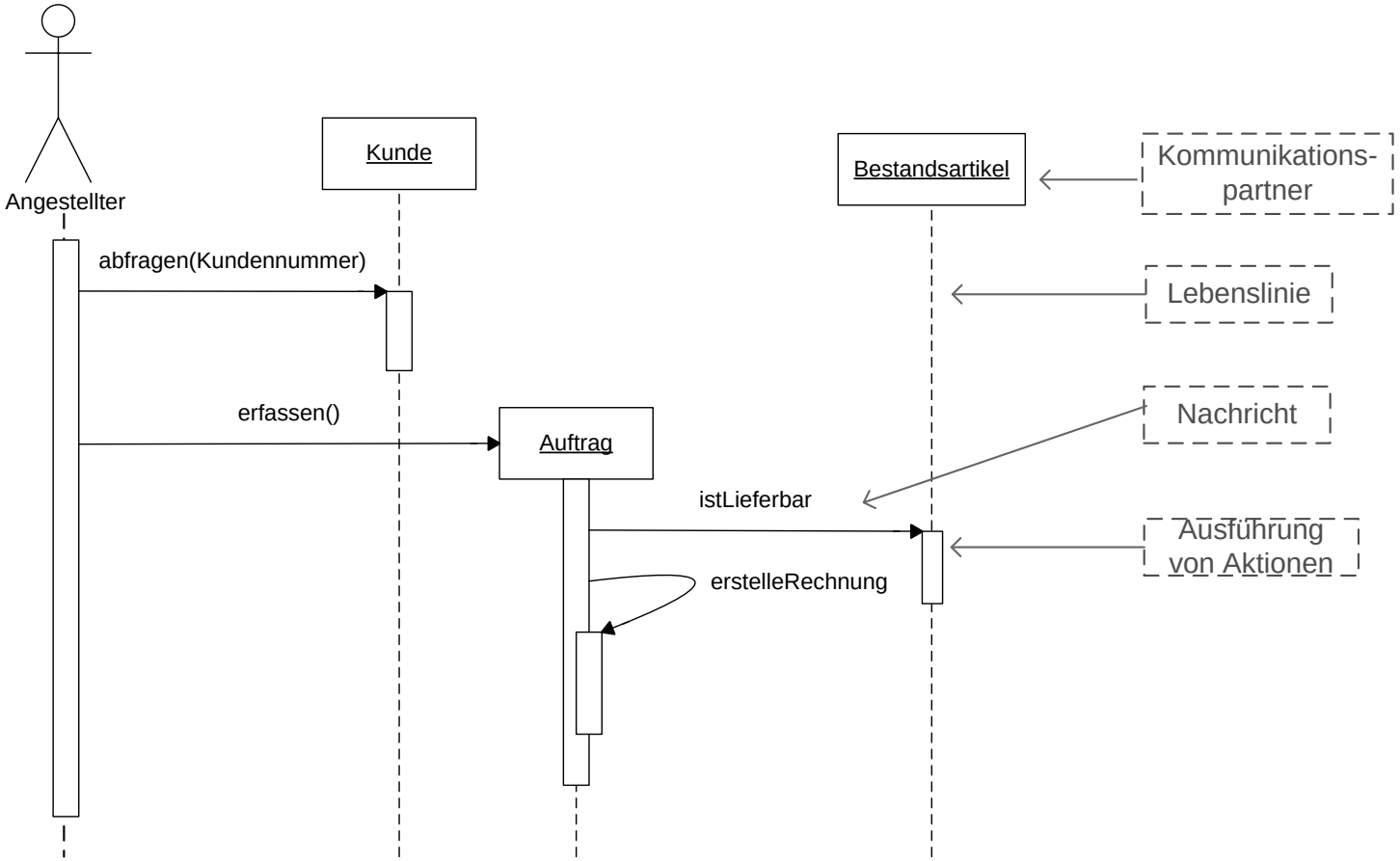

Abb. 14-8: Vereinfachtes Sequenzdiagramm zur Auftragsbearbeitung von einem Kunden

Überarbeitungsschritte der Klassendiagramme

- Identifiziere alle Klassen, die Teil der Softwarelösung sind.
- Zeichne alle diese Klassen in das neue Klassendiagramm.
- Kopiere die Attribute in dieses Klassendiagramm.
- Analysiere die Interaktionsdiagramme und füge entsprechend Methoden hinzu.
- Füge Typinformationen zu Attributen und Methoden hinzu.
- Füge Assoziationen hinzu, um die notwendige Sichtbarkeit von Attributen/Klassen zu unterstützen.
- Füge Pfeile hinzu, um die Sichtbarkeit der Attribute anzuzeigen.

Assoziationsklasse und Navigierbarkeit

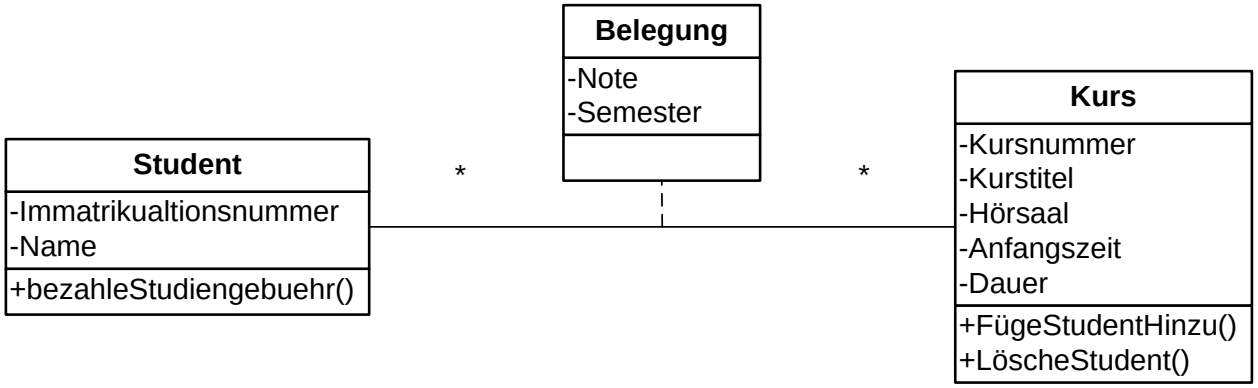

Abb. 14-9: Assoziationsklasse

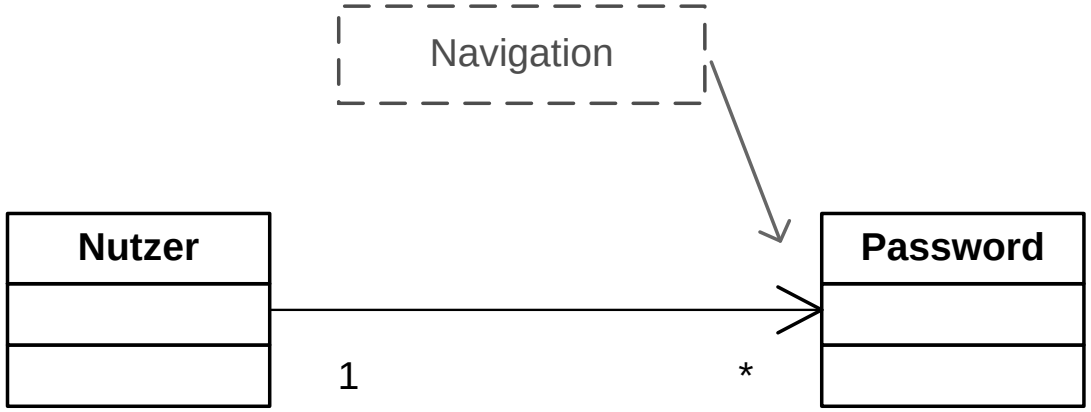

Abb. 14-10: Navigierbarkeit

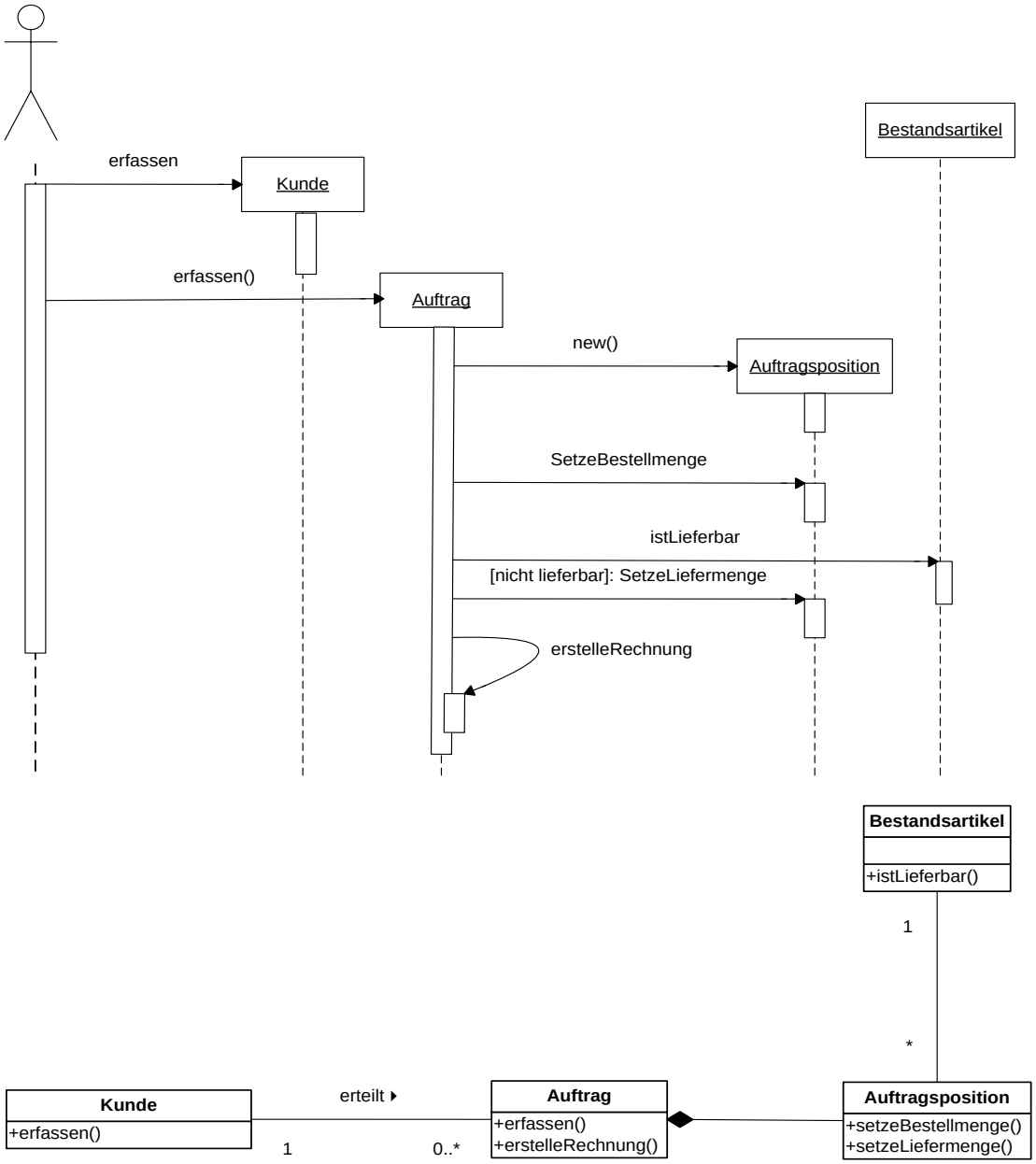

Abb. 14-11: Zuordnung zwischen einem Sequenz-diagramm und einem Teil des Klassendiagramms am Beispiel der Auftrags-bearbeitung (Neukunde)

Werkzeuge zur Unterstützung des manuellen Codierens

- *Editoren* unterstützen die „Textverarbeitung“ des Programmcodes.
- *Debugger* erlauben es, manuell erstellten Code interaktiv zu testen.
- *Bibliotheken* sind ein Hilfsmittel zur Wiederverwendung von Code.
- *Data Dictionaries* dienen der modulübergreifenden Dokumentation von Datenstrukturen.

Definitionen

Testen

bezeichnet jede einzelne (im Allgemeinen stichprobenartige) Ausführung des Testobjekts, d. h. einer Komponente, eines integrierten Teilsystems oder eines Systems (in einer Version), um die (beobachteten) Ergebnisse im Hinblick auf gewisse Eigenschaften zu überprüfen.

Fehler

Ein Fehler ist die Nichterfüllung einer festgelegten Anforderung, eine Abweichung zwischen dem Ist-Verhalten (während der Ausführung der Tests oder des Betriebs festgestellt) und dem Soll-Verhalten (in der Spezifikation oder den Anforderungen festgelegt).

Grundlegende Teststrategie

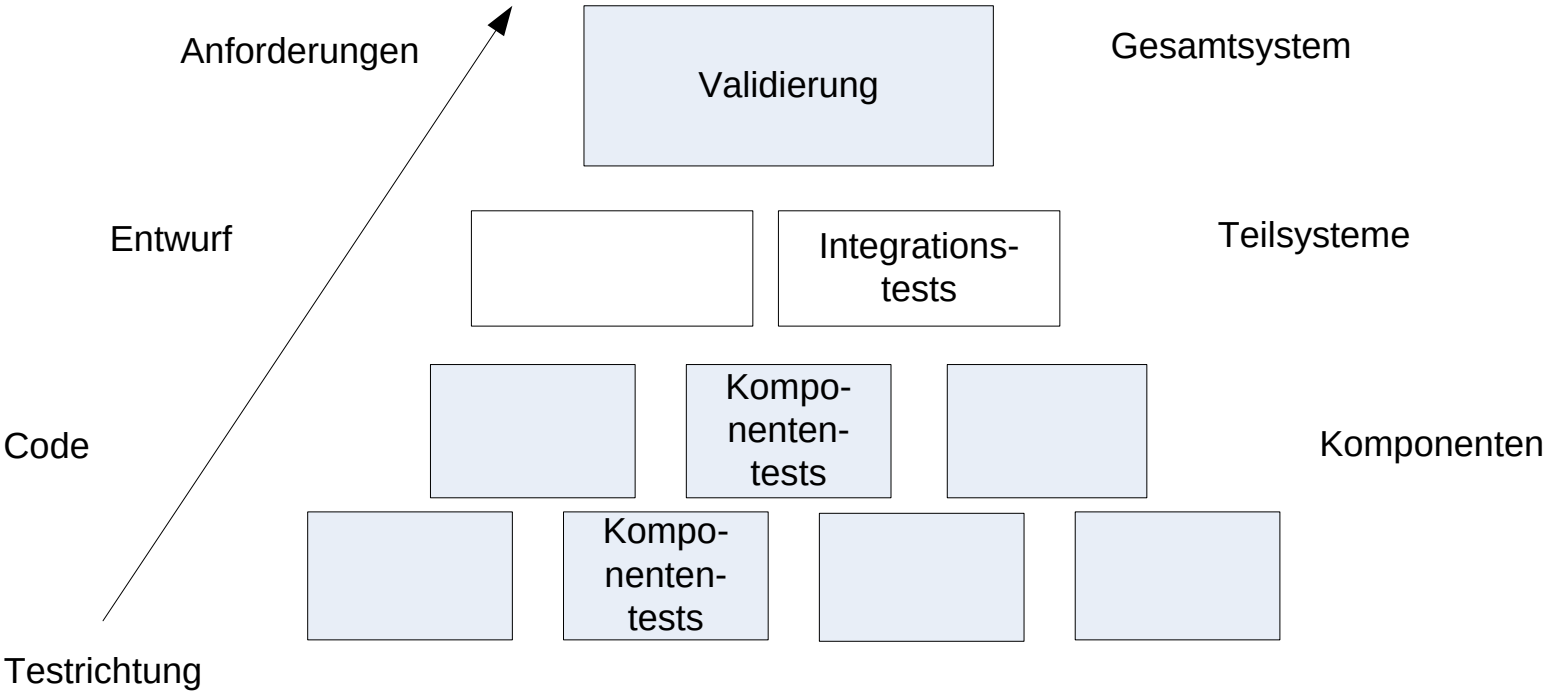

Abb. 14-14: Grundlegende Teststrategie

Testmethoden

- *Black-box-Tests*: Beobachtung des Testobjekts beim Testen nur von außen, wobei das Innere des Testobjekts verborgen bleibt (Anwendung häufig bei der Überprüfung von Anforderungen in den höheren Teststufen).
- *White-box-Tests*: Das Innere des Testobjekts wird berücksichtigt (Anwendung überwiegend bei den Komponententests).
- *Integrationstests*: Schwerpunkt des Testens liegt auf der Interaktion zwischen Komponenten.
- *Regressionstest:* Erneuter Test, um bereits getestete Programmteile nach deren Modifikation erneut auf Fehlerfreiheit zu prüfen

Grundlegender Testprozess

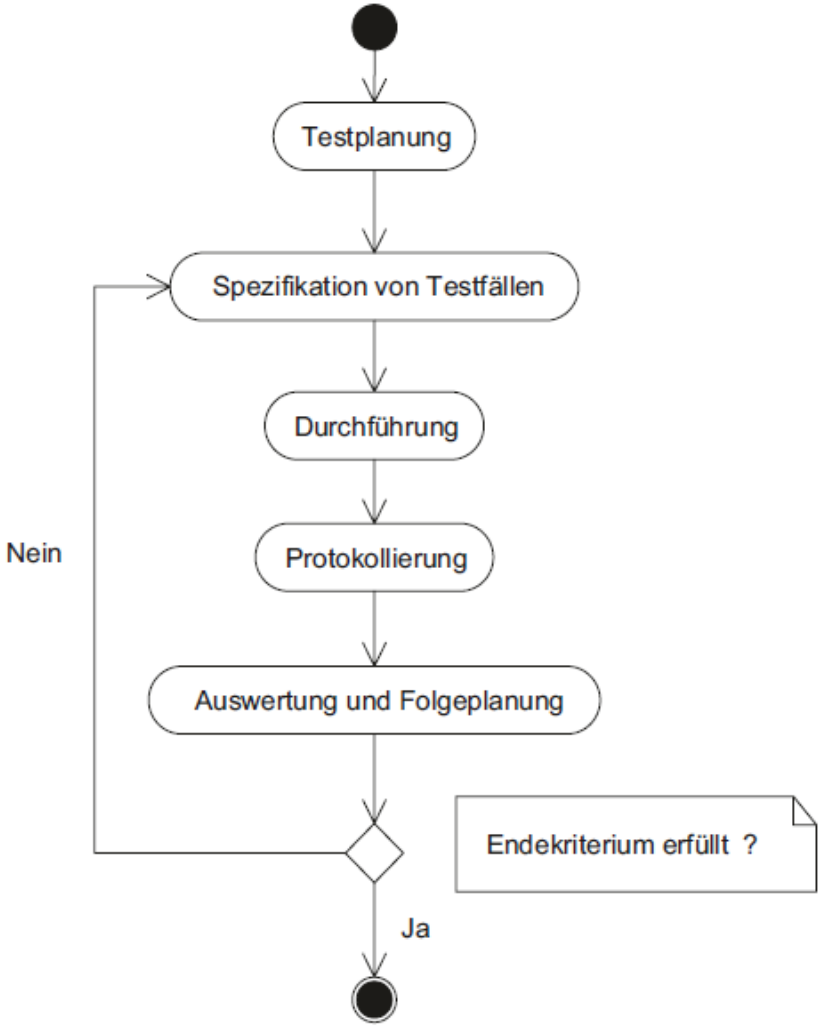

Abb. 14-15: Grundlegender Testprozess