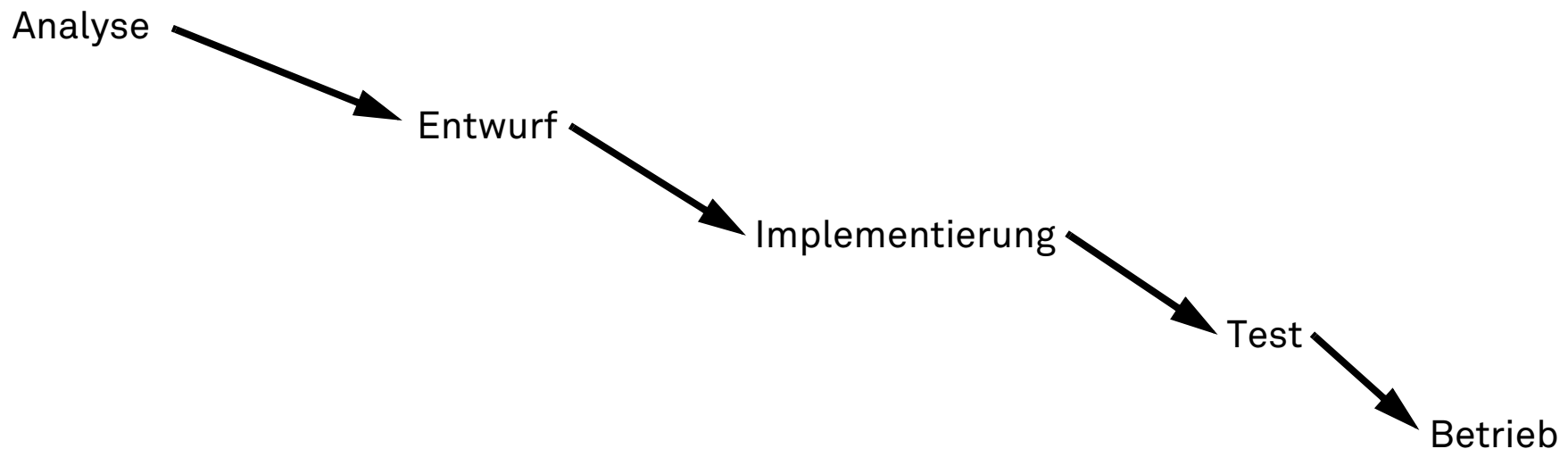


Folien zur Vorlesung **Softwaretechnik**

Teil 5: Vorgehensmodelle **Abschnitt 5.1: Motivation**

Organisation: Prozessmodelle/Vorgehensmodelle (Wiederholung/Folie 342)

intuitiver Ansatz: **Wasserfall-Modell** (Royce 1970)



Beschreibung:

- Alle Tätigkeiten einer Phase werden abgeschlossen, bevor die nächste Phase beginnt.
- Das Softwareprodukt wird in seiner Gesamtheit vollständig weiterentwickelt.
- Es gibt also keine Notwendigkeit für einen Rücksprung in eine frühere Phase.

Organisation: Prozessmodelle/Vorgehensmodelle (Wiederholung/Folie 343)

(Fortsetzung)

intuitiver Ansatz: **Wasserfall-Modell** (Royce 1970)

Vorteile:

- ❑ Die Abläufe sind einfach zu planen.
- ❑ Die Abläufe sind einfach zu überwachen.
- ❑ Das Vorgehen ist ausreichend für kleinere Projekte mit überschaubarer Dauer.

Nachteile:

- ❑ Das Vorgehen ist unflexibel bei geänderten oder neu auftretenden Anforderungen.
- ❑ Beim Erkennen von Fehlern ist eine Überarbeitung der Ergebnisse vorangehender Phasen nicht vorgesehen.
- ❑ Das Vorgehen ist daher für größere Projekte nicht anwendbar.

Organisation: Prozessmodelle/Vorgehensmodelle (Wiederholung/Folie 345)

(Fortsetzung)

Verbesserungen des Wasserfall-Modells:

- ❑ Die Rückkehr in frühere Phasen wird erlaubt.
- ❑ Ein unterschiedlicher Entwicklungsfortschritt für Teile des Projekts wird vorgesehen.
- ❑ Eine geplante schrittweise Vervollständigung des Projekts wird vorgesehen.

⇒ **Alle Verbesserungen führen zu mehr Aufwand für die Projektplanung, Projektsteuerung und Projektüberwachung.**

Verbesserte Vorgehensmodelle

Inkrementelles Vorgehen:

- ❑ Das Softwareprodukt wird in mehreren, **vorab** geplanten Zyklen erstellt und erweitert.
- ❑ Nach einer ersten Anforderungsanalyse werden die Zyklen festgelegt.
- ❑ Jeder Zyklus umfasst Analyse, Entwurf, Implementierung und den Test für einen **vorab** festgelegten Teil des Produktes.

Vorteile:

- ❑ Spätere Zyklen können Ergebnisse und Erfahrungen früherer Zyklen berücksichtigen.
- ❑ Große Produkte können in überschaubare Teile zerlegt werden.

Nachteile:

- ❑ Das Vorgehen besitzt nur eingeschränkte Flexibilität, da die bereits entwickelten Inkremente erhalten bleiben sollen.
- ❑ Die frühe Planung geeigneter Zyklen ist schwierig und fehleranfällig.

Verbesserte Vorgehensmodelle

(Fortsetzung)

Spiralmodell (Boehm, 1988):

- ❑ Ein inkrementelles Vorgehen, bei dem die Zyklen **nacheinander** geplant werden.
- ❑ Softwareprodukt durchläuft in jedem Zyklus vier Segmente:
 - Planung der aktuellen Erweiterung,
 - Festlegen der Ziele dieser Erweiterung (Reifegrad des Produkts),
 - Risikoanalyse (Simulationen, Voruntersuchungen),
 - Entwicklung und Test.

Vorteile:

- ❑ Mehr Flexibilität durch inhaltlich weitgehend abgeschlossene Zyklen.
- ❑ Die Risikoanalyse zeigt frühzeitig Probleme auf.

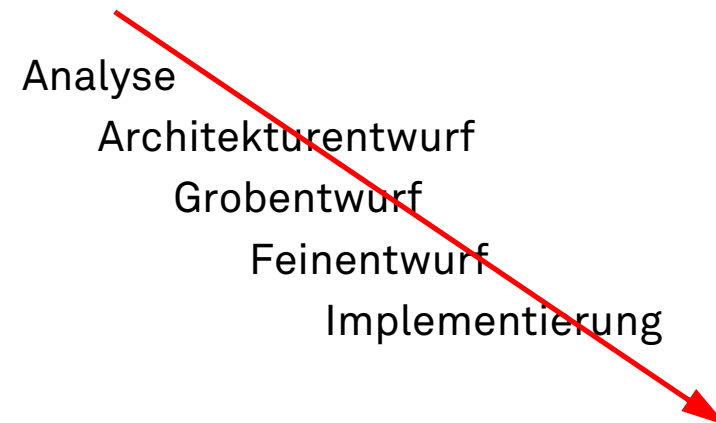
Nachteil:

- ❑ Der Aufwand vergrößert sich.

Organisation: Prozessmodelle/Vorgehensmodelle

(Fortsetzung)

V-Modell (ursprünglich auch von Boehm, 1981)

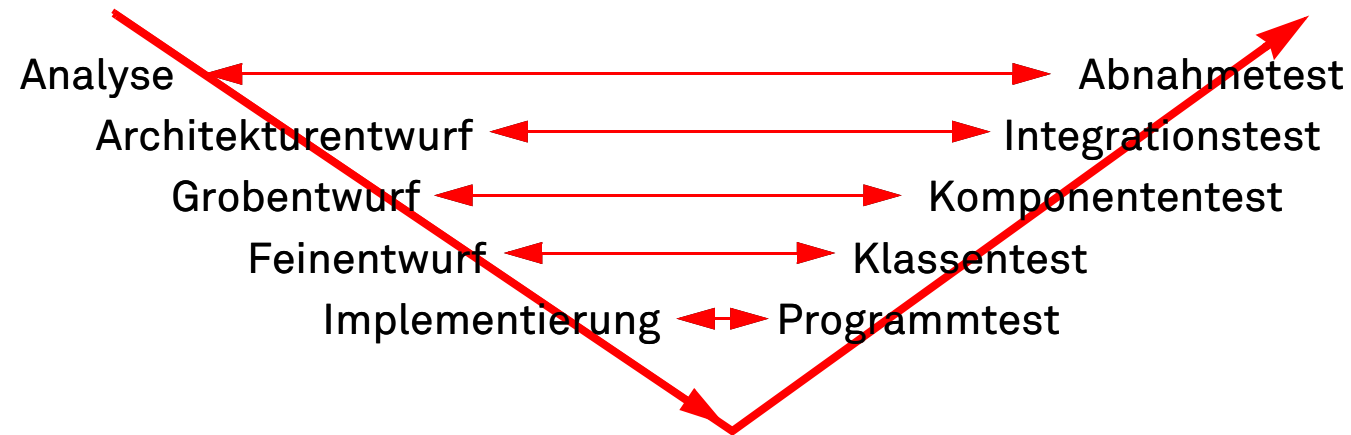


aus Wasserfall-Modell abgeleitet

Organisation: Prozessmodelle/Vorgehensmodelle

(Fortsetzung)

V-Modell (ursprünglich auch von Boehm, 1981)

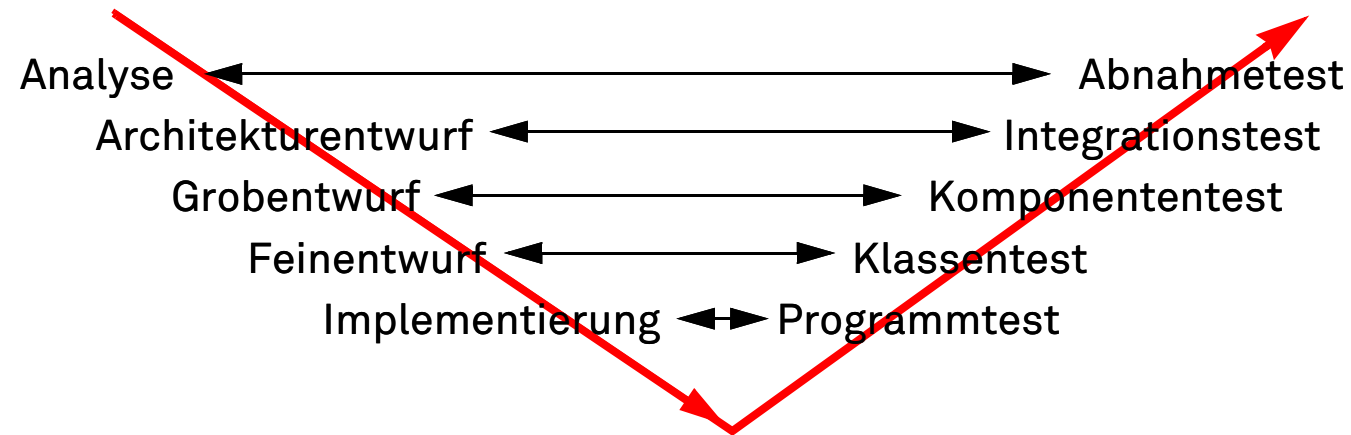


**stärkere Betonung der Qualitätssicherung
auf allen Ebenen der Entwicklung**

Organisation: Prozessmodelle/Vorgehensmodelle

(Fortsetzung)

V-Modell (ursprünglich auch von Boehm, 1981)



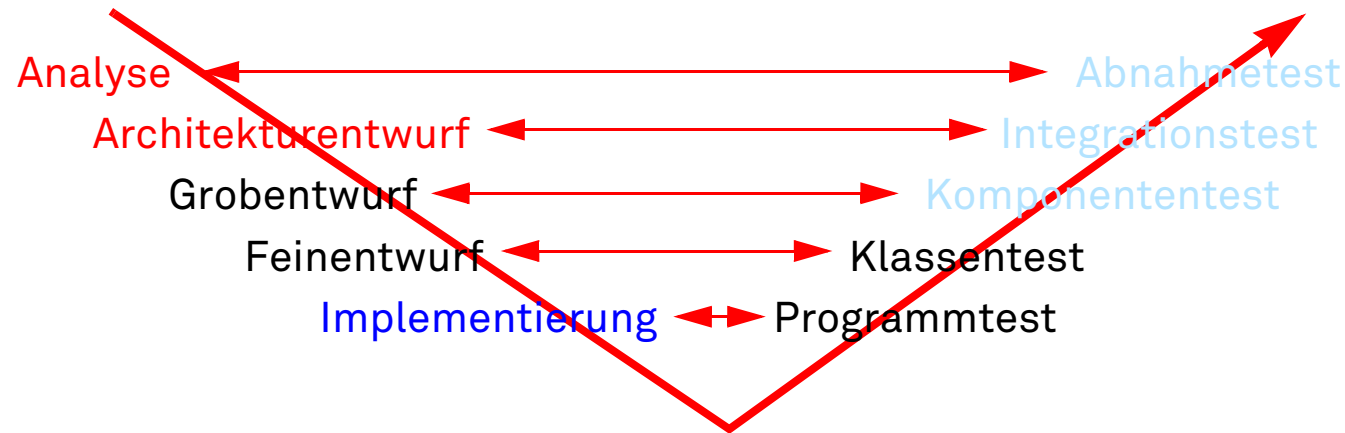
**Das Vorgehen ist heute als V-Modell XT
das fest vorgeschriebene Vorgehen für
Entwicklungsprojekte für den öffentlichen Bereich
in Deutschland.**

V-Modell XT (*eXtreme Tailoring*)

- ❑ V-Modell ist eine geschützte Marke der Bundesrepublik Deutschland.
- ❑ Das V-Modell wurde im Auftrag des Bundesministerium für Verteidigung seit 1986 entwickelt und die Verwendung ist dort vorgeschrieben seit 1991.
- ❑ Seit 1997 wird es auch im Bereich der zivilen Projekte der öffentlichen Verwaltung empfohlen: V-Modell '97
- ❑ Erweiterung zum seit 2005 veröffentlichten V-Modell XT (Version 2.0 seit 2015)
offizielle Web-Seite: <http://www.v-modell-xt.de>
- ❑ Das V-Modell XT fordert Vorgaben zu
Projektmanagement, Qualitätssicherung, Konfigurationsmanagement,
Problem- und Änderungsmanagement.
- ❑ Das V-Modell XT beschreibt die Beiträge von
Auftraggeber und Auftragnehmer.
- ❑ Das V-Modell XT umfasst das Referenzmodell (verpflichtender Teil) und
optionale, organisations- oder projektspezifische Anteile.
- ❑ Das V-Modell XT ist ein produktorientiertes Vorgehensmodell:
Das Ergebnis eines Entwicklungsschritts (Produkt) wird detailliert vorgegeben,
nicht der Vorgang der Erstellung des Produkts

Inhalte der Vorlesung Softwaretechnik

V-Modell



In dieser Vorlesung werden zur Dokumentation der Ergebnisse der verschiedenen Phasen folgende Notationen vorgestellt und eingesetzt:

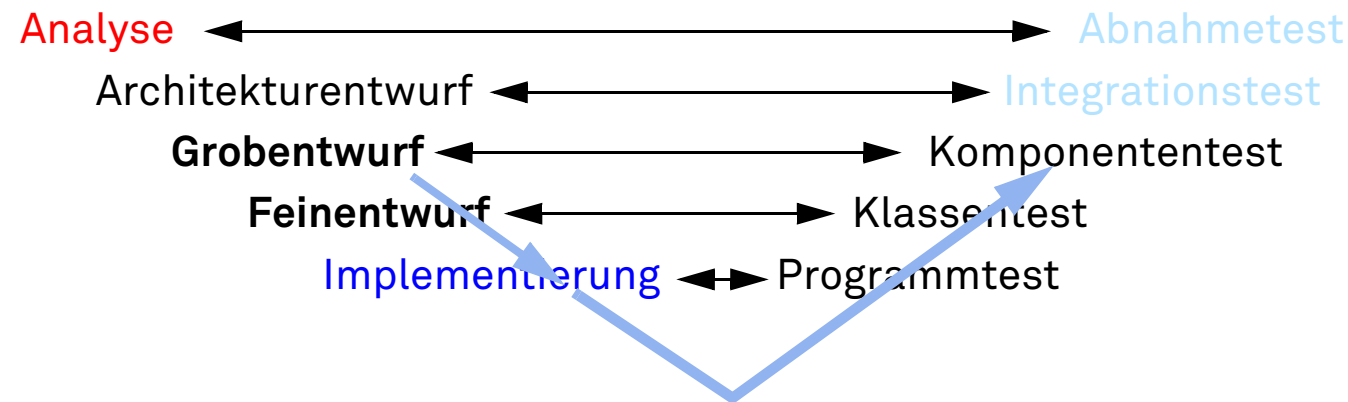
- ❑ Analyse, Entwurf: UML (Unified Modeling Language)
- ❑ Implementierung: Java, Visualisierung durch UML
- ❑ Test: Java (JUnit-Bibliothek), Unterstützung durch UML

Folien zur Vorlesung **Softwaretechnik**

Abschnitt 5.2: Analyse

Rückblick

V-Modell



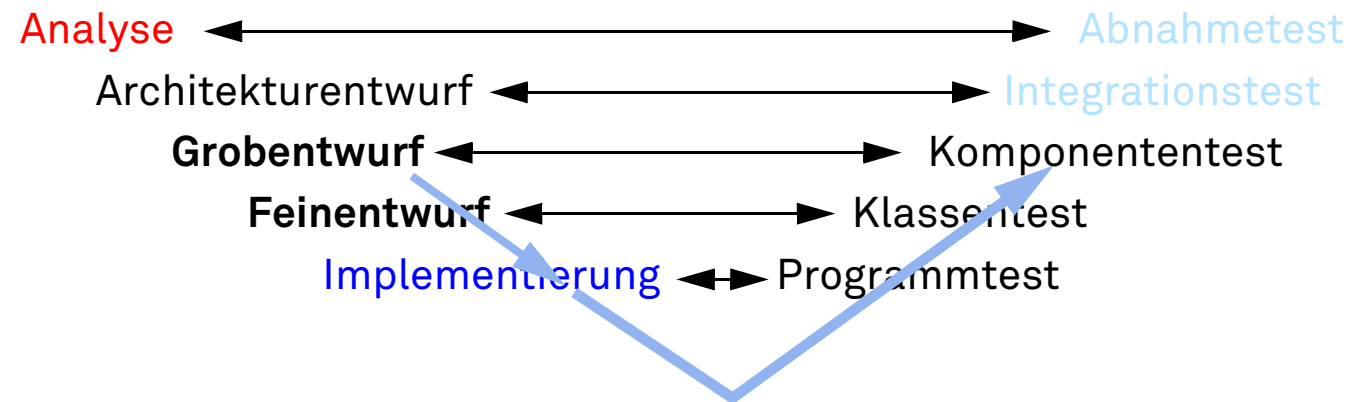
bisher im Rahmen des Entwurfs kennengelernt:

- ❑ Klassendiagramme zur Spezifikation/Dokumentation der Datenorganisation,
- ❑ Objektdiagramme zur Veranschaulichung von Objektstrukturen,
- ❑ Aktivitätsdiagramme zur Beschreibung von Algorithmen:
 - einzelne Methoden
 - grobes Zusammenwirken von Methoden
- ❑ Sequenzdiagramme zur Veranschaulichung von Abläufen mit mehreren Objekten

Rückblick

(Fortsetzung)

V-Modell



Mit den bisher kennengelernten Methoden, Konzepten und Notationen lässt sich Software

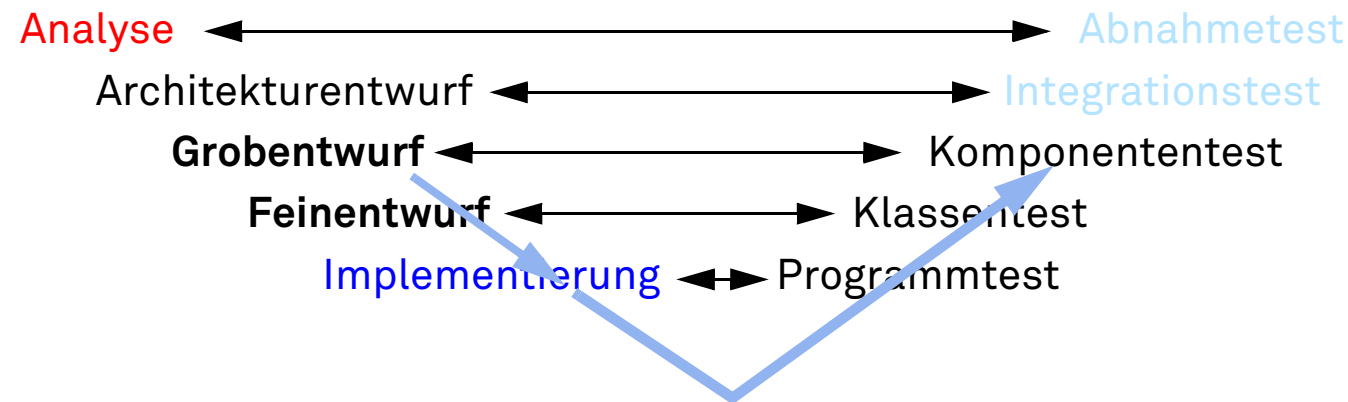
technisch

gestalten, veranschaulichen und prüfen.

Rückblick

(Fortsetzung)

V-Modell



Mit den bisher kennengelernten Methoden, Konzepten und Notationen lässt sich Software

technisch

gestalten, veranschaulichen und prüfen.

offene Fragestellung: **Was** soll die Software tun, die technisch realisiert wird?

Aufgaben der (Problem-)Analyse

Anforderungen an das zu entwickelnde System

- ☐ ermitteln,
- ☐ verstehen,
- ☐ klären,
- ☐ spezifizieren,
- ☐ dokumentieren.

Anforderung

Eine Anforderung ist eine Eigenschaft,
die ein System oder eine Person benötigt,
um ein Problem zu lösen oder ein Ziel zu erreichen

Literatur: Partsch, Helmuth A.: Requirements-Engineering systematisch – Modellbildung für softwaregestützte Systeme, S. 19-68
http://link.springer.com/chapter/10.1007/978-3-642-05358-0_2

Arten von Anforderungen

- ❑ Eine **funktionale Anforderung**
definiert eine vom System/einer Systemkomponente bereitzustellende Funktion.
Beispiele:
"Zugriff zu den Daten soll erst nach Eingabe eines Schlüsselworts möglich sein."
"Die Note soll als gewichteter Durchschnitt berechnet werden."

- ❑ Eine **Qualitätsanforderung**
definiert eine qualitative Eigenschaft eines Systems/einer Systemkomponente oder einer funktionalen Anforderung.
Beispiele:
Vorgaben für Zeitverhalten, Genauigkeit, Speicherbedarf

- ❑ Eine **Rahmenbedingung**
ist eine von außen kommende funktionale Anforderung oder Qualitätsanforderung, die im Rahmen der Entwicklung nicht geändert oder beeinflusst werden kann.
Beispiele:
gesetzliche Regelungen, ethische Aspekte, physikalische Gesetze

Arten von Anforderungen

(Fortsetzung)

Eine Anforderung beschreibt in der Regel das **Ziel**
und nicht seine technische **Umsetzung**.

Arten von Anforderungen

(Fortsetzung)

Eine Anforderung beschreibt in der Regel das **Ziel**
und nicht seine technische **Umsetzung**.

Allerdings sind **Ziel** und **Umsetzung** abhängig vom Betrachter!

- in der Analyse:
Das System soll gut abgesichert sein. ist **Zielvorgabe** des Auftraggebers.

Arten von Anforderungen

(Fortsetzung)

Eine Anforderung beschreibt in der Regel das **Ziel**
und nicht seine technische **Umsetzung**.

Allerdings sind **Ziel** und **Umsetzung** abhängig vom Betrachter!

- ❑ in der Analyse:
 - Das System soll gut abgesichert sein.* ist **Zielvorgabe** des Auftraggebers.
 - Es soll ein 256-bit-Schlüssel verwendet werden.* ist **Umsetzung** des Analysten.
- ❑ im Entwurf:
 - Es soll ein 256-bit-Schlüssel verwendet werden.* ist **Zielvorgabe** des Analysten.
 - Es soll der Rijndael-Algorithmus verwendet werden.* ist **Umsetzung** des Designers.

Arten von Anforderungen

(Fortsetzung)

Eine Anforderung beschreibt in der Regel das **Ziel**
und nicht seine technische **Umsetzung**.

Allerdings sind **Ziel** und **Umsetzung** abhängig vom Betrachter!

- ❑ in der Analyse:
 - Das System soll gut abgesichert sein.* ist **Zielvorgabe** des Auftraggebers.
 - Es soll ein 256-bit-Schlüssel verwendet werden.* ist **Umsetzung** des Analysten.
- ❑ im Entwurf:
 - Es soll ein 256-bit-Schlüssel verwendet werden.* ist **Zielvorgabe** des Analysten.
 - Es soll der Rijndael-Algorithmus verwendet werden.* ist **Umsetzung** des Designers.
- ❑ in der Implementierung:
 - Es soll der Rijndael-Algorithmus verwendet werden.* ist **Zielvorgabe** des Designers.
 - passender Programmcode in Java* ist **Umsetzung** des Programmierers.

Arten von Anforderungen

(Fortsetzung)

Eine Anforderung beschreibt in der Regel das **Ziel**
und nicht seine technische **Umsetzung**.

Allerdings sind **Ziel** und **Umsetzung** abhängig vom Betrachter!

- ❑ in der Analyse:
 - Das System soll gut abgesichert sein.* ist **Zielvorgabe** des Auftraggebers.
 - Es soll ein 256-bit-Schlüssel verwendet werden.* ist **Umsetzung** des Analysten.
- ❑ im Entwurf:
 - Es soll ein 256-bit-Schlüssel verwendet werden.* ist **Zielvorgabe** des Analysten.
 - Es soll der Rijndael-Algorithmus verwendet werden.* ist **Umsetzung** des Designers.
- ❑ in der Implementierung:
 - Es soll der Rijndael-Algorithmus verwendet werden.* ist **Zielvorgabe** des Designers.
 - passender Programmcode in Java* ist **Umsetzung** des Programmierers.

In der Analyse sollen die Anforderungen des Auftraggebers ermittelt werden,
die Zielvorgaben für den folgenden Entwurf sind.

Probleme in der Anforderungsanalyse

- ❑ Es sind viele Personen beteiligt, die direkt oder indirekt Einfluss auf Anforderungen nehmen:
 - spätere Benutzer
 - spätere Betreiber (technische Administration)
 - Experten des Anwendungsbereichs (Domänenexperten)
 - Management des Auftraggebers
 - Mitarbeitervertreter/Betriebsrat
 - eventuell Vertriebspersonal
 - Entwickler
 - ...
- ❑ Es wird angestrebt, **alle** Anforderungen zu erheben.
- ❑ Es wird angestrebt, **alle** Anforderungen **konsistent** zu beschreiben.
- ❑ Neben den eigentlichen Anforderungen müssen Kriterien für das Prüfen der Umsetzung von funktionalen und von qualitativen Anforderungen gefunden werden:
 - Es muss beschrieben werden, in welcher Weise eine Anforderung wie *einfach verständliche Bedienung* am fertigen Produkt geprüft werden kann.

Konsequenzen

Die Analyse ist ein **iterativer** Prozess aus:

- ❑ **Ermitteln** von Anforderungen bei einer Gruppe von Projektbeteiligten.
- ❑ **Verstehen** und **Klären** der ermittelten Anforderungen.
- ❑ **Spezifizieren** und **Dokumentieren** der ermittelten Anforderungen.
- ❑ **Validieren** und **Konsolidieren** der beschriebenen Anforderungen
 - mit der betroffenen Gruppe,
 - durch Vergleich mit anderen, bereits dokumentierten Anforderungen,
 - mit anderen beteiligten Gruppen.

Ermitteln von Anforderungen

- ❑ Verstehen der Anwendungsdomäne:
 - *Beispiel:* Das Ermitteln von Anforderungen für Software zur Kreditabwicklung erfordert Kenntnisse im Kreditgeschäft.
 - *Problem:* Wissenserwerb ist ein langwieriger Vorgang.
- ❑ Verstehen des konkreten Problems:
 - *Beispiel:* Die Abwicklung der Baufinanzierung bei der Bank X muss erlernt werden.
 - *Problem:* Für die Abwicklung bei X müssen Informationsquellen bestimmt werden.
- ❑ Verstehen des Geschäftsumfelds:
 - *Beispiel:* Der Beitrag der Software zum gesamtunternehmerischen Erfolg muss verstanden werden.
 - *Problem:* Hier hat z.B. das Management eine andere Sicht als die späteren Benutzer.
- ❑ Verstehen der beteiligten Geschäftsprozesse:
 - *Beispiel:* Software betrifft unterschiedliche Bereiche (Kreditwesen, Innenrevision).
 - *Problem:* Die veränderlichen Parameter dieser Bereiche, z.B. durch Gesetzesänderungen, müssen verstanden werden.

Ermitteln von Anforderungen

(Fortsetzung)

Informationsquellen zur Ermittlung von Anforderungen sind:

- ❑ Die Befragung von Personen,
die ein potentiell Interesse am zukünftigen System haben.
- ❑ Die Analyse von existierenden Dokumenten.
- ❑ Die Beobachtung von existierenden Abläufen.
- ❑ Die Analyse existierender Systeme:
 - Alt- oder Vorgängersysteme,
 - Konkurrenzprodukte,
 - ähnliche Systeme für andere Domänen.

Ermitteln von Anforderungen

(Fortsetzung)

Die durchzuführenden Tätigkeiten sind:

- ❑ Anforderungen erfassen, beschreiben und verfeinern.
- ❑ Szenarien erfassen und beschreiben.
Ein *Szenario* beschreibt ein konkretes Beispiel für die Erfüllung oder auch Nicht-Erfüllung von einer oder mehreren Anforderungen und konkretisiert diese dadurch.
- ❑ Lösungsorientierte Anforderungsaspekte erfassen und beschreiben:
 - Datenperspektive (strukturelle Beziehungen von Daten)
 - Funktionsperspektive (Manipulation von Daten durch Funktionen)
 - Verhaltensperspektive (Reaktion auf externe Signale, Zustandsänderungen)

Verstehen und Klären der ermittelten Anforderungen

- ❑ Prüfen der *Notwendigkeit*:
 - Leistet die Anforderung einen Beitrag zur Problemlösung?
- ❑ Prüfen der *Konsistenz*:
 - Stehen Anforderungen zueinander im Konflikt oder behindern sich?
 - Unterstützen sich Anforderungen, sind sie möglicherweise äquivalent?
- ❑ Prüfen der *logischen Vollständigkeit*:
 - Werden alle erfassten Daten verarbeitet?
 - Werden die zur Ausführung von Funktionen benötigten Daten bereitgestellt?
- ❑ Prüfen der *Machbarkeit*:
 - Sind die Anforderungen unter Zeit-, Personal-, Budgetrestriktionen umsetzbar?
 - Kann die Anforderungserfüllung geprüft werden?
- ❑ *Priorisierung*:
 - Die Bedeutung der Anforderungserfüllung für das Gesamtvorhaben wird festgelegt.

Verstehen und Klären der ermittelten Anforderungen

(Fortsetzung)

Die Analyse von Anforderungen ist zeitaufwändig und teuer:

- ❑ Experten müssen sich das durch die Anforderungen beschriebene System *vorstellen* und seine Ausführung – zumindest mental – *simulieren*.
- ❑ Die Konsequenzen des Zusammenwirkens aller Anforderungen müssen *durchdacht* werden.

Ein allgemeines systematisches Vorgehen für die Analyse lässt sich nicht angeben.

Validieren und Konsolidieren der ermittelten beschriebenen Anforderungen

- ❑ Überprüfen der gesammelten Anforderungen
 - mit Auftraggeber,
 - mit Betroffenen,
 - mit externen Experten.

- ❑ Prüfen von
 - Richtigkeit,
 - Verständlichkeit,
 - Nachvollziehbarkeit,
 - Verifizierbarkeit.

- ❑ Techniken:
 - Lesen, Vorstellen, Nachvollziehen
 - Prototyp-Erstellung,
 - Definition von Testfällen.

Spezifizieren und Dokumentieren von Anforderungen

Die Dokumentation kann erfolgen durch:

- ❑ natürlich-sprachliche Texte,
- ❑ standardisierte Formulare,
- ❑ graphische Notationen oder
- ❑ formale Spezifikationen.

Spezifizieren und Dokumentieren von Anforderungen

(Fortsetzung)

Inhalte einer solchen Dokumentation sind:

- ❑ eindeutige Identifikation einer Anforderung,
- ❑ Festlegen der Bedeutung der Anforderung,
- ❑ Angabe des Kontextes, in dem die Anforderung aufgetreten oder wichtig ist,
- ❑ ausführliche Beschreibung der Anforderung,
- ❑ Bezüge zu anderen Anforderungen.

Beispiel für Notation:
tabellarische Übersicht

Spezifizieren und Dokumentieren von Anforderungen

(Fortsetzung)

Tabellarische Dokumentation für **eine** Anforderung:

Abschnitt	Inhalt
Identifikation	Bezeichnung, Autoren, Version, Erstellungsdatum, Änderungshistorie
Kritikalität	Wichtigkeit der Anforderung, Bedeutung für den Erfolg
Kontext	Bezeichnung der Quellen, die die Anforderung genannt haben, Profiteure der Realisierung der Anforderung; Angabe der Detaillierungsebene der Anforderung
Beschreibung	ausführliche Beschreibung des Anforderung, Szenarien benennen, die die Anforderung erläutern
Bezüge zu anderen Anforderungen	übergeordnete Ziele: Angabe allgemeinerer Anforderungen, untergeordnete Ziele: detailliertere Anforderungen Konflikte: Konkurrenzbeziehungen zu anderen Anforderungen

Hinweis: Es sollen Ziele und nicht Realisierungen beschrieben werden.

Spezifizieren und Dokumentieren von Anforderungen

(Fortsetzung)

Dokumentation von Szenarien

- ❑ UML-Anwendungsfalldiagramm (*Use-Case-Diagramm*)
- ❑ tabellarische Beschreibungen
- ❑ UML-Sequenzdiagramm (*bereits bekannt*)
- ❑ UML-Aktivitätsdiagramm (*bereits bekannt*)



Folien zur Vorlesung **Softwaretechnik**

Abschnitt 5.3: Anwendungsfalldiagramme

Spezifizieren und Dokumentieren von Anforderungen

(Fortsetzung)

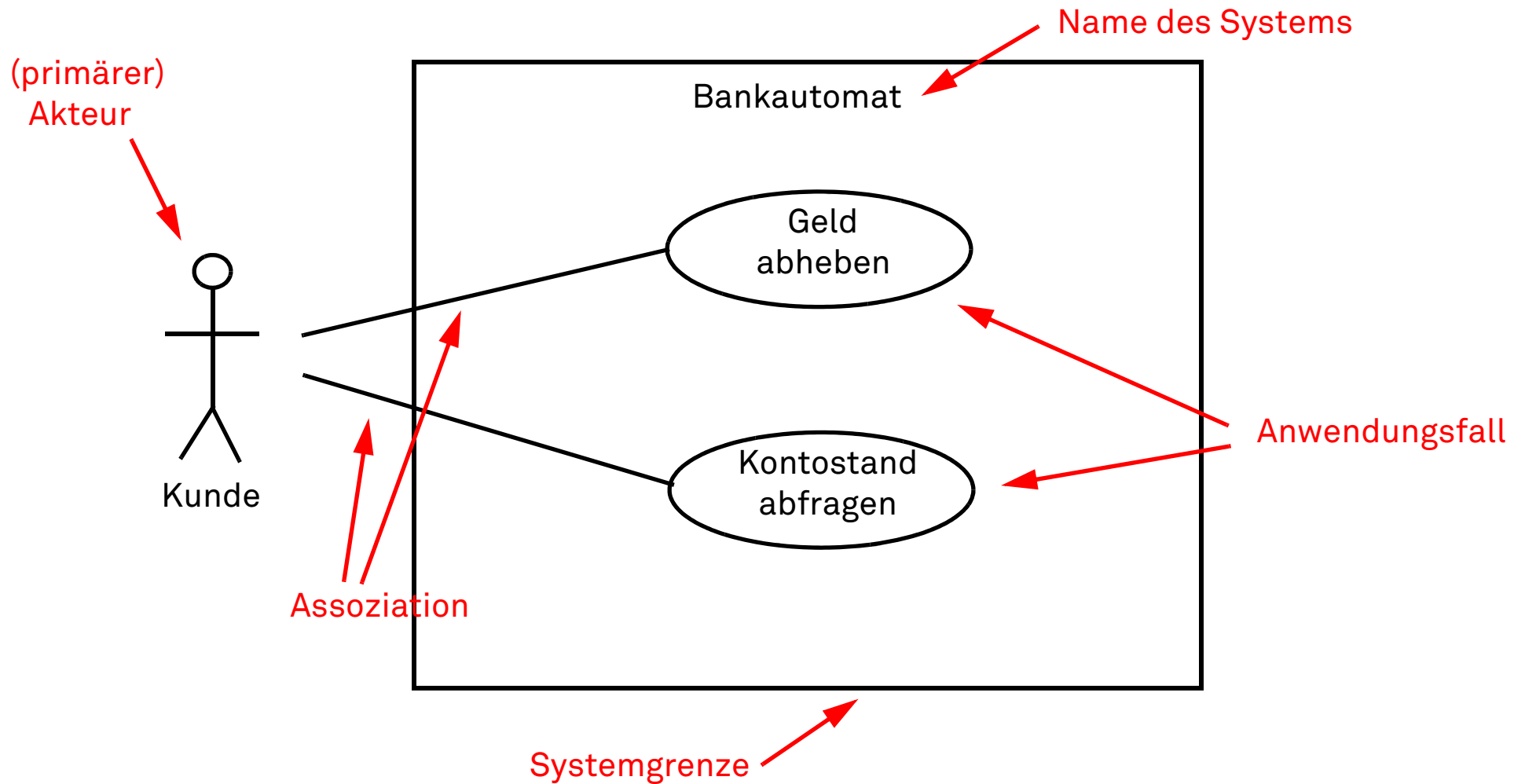
Anwendungsfalldiagramm

Die wesentlichen Modellierungselemente sind:

- ❑ **Akteur**
ist ein externes System,
das mit dem zu entwickelnden System interagiert:
Person, andere Hardware oder andere Software
- ❑ **Anwendungsfall** (engl. Use Case)
ist ein aus Sicht eines Akteurs zusammenhängendes Verhalten,
welches das zu entwickelnde System nach außen sichtbar anbieten soll.

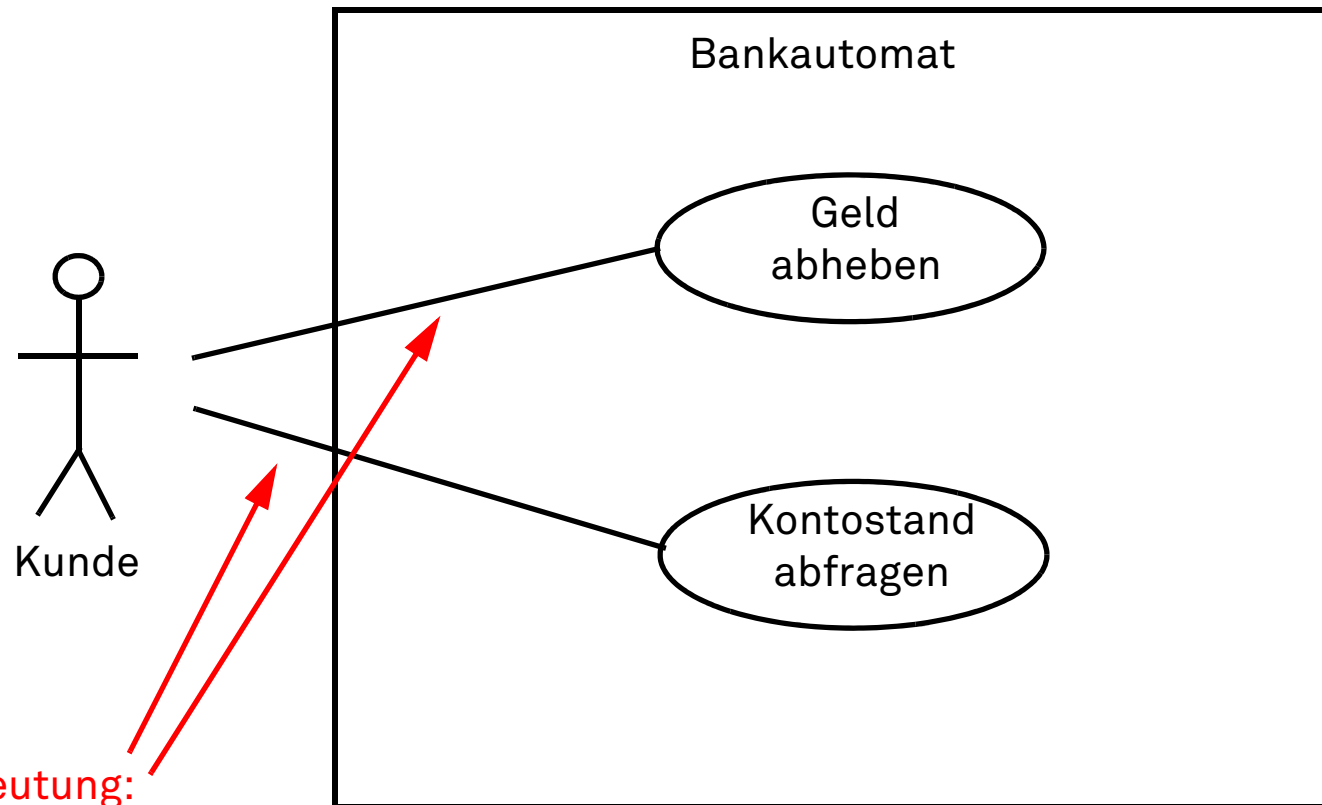
Literatur: Seemann, Jochen; von Gudenberg, Jürgen: Software-Entwurf mit UML 2 – Objektorientierte Modellierung mit Beispielen in Java, S. 15-25
<http://www.springerlink.com/content/jm3124/#section=390797&page=1&locus=0>

Anwendungsfalldiagramm (Beispiel)



Anwendungsfalldiagramm

(Fortsetzung)



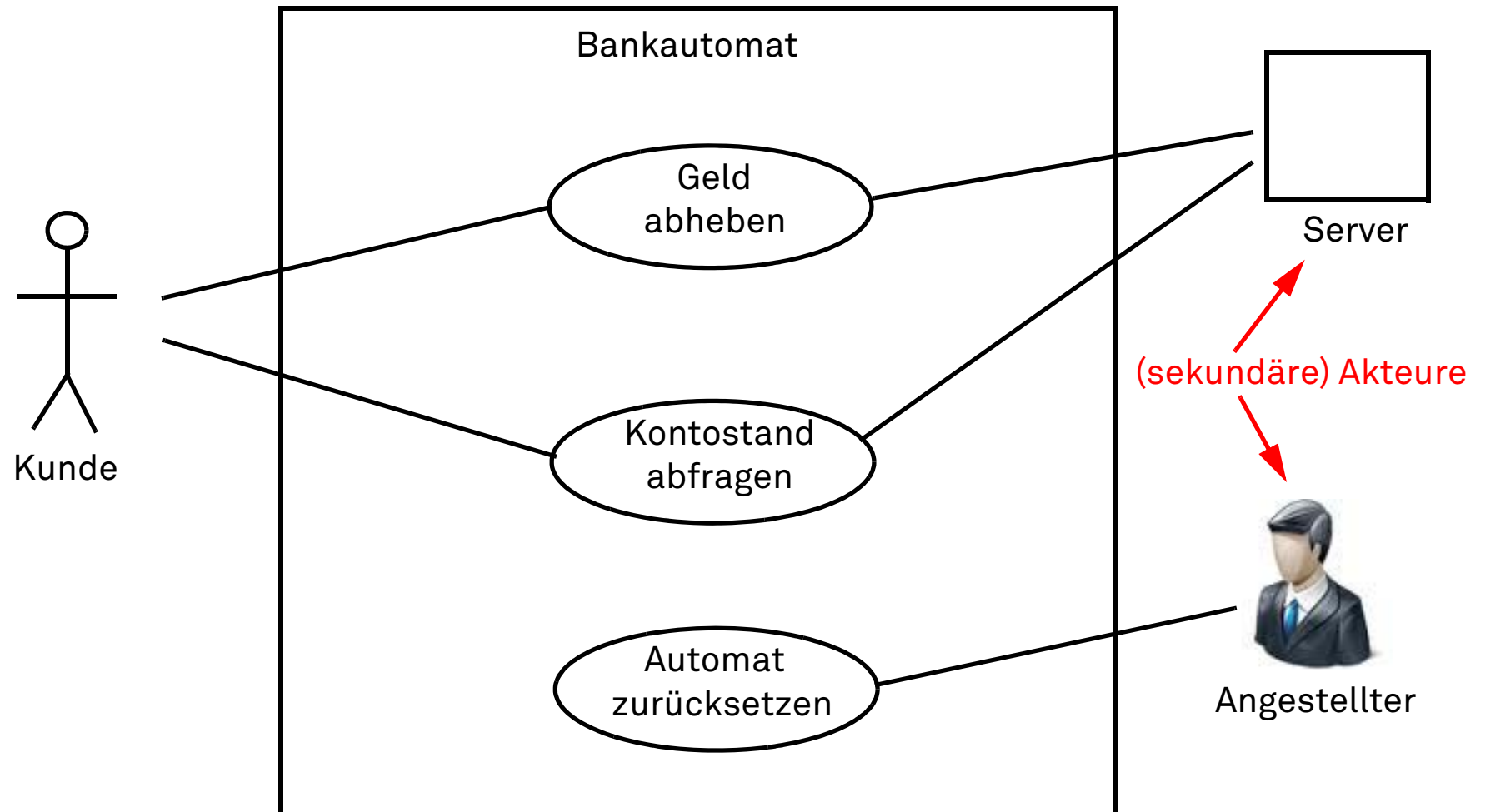
Bedeutung:

Kunde ist an den Anwendungsfällen *Geld abheben*
und *Kontostand abfragen* beteiligt.

Beide Anwendungsfälle können unabhängig voneinander ausgeführt werden.

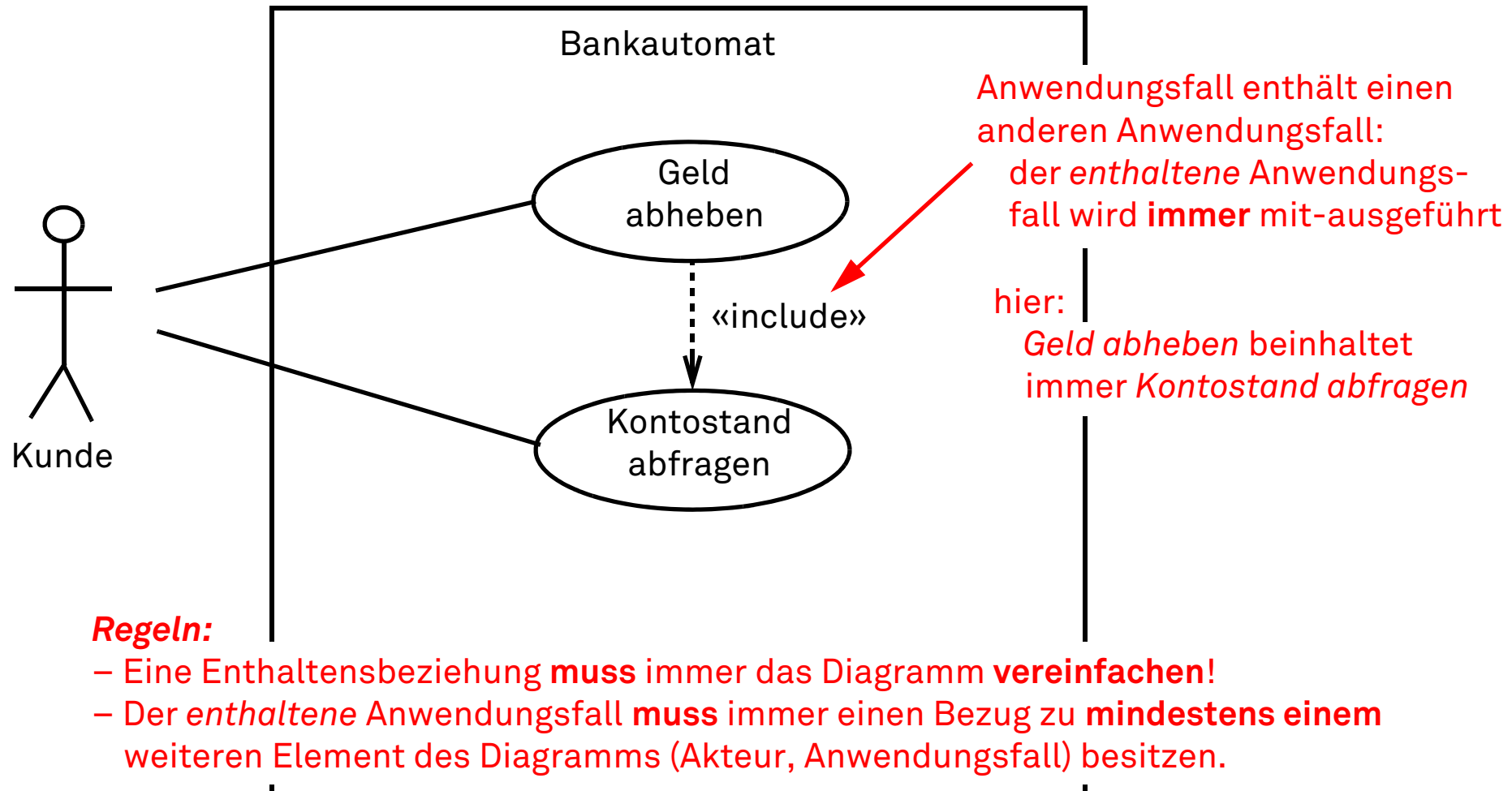
Anwendungsfalldiagramm

(Fortsetzung)



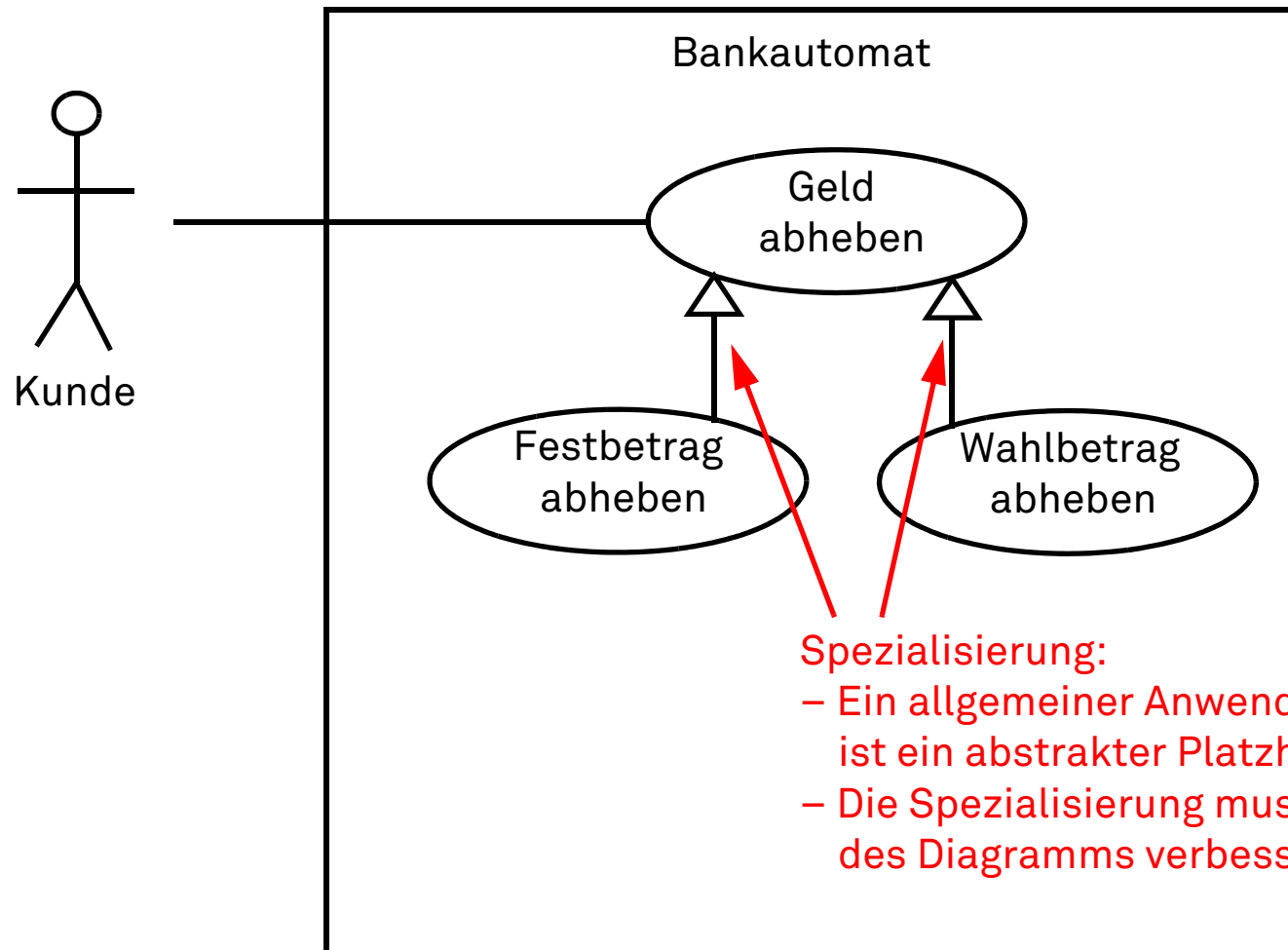
Anwendungsfalldiagramm

(Fortsetzung)



Anwendungsfalldiagramm

(Fortsetzung)

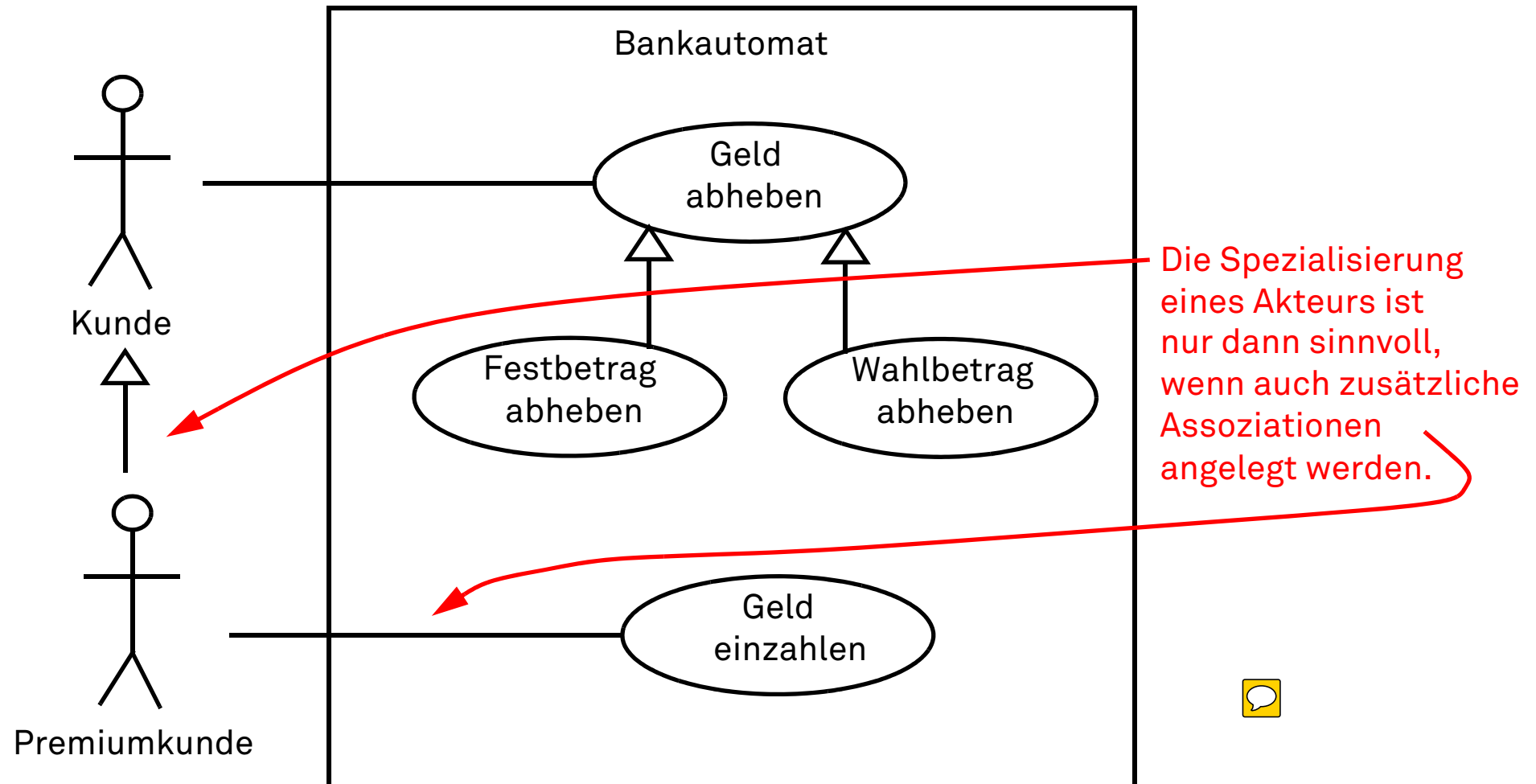


Spezialisierung:

- Ein allgemeiner Anwendungsfall ist ein abstrakter Platzhalter.
- Die Spezialisierung muss die Verständlichkeit des Diagramms verbessern.

Anwendungsfalldiagramm

(Fortsetzung)



Anwendungsfalldiagramm

(Fortsetzung)

zusätzlich **immer** notwendig: tabellarische Beschreibung für **jeden** Anwendungsfall

Abschnitt	Inhalt
Identifikation	Bezeichner des Anwendungsfalls, Autoren, Version, Erstellungsdatum, Änderungshistorie
Kritikalität	Wichtigkeit, Bedeutung für den Erfolg des Systems beschreiben
Ursprung	Bezeichnung der Quellen, auf die der Anwendungsfall zurück geht
Beschreibung	kurze Beschreibung des Anwendungsfalls, Ziele, die durch den Anwendungsfall erfüllt werden sollen
Vorbedingung	Voraussetzungen für die Ausführung des Anwendungsfalls
Nachbedingung	Zustand nach Ausführung des Anwendungsfalls
Hauptszenario	Beispiel für normalen Ablauf (tabellarisch; auch möglich als Sequenz- oder Aktivitätsdiagramm)
Alternativszenarien	Darstellung von Variationen des normalen Ablaufs
Ausnahmeszenarien	Abläufe, bei denen die des Anwendungsfalls Ziele nicht erreicht werden
Qualität	Bezüge zu Qualitätsanforderungen

Anwendungsfalldiagramm

(Fortsetzung)

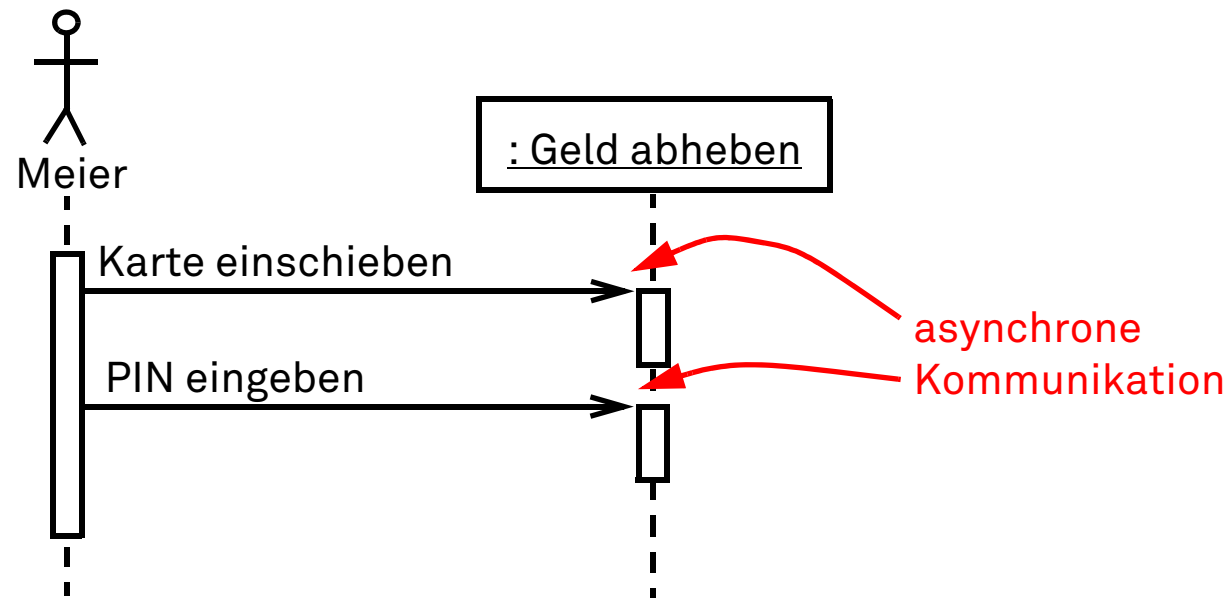
- ❑ Ein Anwendungsfalldiagramm zeigt
 - funktionale Anforderungen (= Anwendungsfälle),
 - Benutzer/externe Schnittstellen (= Akteure),
 - Interaktionsmöglichkeiten der Benutzer mit dem System (Assoziationen).
- ❑ Ein Anwendungsfalldiagramm zeigt **keine** Abläufe, sondern nur **strukturelle** Zusammenhänge.
- ❑ Ein Anwendungsfalldiagramm beschreibt die Konzeption aus Sicht der Akteure und **keine** technische (De-)Komposition

- ❑ Hinweis:
Daraus folgt, dass **nie** Assoziationen zwischen Anwendungsfällen auftreten.

Spezifizieren und Dokumentieren von Anforderungen: Sequenzdiagramm

Sequenzdiagramm
in der Analyse:

- ❑ Sequenzdiagramme ergänzen die Aussage von Anwendungsfalldiagrammen.
- ❑ Das Diagramm zeigt die Interaktion zwischen Akteuren und Anwendungsfällen.
- ❑ Die Kanten zeigen den *Nachrichtenfluss* (= die Weitergabe von Informationen).
- ❑ Die Nachrichten sind asynchron.
- ❑ Die Nachrichten werden informell beschrieben.



Folien zur Vorlesung **Softwaretechnik**

Abschnitt 5.4: Aktivitätsdiagramme (Ergänzung zu Teil 4.2)

Spezifizieren und Dokumentieren von Anforderungen: Aktivitätsdiagramm

Aktivitätsdiagramm
in der Analyse:

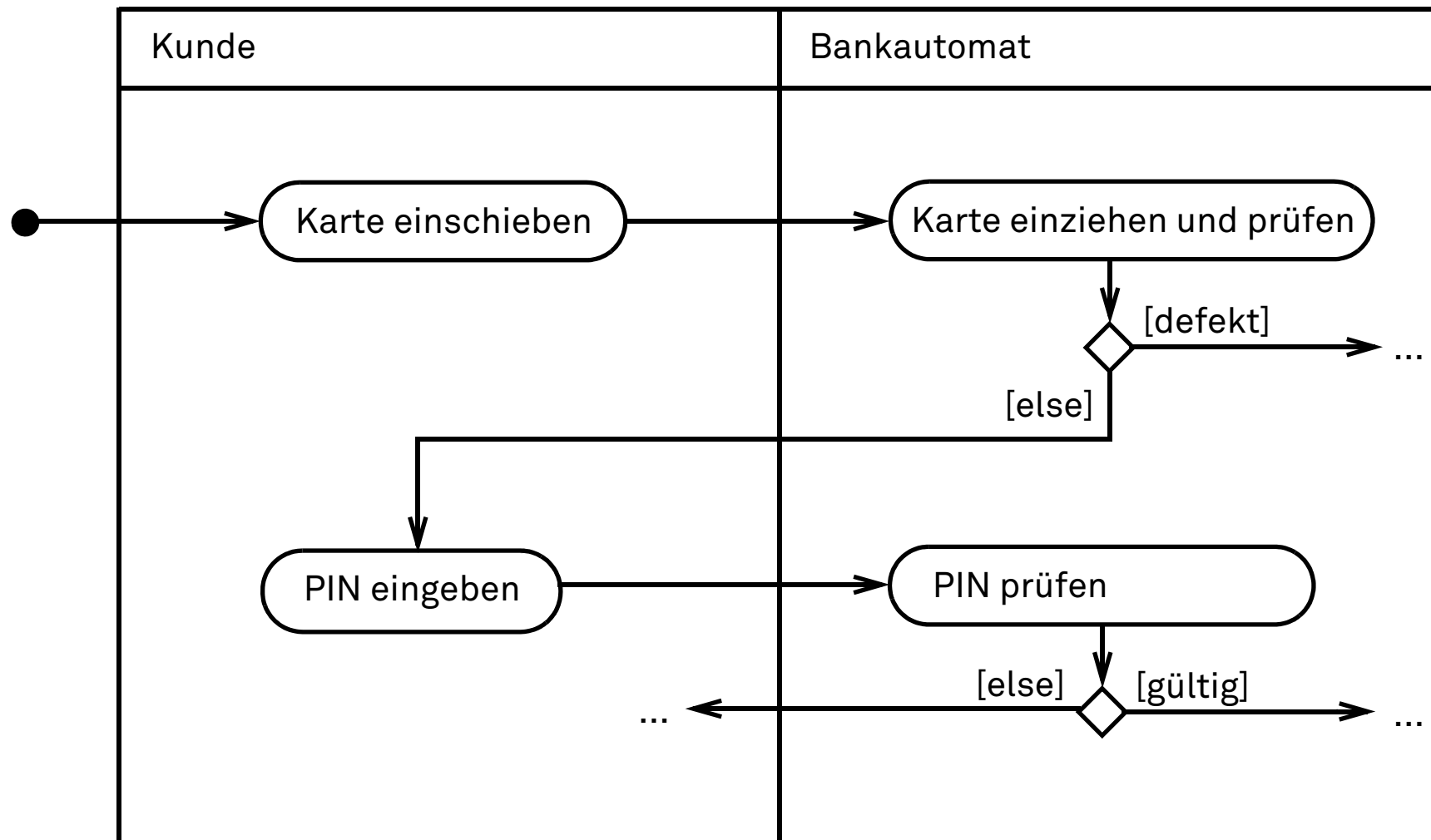
- ❑ Das Diagramm zeigt eventuell die Folge der Interaktionen zwischen Akteuren und Anwendungsfällen.
- ❑ Die Interaktionen werden informell beschrieben.
- ❑ Hilfsmittel zur Strukturierung von Aktivitätsdiagrammen:

Verantwortungsregionen

Region 1	Region 2	Region 3

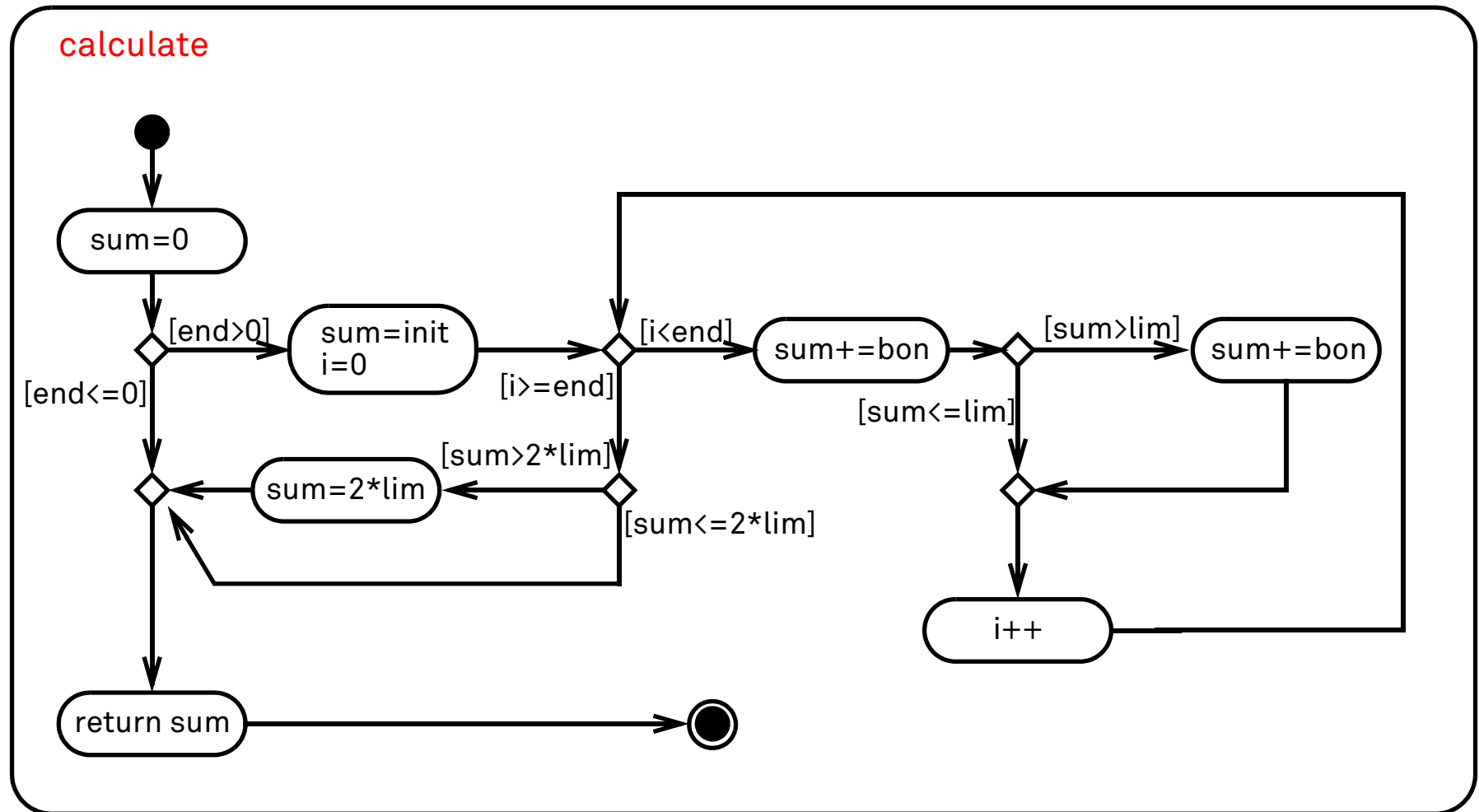
Dokumentieren von Anforderungen: Aktivitätsdiagramm

(Fortsetzung)



Aktivitätsdiagramm (Wiederholung Folie 360)

= Spezifikation eines Verhaltens als koordinierte Folge der Ausführung von Aktionen



Aktivitätsdiagramm – weitere Modellierungselemente

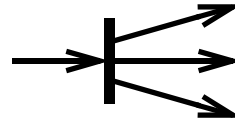
bisher:

- ❑ Aktivitätsdiagramme zur Visualisierung/Planung von Java-Programmen eingesetzt
- ❑ Folge: nur sequentielle Abläufe modelliert
- ❑ Im Aktivitätsdiagramm wird nur eine Marke erzeugt.

Ergänzen:

- ❑ Elemente zur Modellierung von nebenläufigen Prozessen durch Aktivitätsdiagramme:
 - Verteilungsknoten
 - Synchronisationsknoten
- ❑ Folge: Modellierung von Abläufen mit autonom handelnden Akteuren ist möglich.

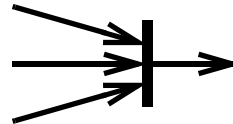
Verteilungsknoten



- ❑ Jede eingehende Marke wird so oft **vervielfacht**,
dass auf jeder Ausgangskante genau eine Marke weitergeleitet wird.
- ❑ Da mehrere Marken erzeugt werden,
entstehen mehrere nebenläufige Ausführungsstränge:

⇒ Mehrere Aktionen einer Aktivität können gleichzeitig aktiv sein.

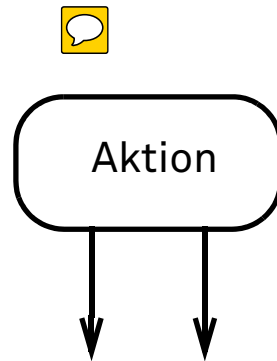
Synchronisationsknoten



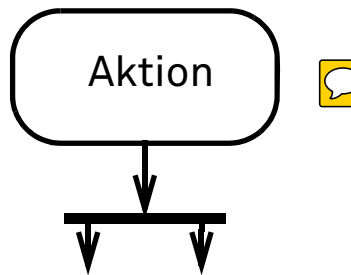
- ❑ Der Synchronisationsknoten schaltet nur genau dann, wenn auf **allen** eingehenden Kanten eine Marke eintrifft.
 - ❑ Auf **jeder** Eingangskante wird dabei genau eine Marke vernichtet, auf der Ausgangskante wird zeitgleich **genau eine einzige** Marke erzeugt.
 - ❑ Es werden so mehrere nebenläufige Ausführungen zusammengeführt und zeitlich synchronisiert.
-
- ❑ Die Kombination von Verteilungs- und Synchronisationsknoten muss Eigenschaften beider Typen besitzen.

implizite Verteilung und Synchronisation

- Aktionen und Verteilung:



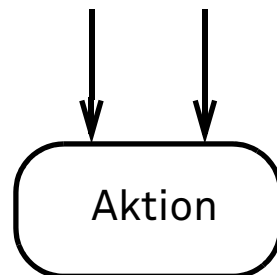
entspricht



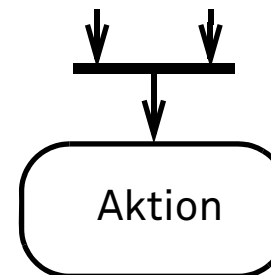
übersichtlichere Darstellungsform

- Aktionen und Synchronisation:

Vorsicht!



entspricht



Aktionen sollten also immer nur genau eine Eingangs- und genau eine Ausgangskante besitzen!

Beispiel – Nebenläufigkeit in Aktivitätsdiagrammen

Problemstellung:

Bei der Bestellung in einem Online-Shop soll der Kunde zunächst drei Tätigkeiten in **beliebiger** Reihenfolge durchführen können:

- ☐ das Angeben der Lieferanschrift,
- ☐ das Angeben der Kontodaten für die aktuelle Bestellung,
- ☐ das sequentielle Auswählen von Produkten.

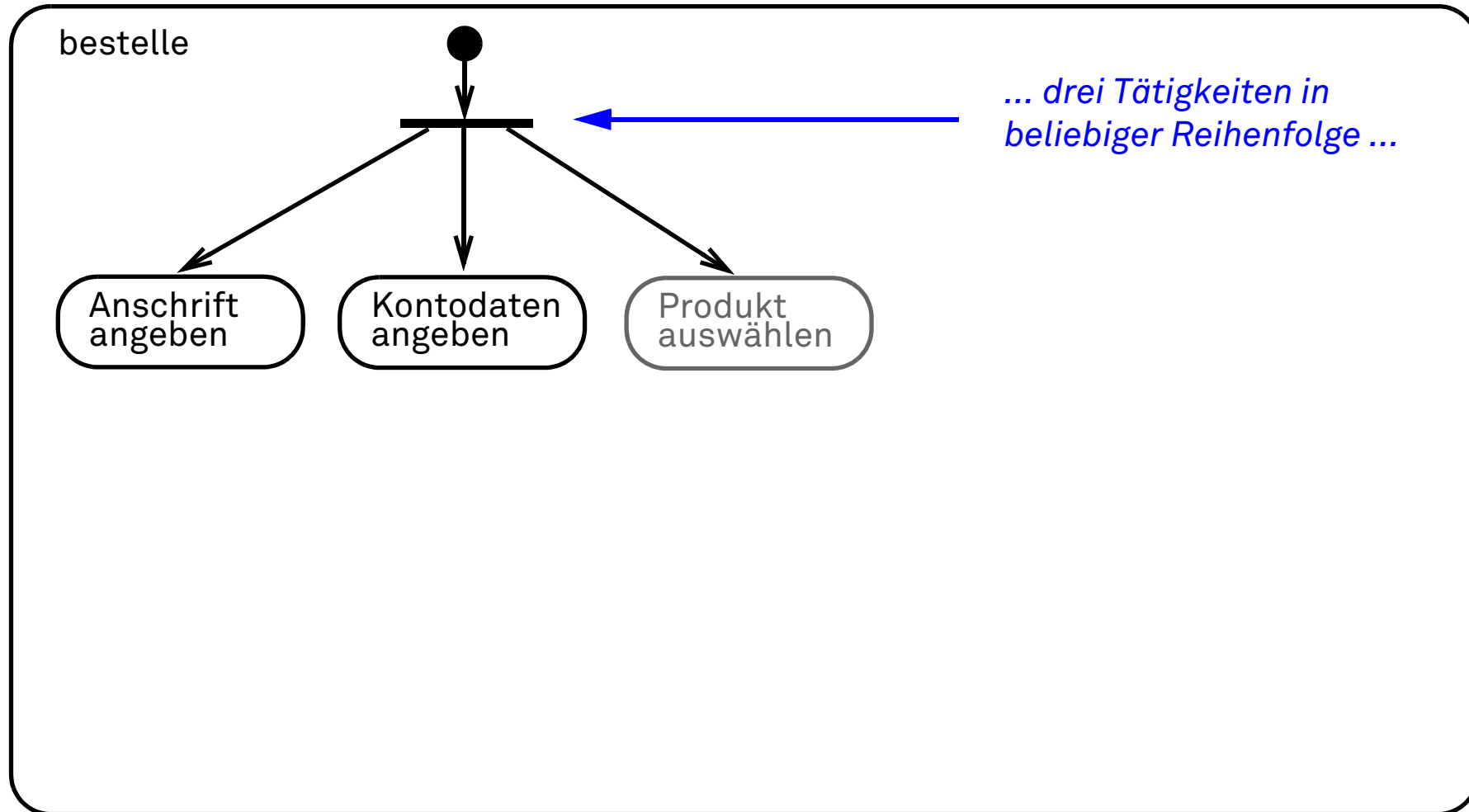
Nach Abschluss der Produktauswahl und der Angabe der Kontodaten soll dem Kunden eine Bestellübersicht angezeigt werden.

Aufgabe:

Dieser Vorgang soll als Aktivitätsdiagramm modelliert werden.

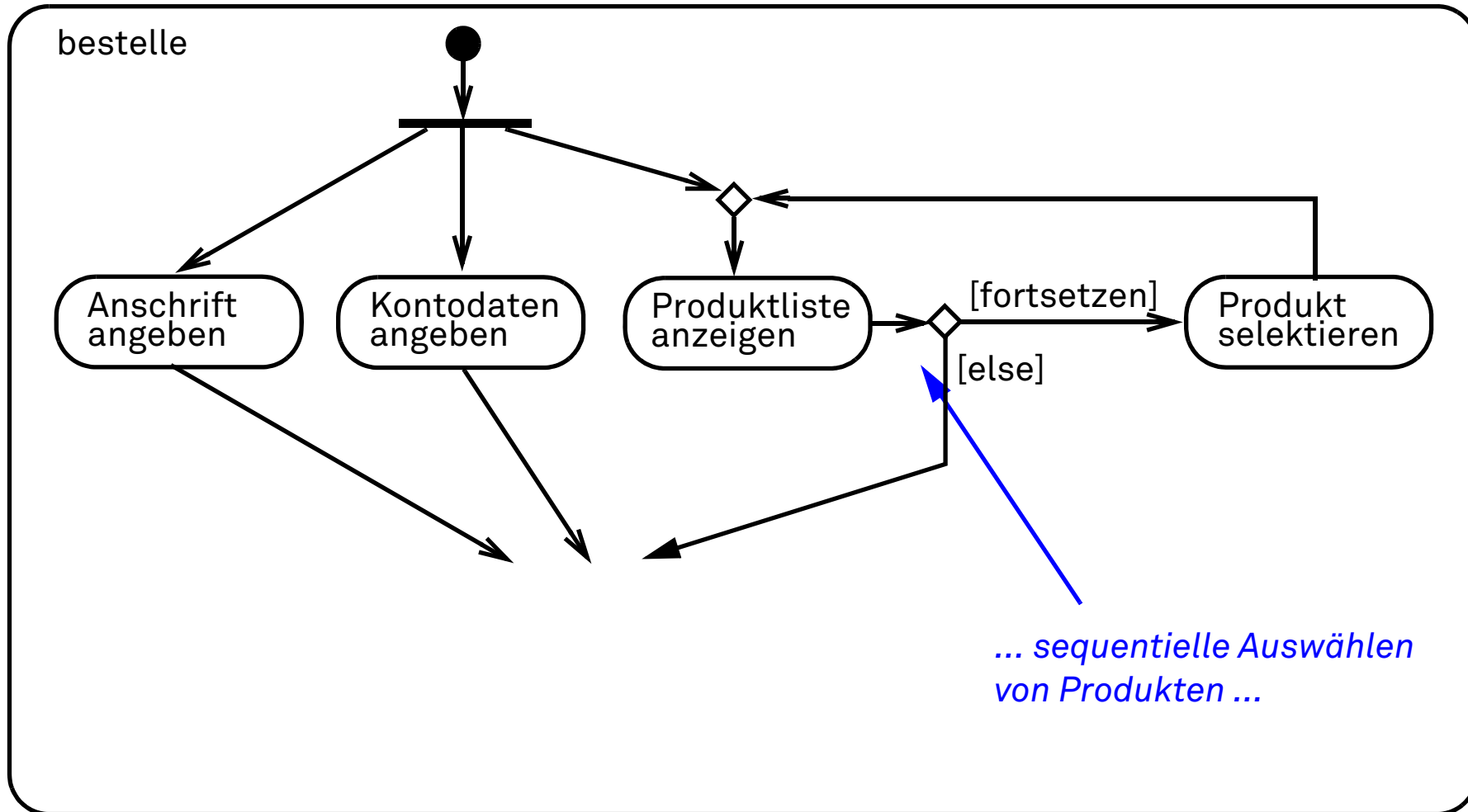
Beispiel – Nebenläufigkeit in Aktivitätsdiagrammen

(Fortsetzung)



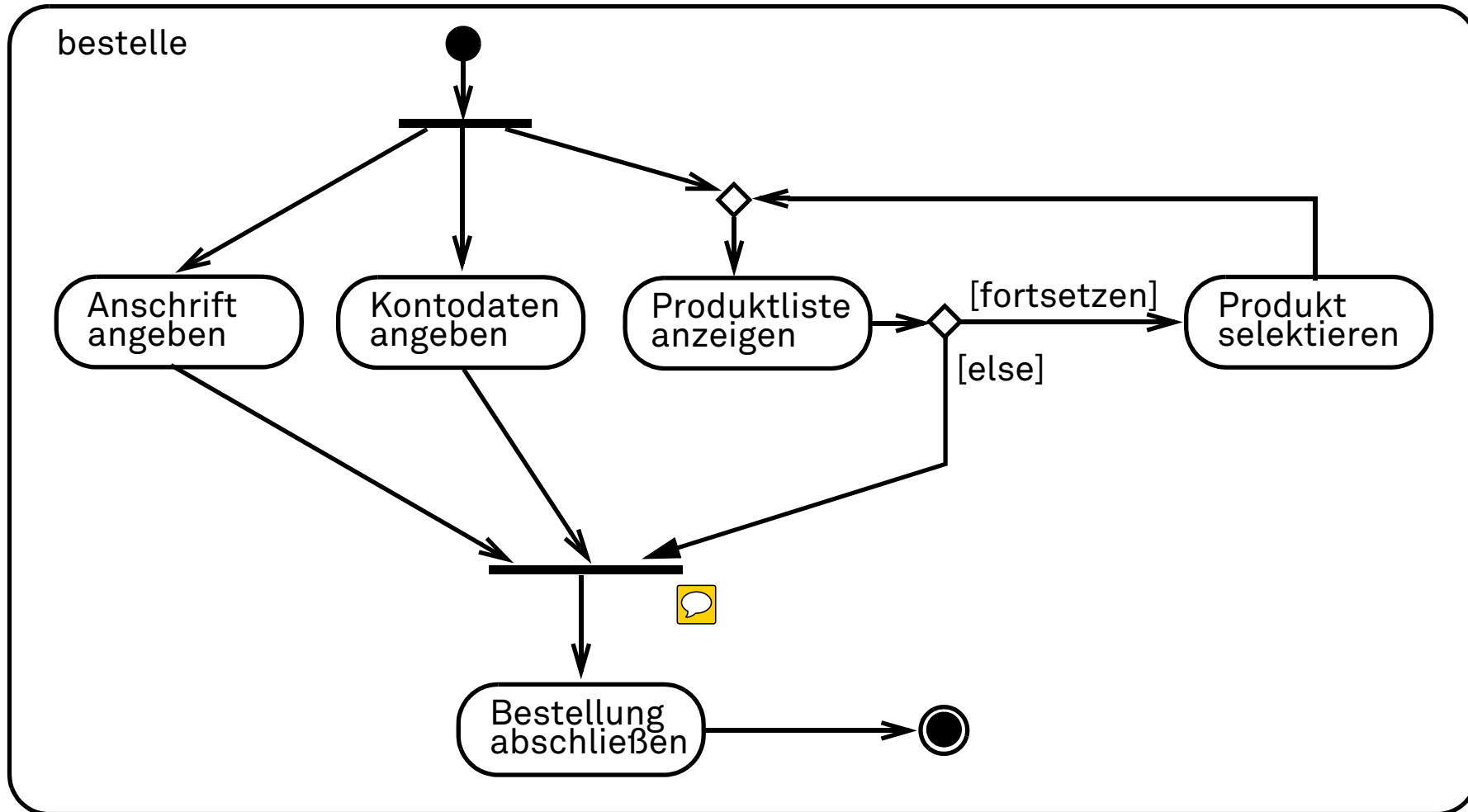
Beispiel – Nebenläufigkeit in Aktivitätsdiagrammen

(Fortsetzung)



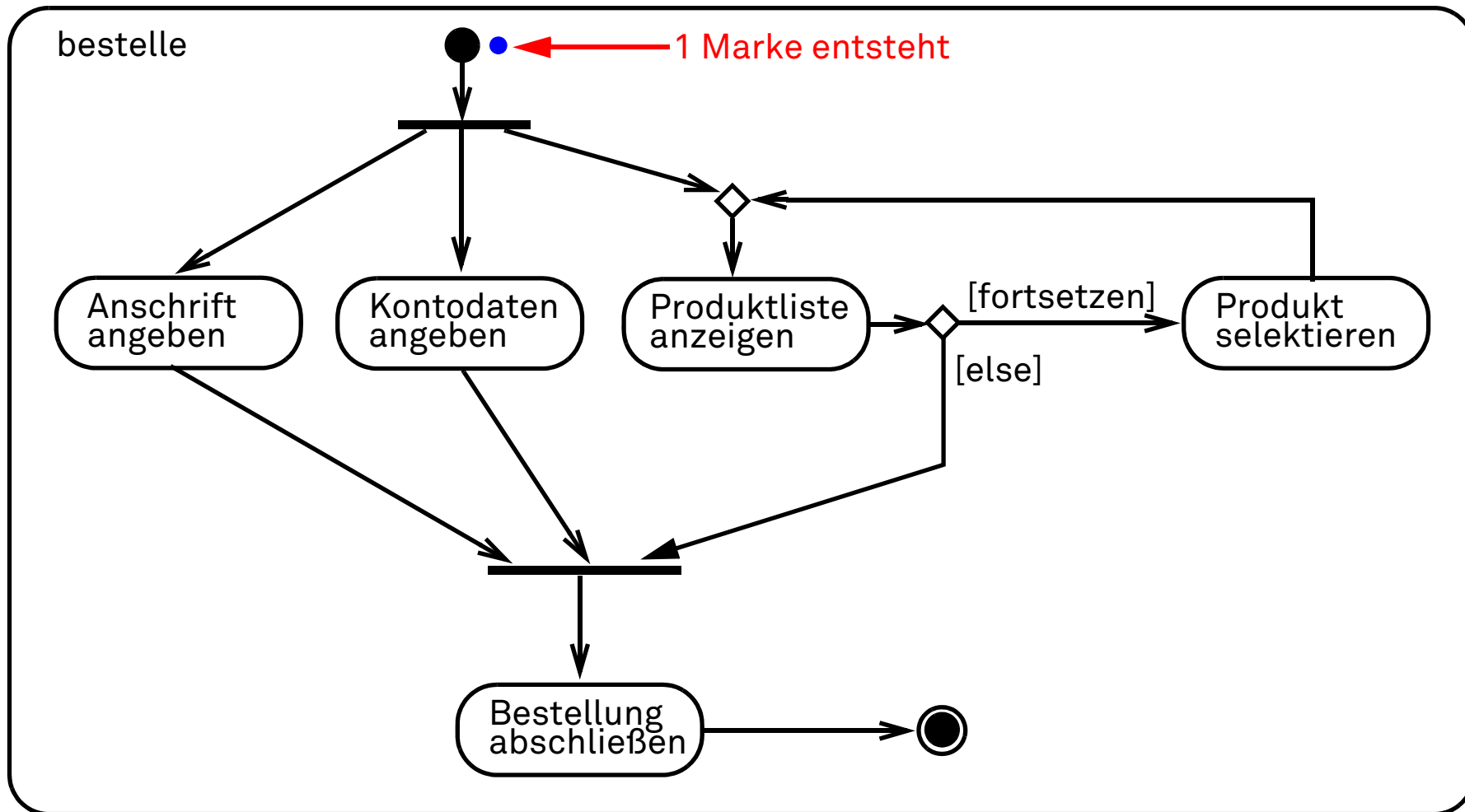
Beispiel – Nebenläufigkeit in Aktivitätsdiagrammen

(Fortsetzung)



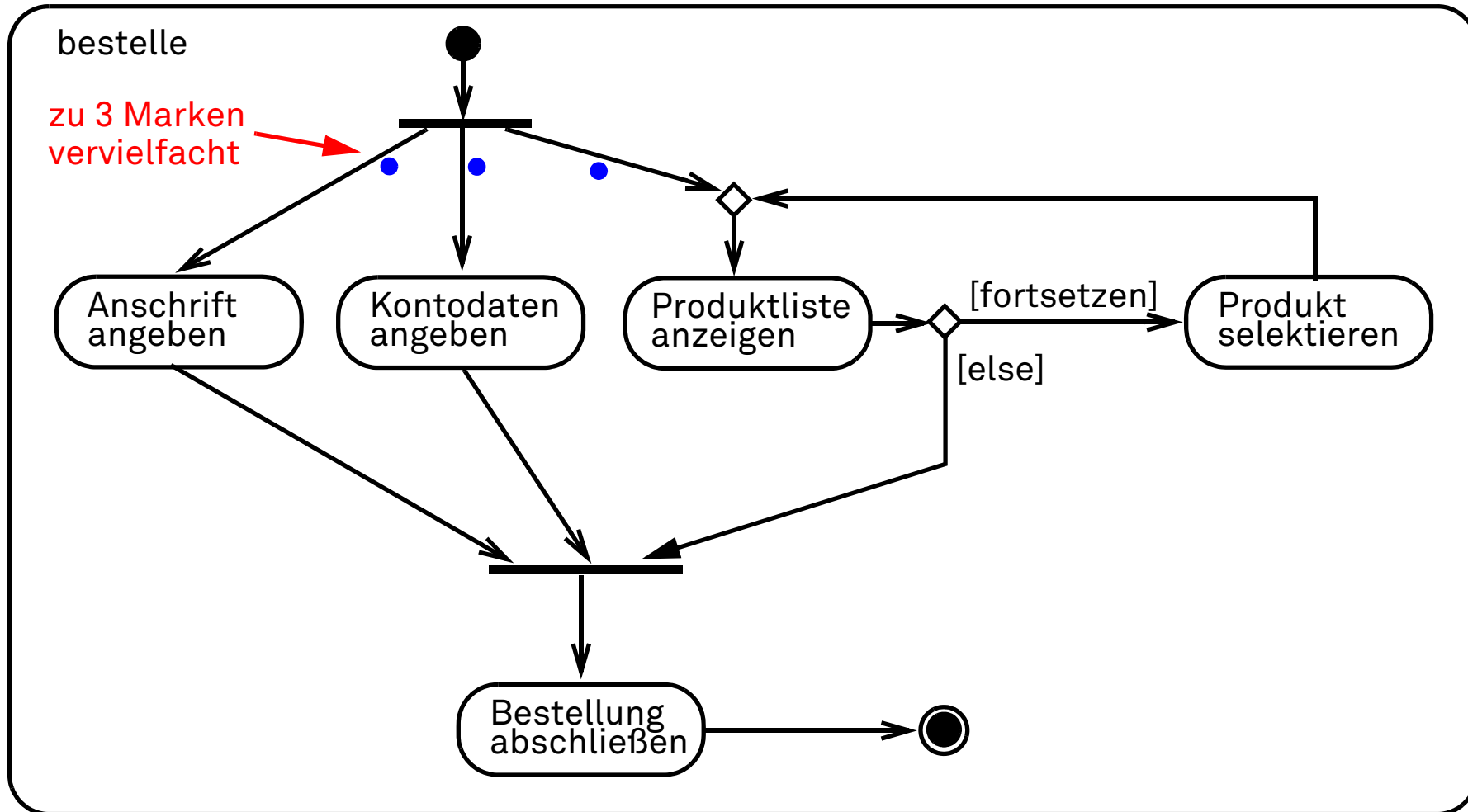
Beispiel – Nebenläufigkeit in Aktivitätsdiagrammen

(Fortsetzung)



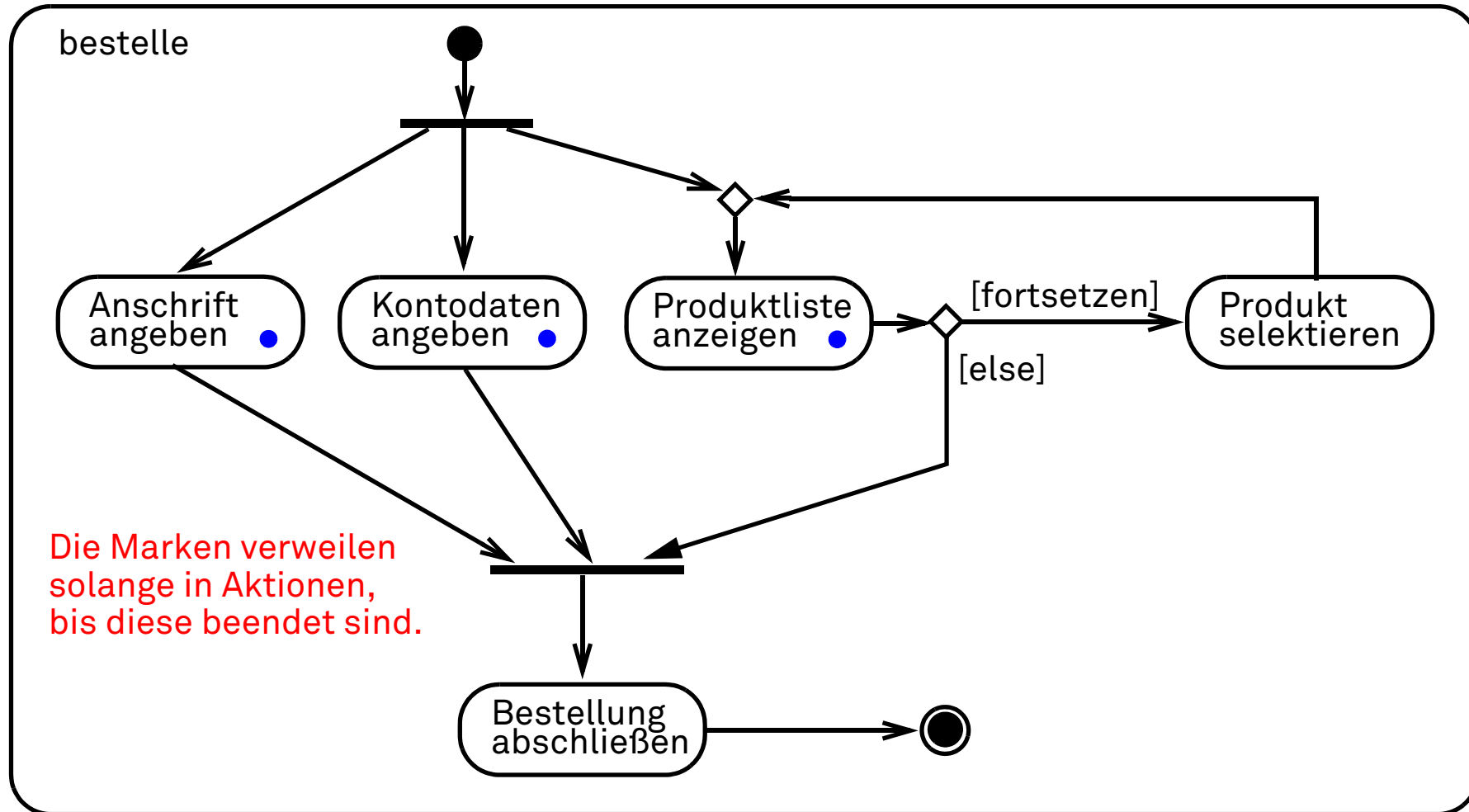
Beispiel – Nebenläufigkeit in Aktivitätsdiagrammen

(Fortsetzung)



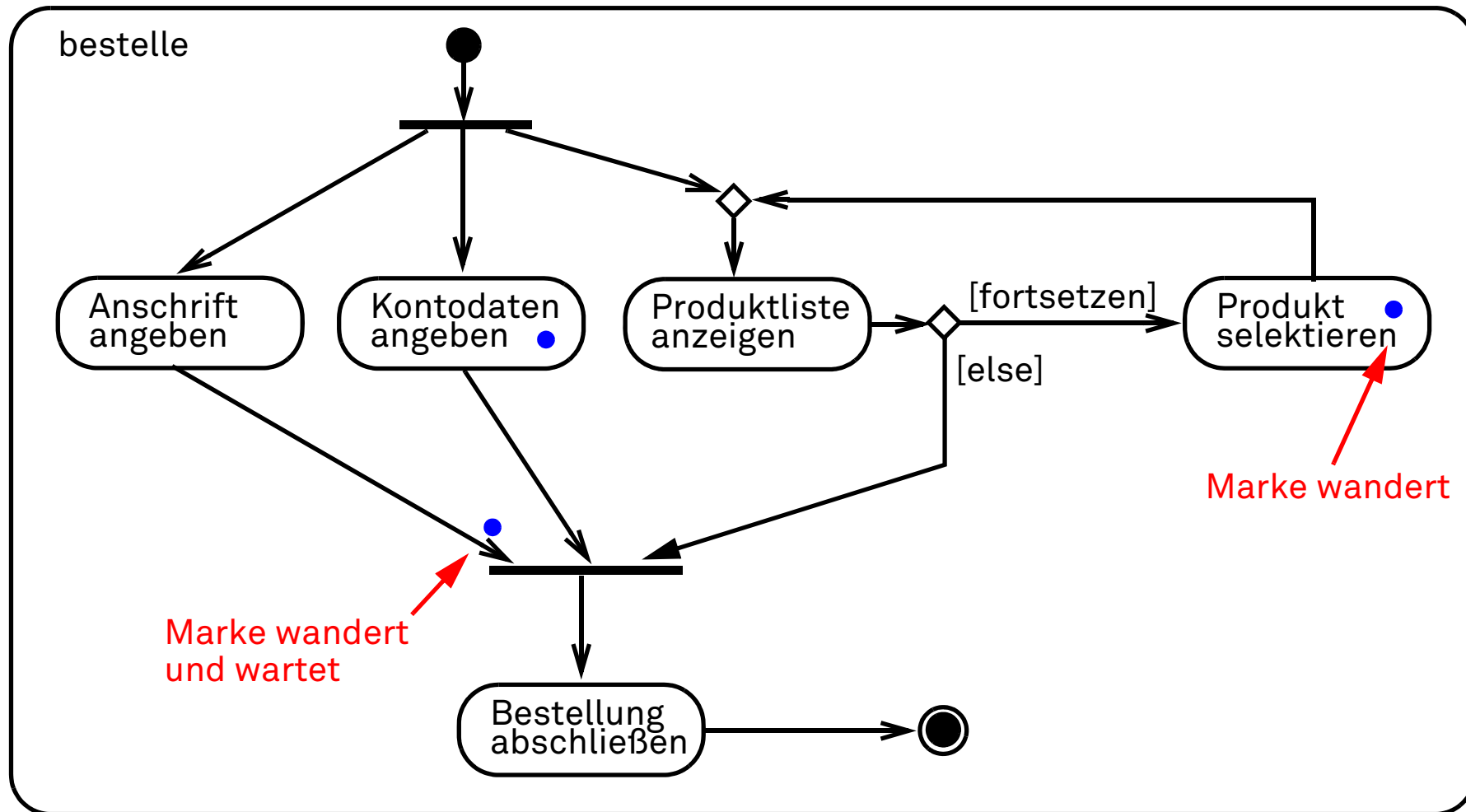
Beispiel – Nebenläufigkeit in Aktivitätsdiagrammen

(Fortsetzung)



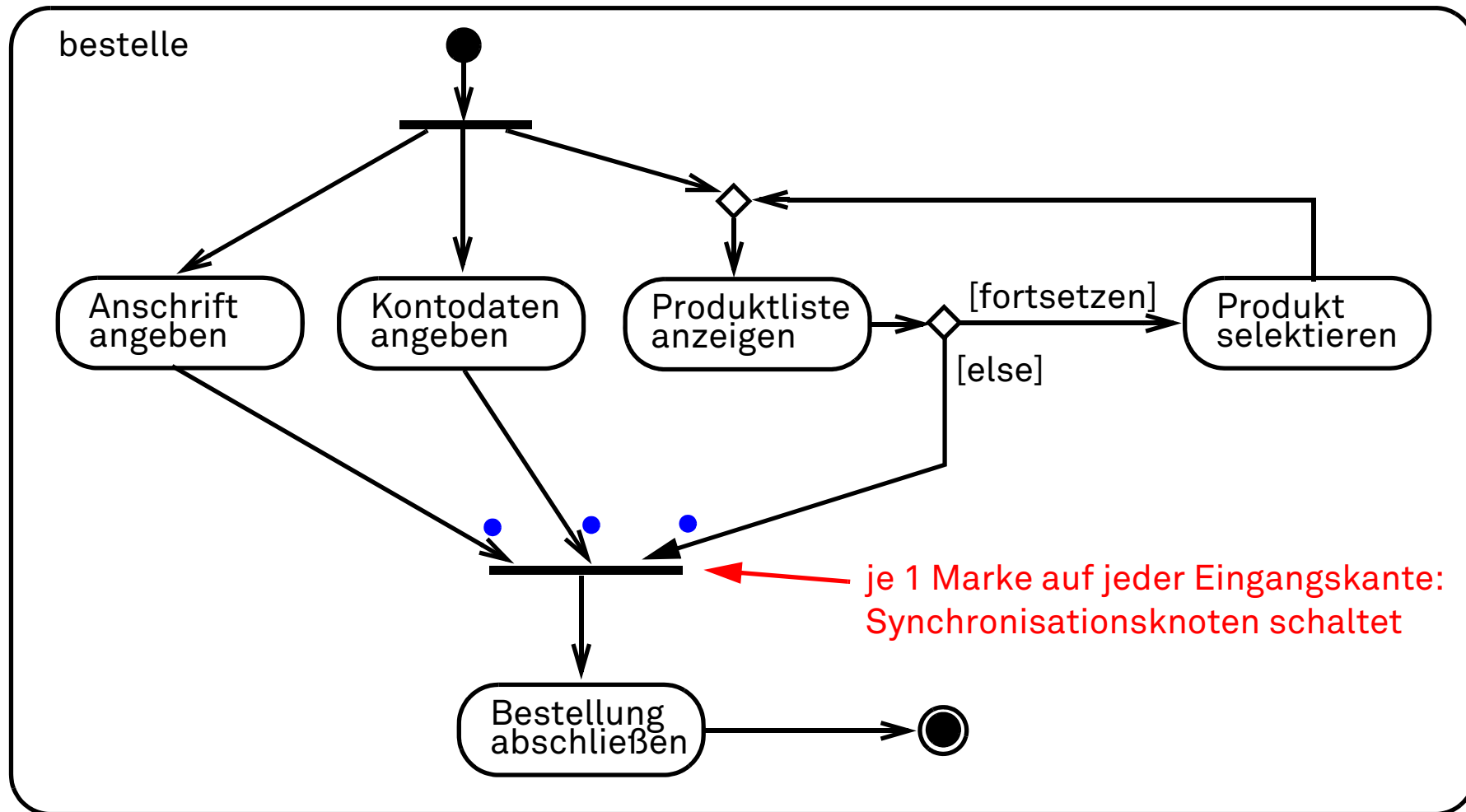
Beispiel – Nebenläufigkeit in Aktivitätsdiagrammen

(Fortsetzung)



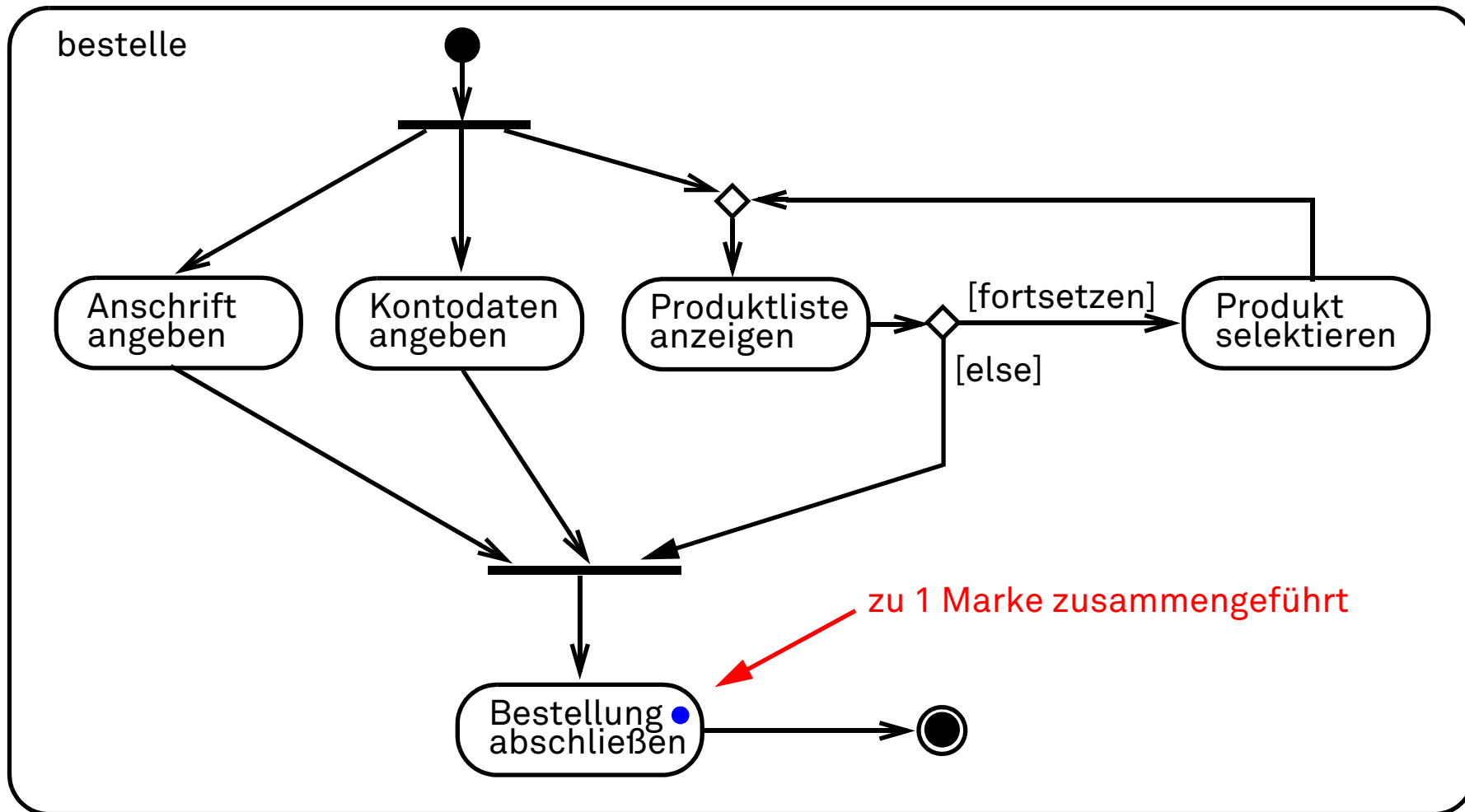
Beispiel – Nebenläufigkeit in Aktivitätsdiagrammen

(Fortsetzung)



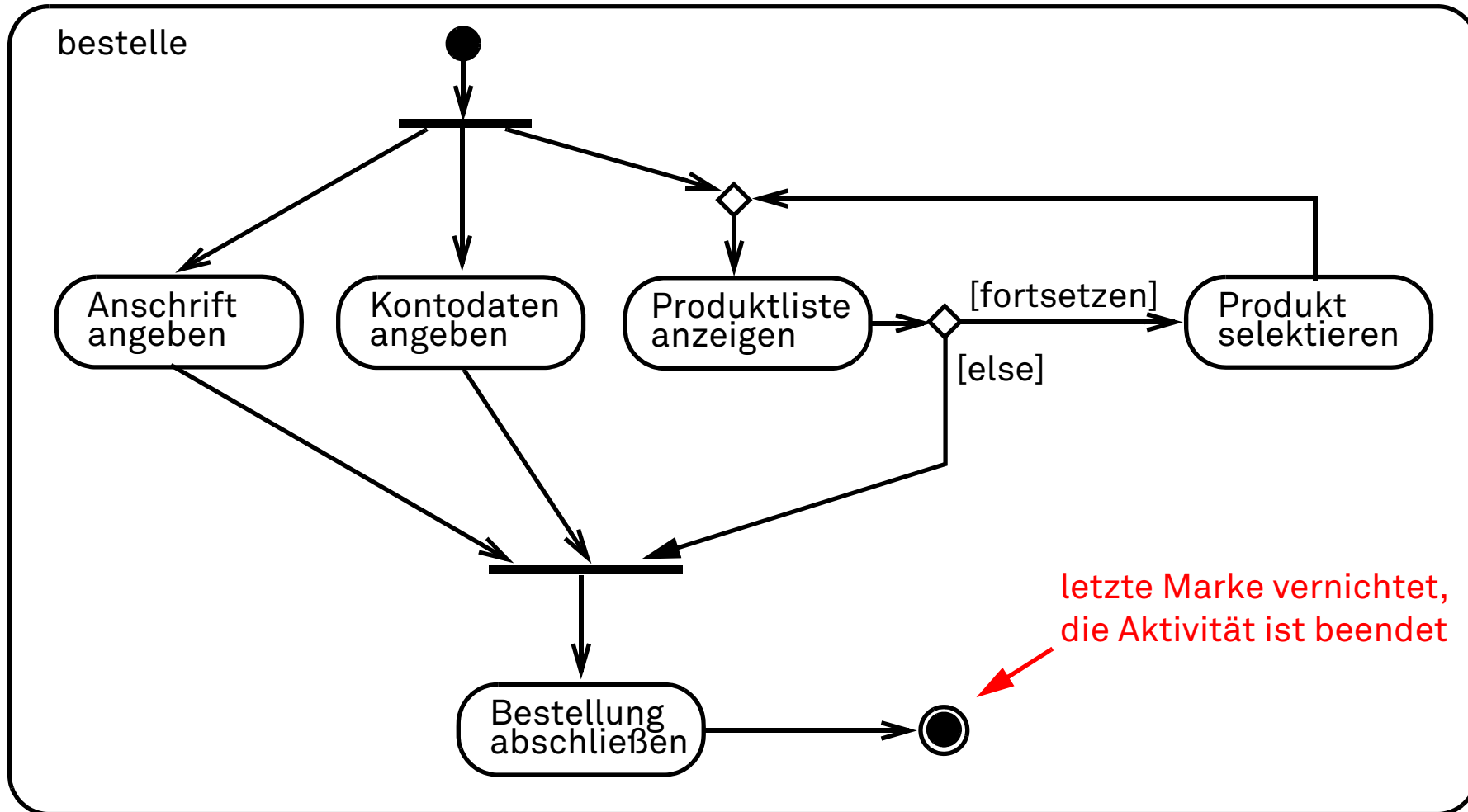
Beispiel – Nebenläufigkeit in Aktivitätsdiagrammen

(Fortsetzung)

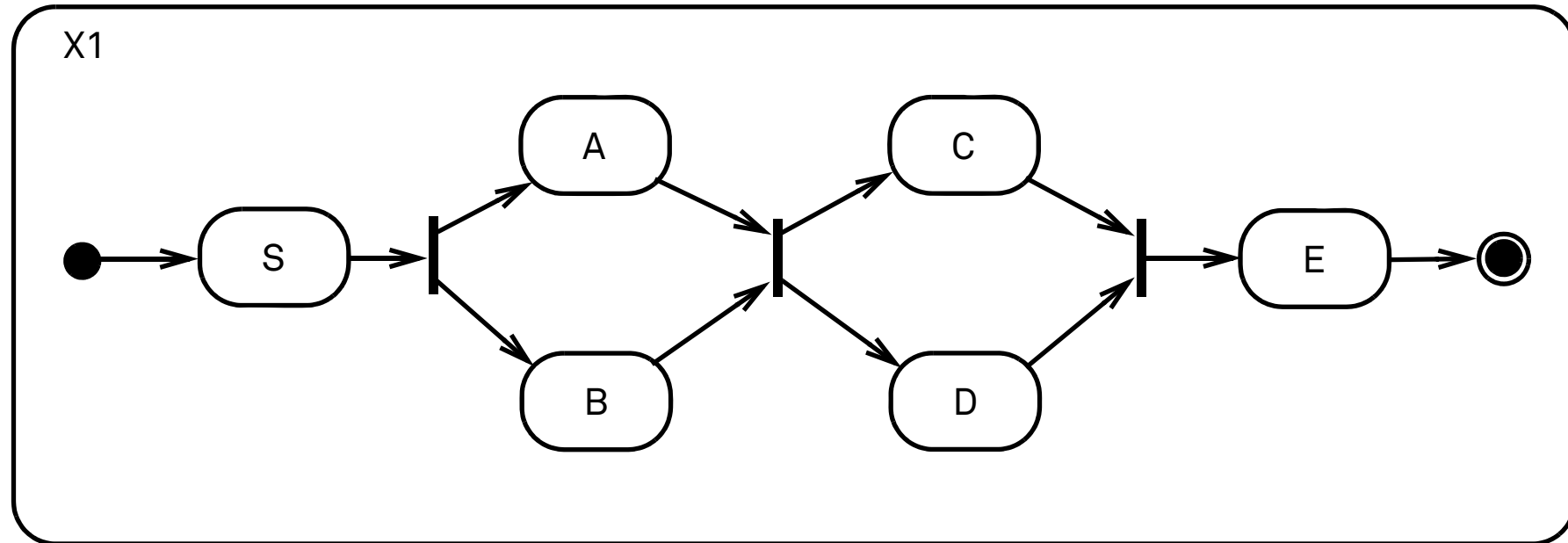


Beispiel – Nebenläufigkeit in Aktivitätsdiagrammen

(Fortsetzung)



Analyse von Aktivitätsdiagrammen – Beispiele

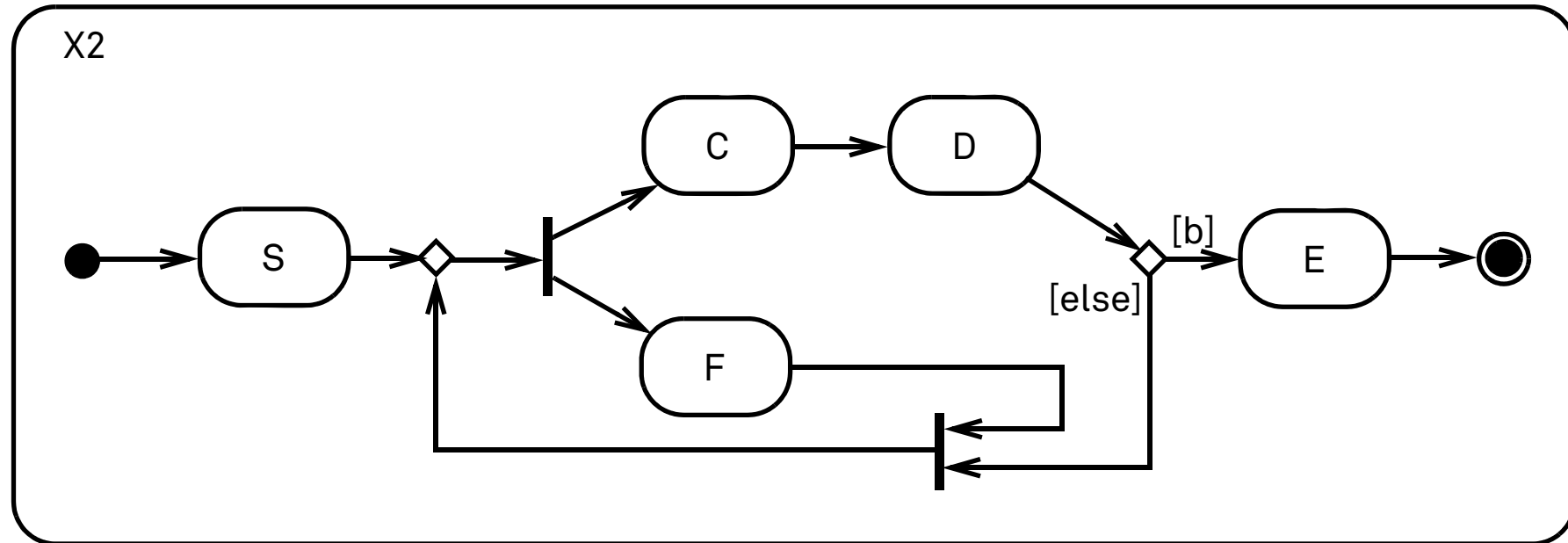


Aussagen, die sich aus diesem Diagramm ableiten lassen:

- ❑ Die Aktion S wird immer **vor** allen anderen Aktionen ausgeführt.
- ❑ A und B sowie C und D starten immer gleichzeitig.
- ❑ Die Aktionen A und B werden **immer vor** C und D ausgeführt und abgeschlossen.
- ❑ Die Aktion E wird erst nach Abschluss von S, A, B, C **und** D ausgeführt.
- ❑ Bei den Paaren A, B und C, D bestimmt jeweils die länger dauernde Aktion den Fortschritt des Kontrollflusses am folgenden Synchronisationsknoten.

Analyse von Aktivitätsdiagrammen – Beispiele

(Fortsetzung)

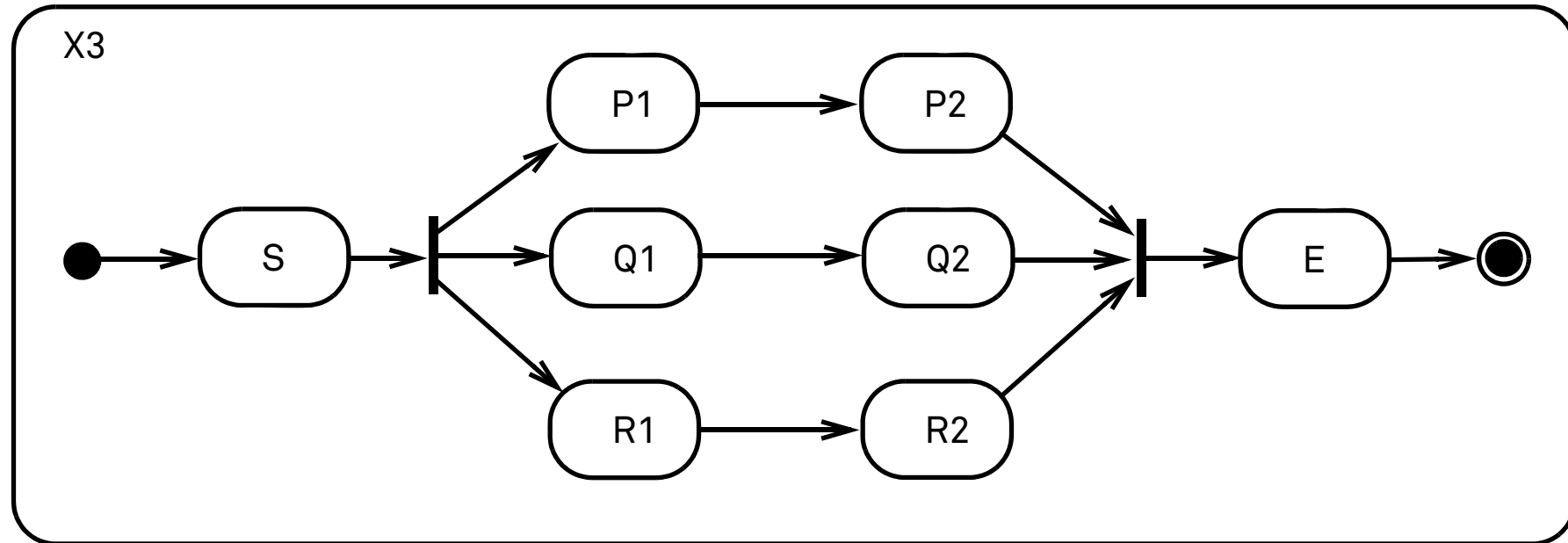


Aussagen:

- ❑ C und F starten gleichzeitig.
- ❑ Die Ausführung von Aktion C wird immer **vor** D abgeschlossen.
- ❑ Wenn nach Abschluss von D die Bedingung [b] **falsch** ist **und** F ausgeführt worden ist, wird der Durchlauf hinter S fortgesetzt.
- ❑ F kann vor oder gleichzeitig mit C ausgeführt werden und eventuell erst nach D enden.
- ❑ Ist [b] **wahr**, so wird die Aktivität über E beendet:
die in F oder hinter F verbliebene zweite Marke wird dann gelöscht.

Analyse von Aktivitätsdiagrammen – Beispiele

(Fortsetzung)



Aussagen:

- ☐ Die Aktion S wird **vor** allen anderen Aktionen ausgeführt.
- ☐ Die Aktionen P1, Q1, R1 starten zur **gleichen** Zeit.
- ☐ P1 wird **vor** dem Starten von P2 beendet, Q1 **vor** Q2, R1 **vor** R2.
- ☐ Die Aktion E wird **nach** Abschluss aller anderen Aktionen ausgeführt.
- ☐ Die weiteren Abläufe sind unbestimmt:
P2 kann z.B. parallel zu Q1 aber auch erst nach Q2 ausgeführt werden.

Zusammenfassung

Aktivitätsdiagramme

- ❑ dienen zur Beschreibung von Abläufen als Folgen von Aktionen,
- ❑ ermöglichen die graphische Beschreibung von nebenläufigem Verhalten.

Vorteile der Nutzung von Aktivitätsdiagrammen:

- ❑ Es sind unterschiedliche Stufen von Detaillierung und Abstraktion möglich.
- ❑ Die visuelle Überprüfung von Abläufen ist möglich.
- ❑ Die reduzierte Syntax der Diagramme erlaubt eine einfache Übertragung in eine formalisierte (und damit analysierbare) Beschreibung.

Grenzen der Visualisierung:

- ❑ Es werden i.d.R. nur Abläufe dargestellt.
- ❑ Der Umgang mit umfangreichen Diagrammen ist schwierig.
- ❑ Die Zwischenzustände der Ausführung ergeben sich aus der Belegung der Aktionen mit Marken. Die Bestimmung der Zwischenzustände benötigt eine Simulation.

Folien zur Vorlesung **Softwaretechnik**

Abschnitt 5.5: Zusammenfassung Analyse

Ermitteln von Anforderungen (Wiederholung Folie 507)




Die durchzuführenden Tätigkeiten sind:

- ❑ Anforderungen erfassen, beschreiben und verfeinern.
- ❑ Szenarien erfassen und beschreiben.
Ein *Szenario* beschreibt ein konkretes Beispiel für die Erfüllung oder auch Nicht-Erfüllung von einer oder mehreren Anforderungen und konkretisiert diese dadurch.
- ❑ Lösungsorientierte Anforderungsaspekte erfassen und beschreiben:
 - Datenperspektive (strukturelle Beziehungen von Daten)
 - Funktionsperspektive (Manipulation von Daten durch Funktionen)
 - Verhaltensperspektive (Reaktion auf externe Signale, Zustandsänderungen)

Ermitteln von Anforderungen (Wiederholung Folie 507)

(Fortsetzung)

Die durchzuführenden Tätigkeiten sind:






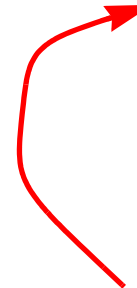
- ❑ Anforderungen erfassen, beschreiben und verfeinern. 
- ❑ Szenarien erfassen und beschreiben. 
Ein *Szenario* beschreibt ein konkretes Beispiel für die Erfüllung oder auch Nicht-Erfüllung von einer oder mehreren Anforderungen und konkretisiert diese dadurch.
- ❑ Lösungsorientierte Anforderungsaspekte erfassen und beschreiben:
 - Datenperspektive (strukturelle Beziehungen von Daten)
 - Funktionsperspektive (Manipulation von Daten durch Funktionen) 
 - Verhaltensperspektive (Reaktion auf externe Signale, Zustandsänderungen)

Anwendungsfalldiagramm
Sequenzdiagramm
Aktivitätsdiagramm

Ermitteln von Anforderungen (Wiederholung Folie 507)

(Fortsetzung)

Die durchzuführenden Tätigkeiten sind:

- ❑ Anforderungen erfassen, beschreiben und verfeinern. 
 - ❑ Szenarien erfassen und beschreiben.
Ein *Szenario* beschreibt ein konkretes Beispiel für die Erfüllung oder auch Nicht-Erfüllung von einer oder mehreren Anforderungen und konkretisiert diese dadurch. 
 - ❑ Lösungsorientierte Anforderungsaspekte erfassen und beschreiben:
 - Datenperspektive (strukturelle Beziehungen von Daten)
 - Funktionsperspektive (Manipulation von Daten durch Funktionen)
 - Verhaltensperspektive (Reaktion auf externe Signale, Zustandsänderungen)
- Anwendungsfalldiagramm**
Sequenzdiagramm
Aktivitätsdiagramm
Klassendiagramm

Klassendiagramm in der Analyse

Problembereich (auch als Domäne bezeichnet)
ist der Ausschnitt der Realität, in den das zu modellierende System eingebettet ist.

Problembereichsmodell (Domänenmodell)

- ❑ Modell der *Entitäten* (Dinge) des Problembereichs,
- ❑ die (vermutlich) als Informationen im System verwaltet werden müssen,
- ❑ zusammen mit ihren Beziehungen untereinander.

⇒ **Klassendiagramm**

Ein Klassendiagramm für die Analyse enthält

- ❑ keine Angaben zur technischen Realisierung (wie z.B. Zugriffsrechte),
- ❑ keine Angaben zu Navigationsrichtungen bei Assoziationen,
- ❑ möglicherweise keine oder nur wenige (wesentliche) Methoden.

Lastenheft/Pflichtenheft

- ❑ Lastenheft:
enthält die Gesamtheit der Forderungen an die Lieferungen und Leistungen,
die ein Auftraggeber erwartet. [DIN 69905 1997]
(Vorgabe des Auftraggebers)

- ❑ Pflichtenheft:
enthält die vom Auftragnehmer aufgrund des Lastenhefts erarbeiteten Leistungen und
beschreibt das zu erstellende System.
[DIN 69905 1997]
(Planung des Auftragnehmers)

- ❑ Bedeutung des Pflichtenhefts:
 - vertragliche Basis zwischen Auftraggeber und Auftragnehmer,
 - Planungsgrundlage für den Auftragnehmer,
 - inhaltliche Vorgabe für die Arbeit des Auftragnehmers,
 - inhaltliche Vorgabe für die Prüfung des Ergebnisses der Arbeit des Auftragnehmers.

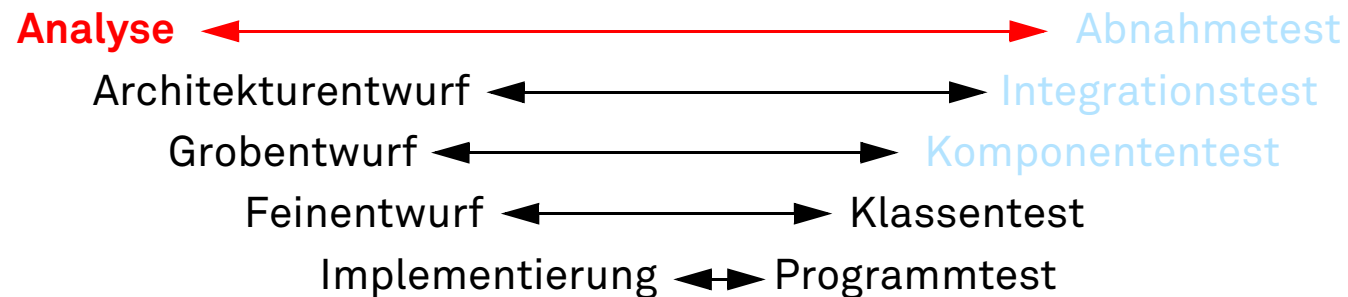
Aufbau eines Pflichtenhefts



Abschnitt	Inhalt
Erstellungsdaten	Autoren, Version, Erstellungsdatum, Änderungshistorie
Geschäftsfeld	Anwendungsbereich, Zielgruppen, Betriebsbedingungen, Modell des Problembereichs
Glossar	Erklärung fachlicher Begriffe
Zielbestimmung	Muss- und Wunschziele auflisten
Geschäftsprozesse	Beschreibung der Geschäftsprozesse durch Anwendungsfalldiagramme oder Aktivitätsdiagramme
Produktfunktionen	Beschreibung der Produktfunktionen durch Anwendungsfalldiagramme oder Aktivitätsdiagramme, Erläuterung durch Sequenzdiagramme, eventuell auch Angabe von Skizzen für Benutzungsschnittstellen
Produktdaten	Beschreibung der langfristig zu verwaltenden Daten (Klassendiagramm)
Qualität	Beschreibung der Qualitätsanforderungen
Testfälle	Testfälle zu einzelnen Funktionen und Qualitätsanforderungen
organisatorische Voraussetzungen	vorausgesetzte Hardware, vorausgesetzte Software, vorausgesetzte Abläufe

Zusammenfassung

V-Modell



als Notationen kennengelernt:

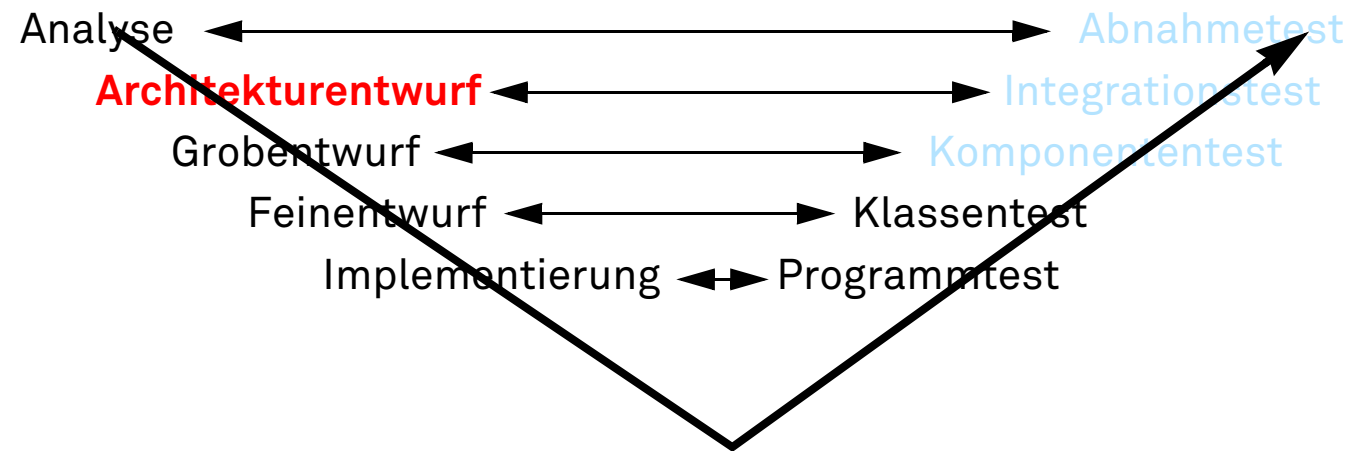
- | | |
|--|--------------------------|
| <input type="checkbox"/> Anwendungsfalldiagramme | Analyse |
| <input type="checkbox"/> Klassendiagramme | Analyse, Implementierung |
| <input type="checkbox"/> Objektdiagramme | |
| <input type="checkbox"/> Aktivitätsdiagramme | Analyse, Implementierung |
| <input type="checkbox"/> Sequenzdiagramme | Analyse, Implementierung |

Folien zur Vorlesung **Softwaretechnik**

Abschnitt 5.6: Übergang Analyse – Architekturentwurf

Architekturentwurf

V-Modell



in der Realität gilt:

- ❑ Umfangreiche Systeme bestehen aus mehreren ausführbaren Programmen.
- ❑ Die Entwicklung verschiedener Teile der Software findet in verschiedenen (Teil-)Projekten an verschiedenen Orten und zu verschiedenen Zeiten statt.

Lösung:

Es muss eine übergreifende **Softwarearchitektur** festgelegt werden.

Softwarearchitektur

ist die strukturierte Anordnung von Komponenten
mit einer Beschreibung ihrer Beziehungen.
Dabei bilden die Komponenten eine Zerlegung des Gesamtsystems.

Das Festlegen einer Softwarearchitektur erfolgt **als frühe** technische Entscheidung
aufgrund der Anforderungen aus der Analyse:

Architekturentwurf

Anmerkungen:

- ❑ Softwarearchitektur ist ein unscharfer, weit gefasster Begriff.
- ❑ Eine Softwarearchitektur beschreibt sehr abstrakt den groben Aufbau eines Softwareprodukts.
- ❑ Die Softwarearchitektur wird meist für jedes Softwareprodukt individuell festgelegt, es gibt aber einige allgemeine Stilarten.
- ❑ Für einzelne Anwendungsbereiche gibt es vordefinierte Architekturen:
 - Framework
 - Application Server

Beispiel: Softwareunterstützung für eine (Ausleih-)Bibliothek

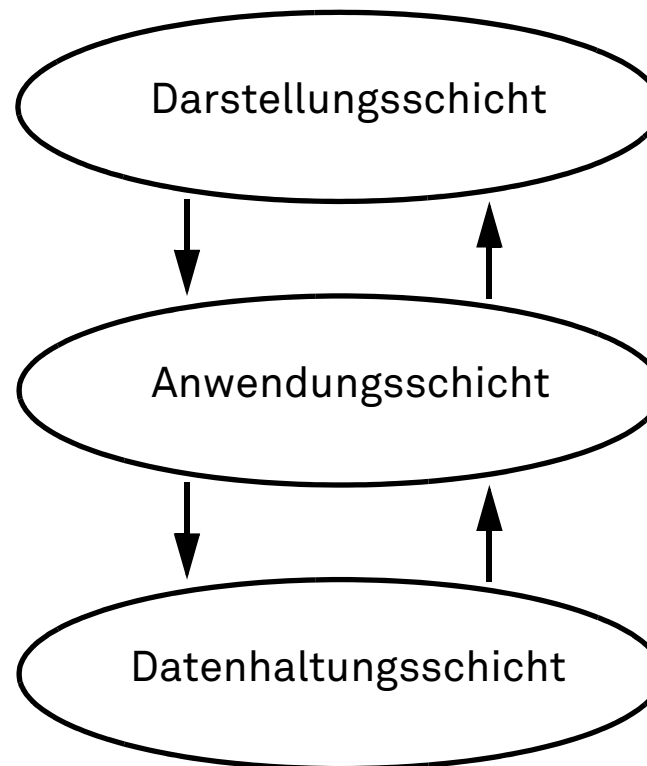
- ❑ Es werden verschiedene Datenmengen verwaltet:
 - Bücher
 - Leser
 - Buchhändler
 - Verlage
 - ❑ Es treten verschiedene Geschäftsvorfälle auf:
 - Recherche, Reservierung, Ausleihe, Rückgabe, Verlängerung, Mahnwesen, Gebührenberechnung, Beschaffung, Inventarisierung, ...
 - ❑ Es gibt verschiedene Akteure:
 - Leser an verschiedenen Orten
 - Personal mit verschiedenen Aufgaben an verschiedenen Orten
- ⇒ Es werden verschiedene, voneinander unabhängige Softwarekomponenten benötigt, die auf unterschiedlicher Hardware betrieben werden müssen.
- ❑ Es gibt einige wichtige Qualitätsanforderungen:
 - Skalierbarkeit, Zuverlässigkeit, Robustheit, Wartbarkeit

Beispiel: Softwareunterstützung für eine (Ausleih-)Bibliothek

(Fortsetzung)

mögliche Lösung:

3-Schichten-Architektur
(nimmt eine Aufteilung anhand **technischer** Aufgaben vor)

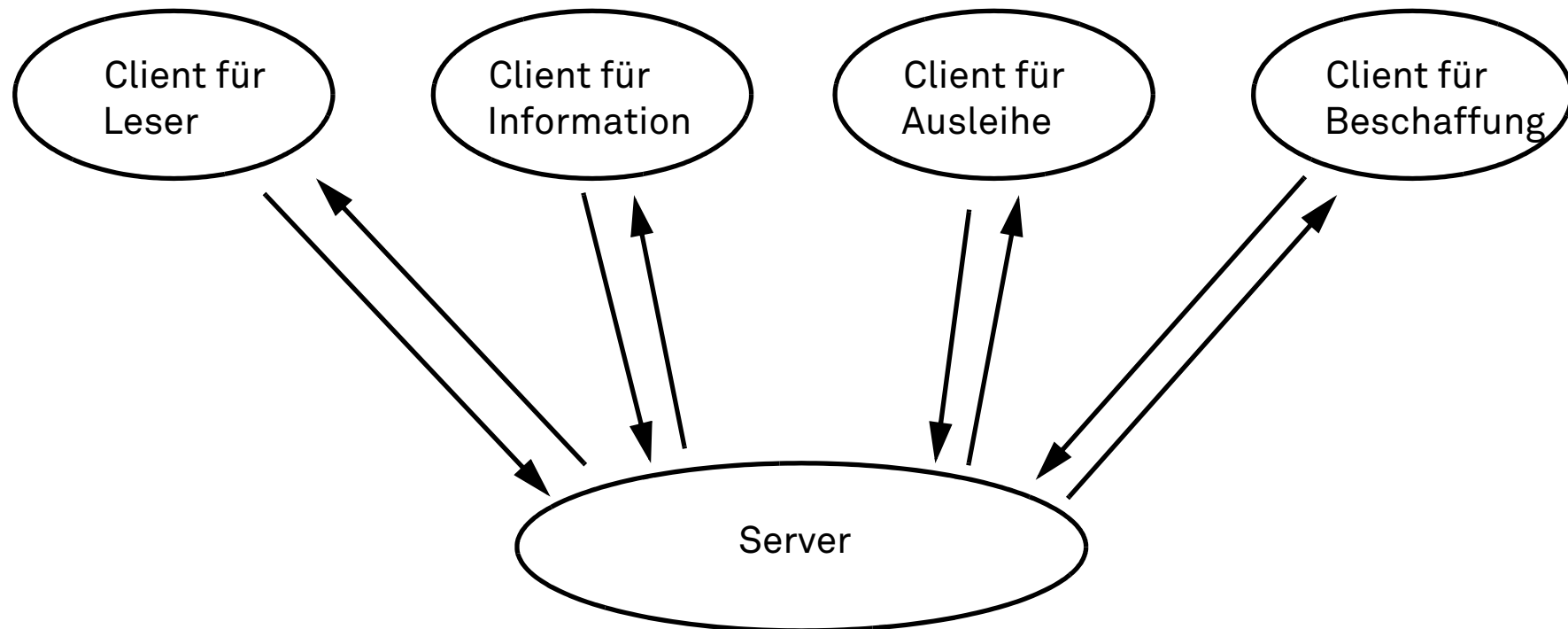


Beispiel: Softwareunterstützung für eine (Ausleih-)Bibliothek

(Fortsetzung)

andere Lösung:

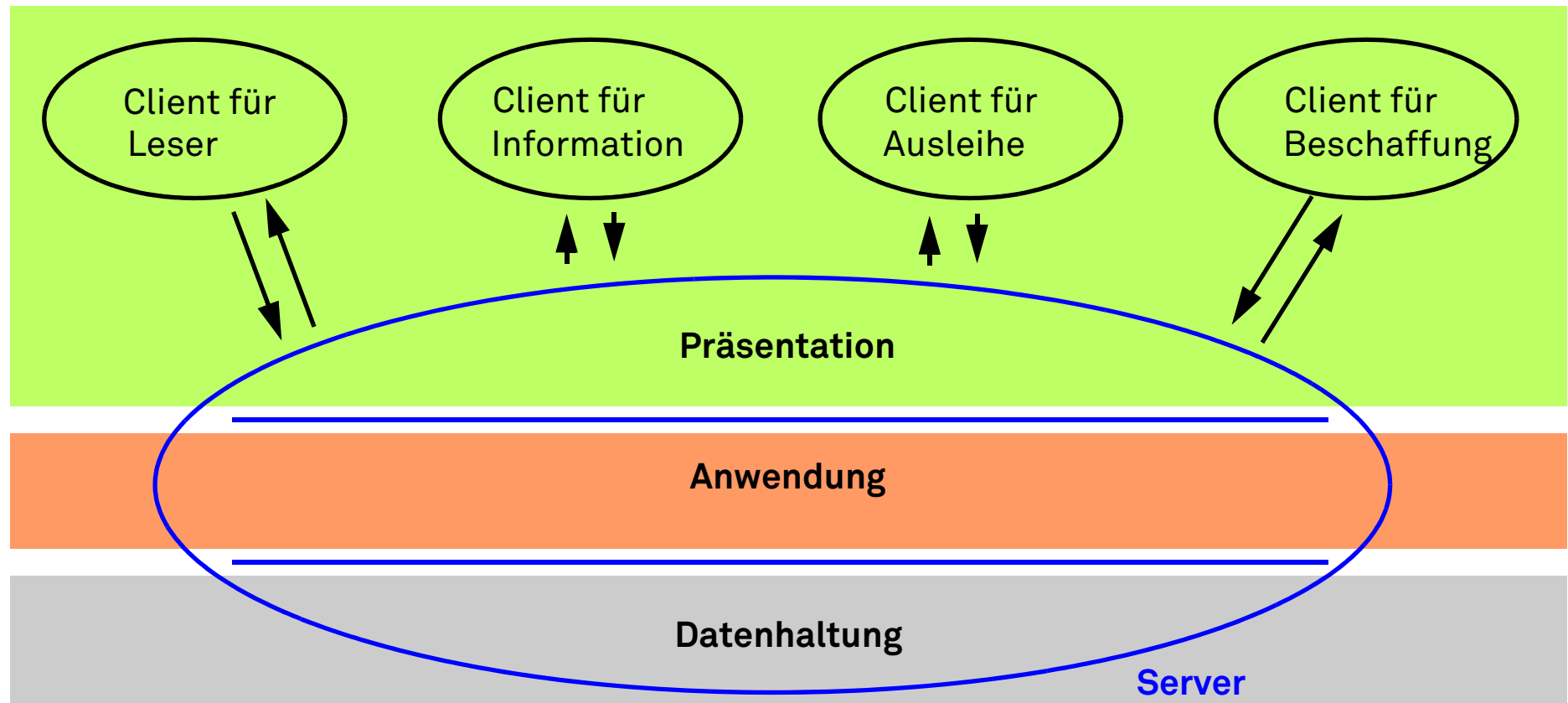
Client-Server-Architektur
(nimmt eine Aufteilung anhand *räumlicher* Aspekte vor)



Beispiel: Softwareunterstützung für eine (Ausleih-)Bibliothek

(Fortsetzung)

Kombination: 3-Schichten-Client-Server-Architektur



Beispiel: Softwareunterstützung für eine (Ausleih-)Bibliothek

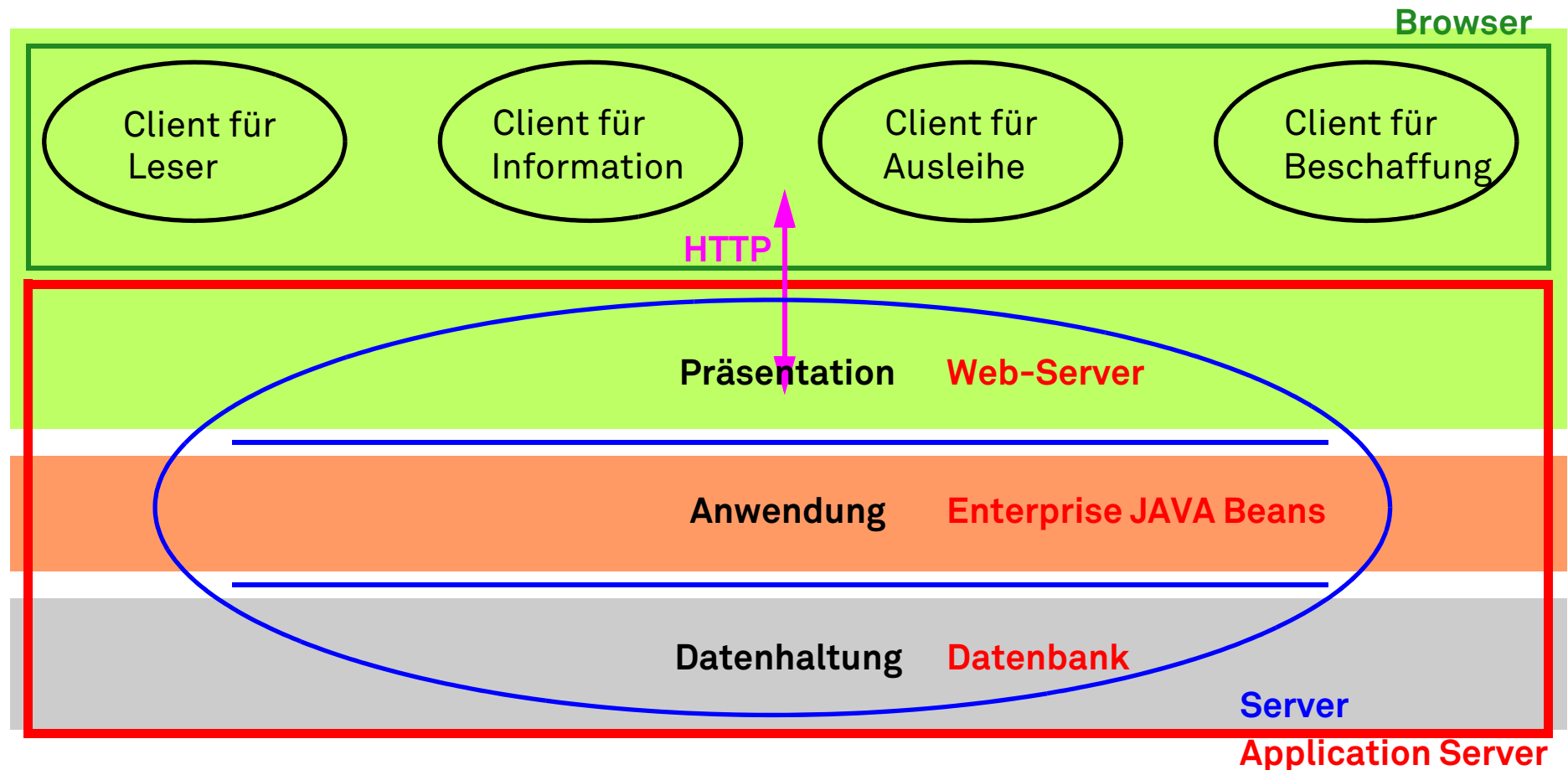
(Fortsetzung)

Kombination:

3-Schichten-Client-Server-Architektur

Realisierung:

durch den Einsatz eines **Application Servers**



Architekturstil

gibt grobe Restriktionen für die Beziehungsstrukturen zwischen den Komponenten der Architektur vor.

- ❑ Stile für die Organisation von Komponenten:
 - Schichtenarchitektur
 - Pipes-and-Filters-Architektur
 - Repository-basierte Architektur
- ❑ Stil für verteilte Systeme:
 - Client-Server-Architektur

Gemeinsamkeiten aller Architekturen sind:

- ❑ Komponenten und
- ❑ Konnektoren/Kommunikationskanäle zwischen Komponenten.

Anmerkung:

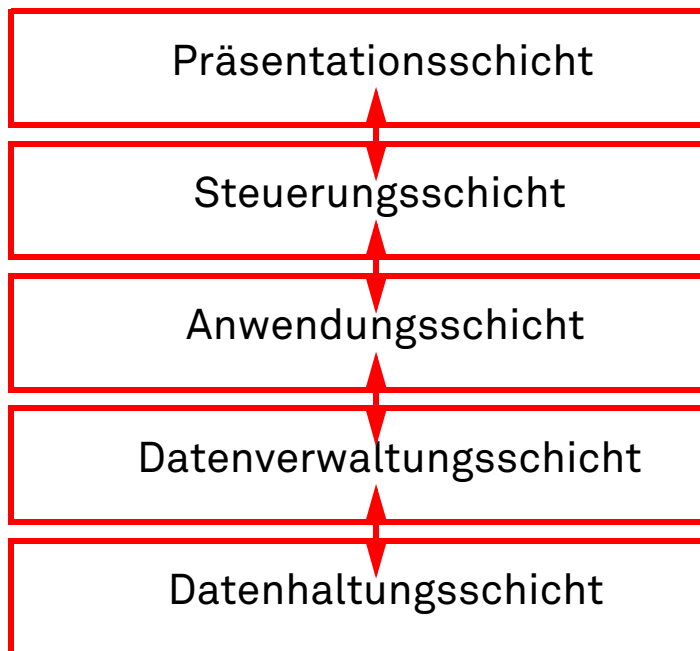
Die konkrete Architektur bestimmt die Qualität des Softwareprodukts:

- ❑ Performanz wird auch vom Kommunikationsaufwand zwischen den Komponenten beeinflusst.
- ❑ Robustheit hängt auch von den zwischen den Komponenten genutzten Protokollen ab.
- ❑ Skalierbarkeit ergibt sich aus der Möglichkeit, die Anzahl ausgewählter Komponenten anzupassen.
- ❑ Zuverlässigkeit kann durch redundante Komponenten verbessert werden.
- ❑ Wartbarkeit verbessert sich, wenn Komponenten austauschbar sind.

⇒ Der Architekturentwurf hat großen Einfluss auf das entstehende Softwaresystem und muss sorgfältig geplant werden.

Architekturstil »Schichtenarchitektur«

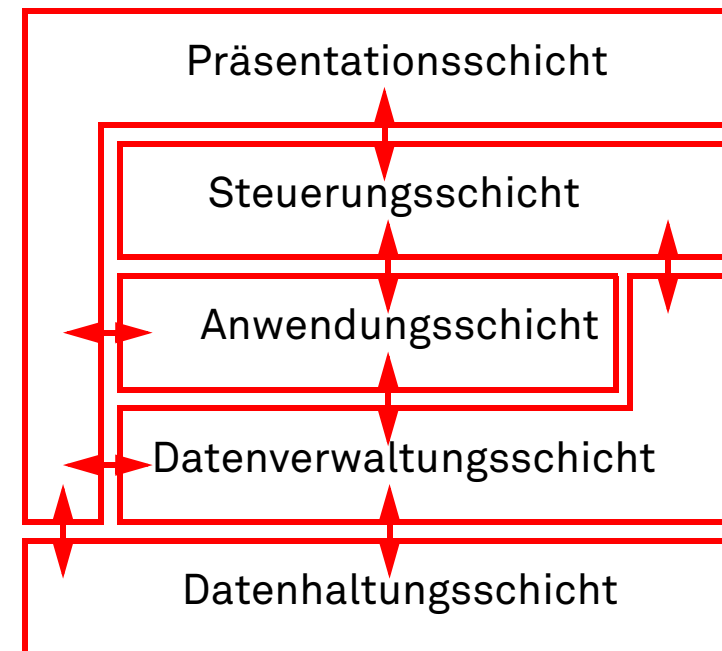
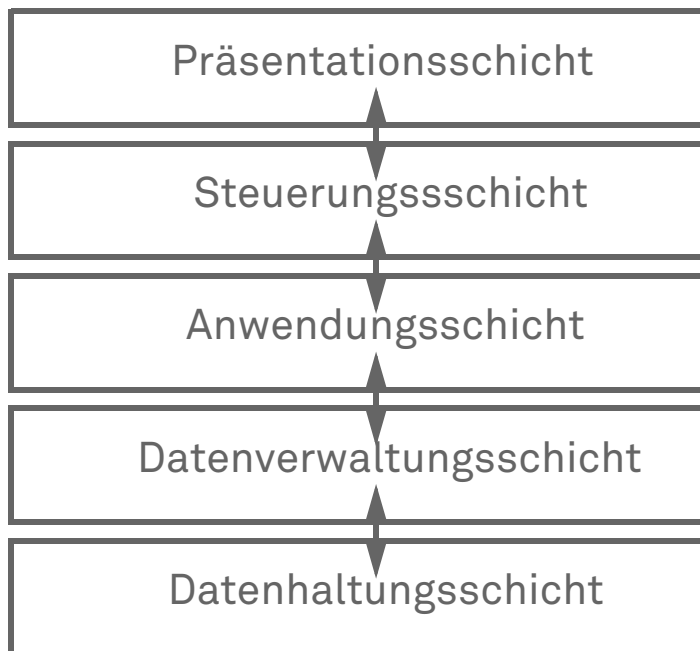
- ❑ Die Komponenten werden auf semantisch zusammenhängende Schichten verteilt. (Statt Schicht wird auch von *Ebene* (engl. *tier* oder *layer*) gesprochen.)
- ❑ Innerhalb einer Schicht können die Komponenten beliebig kommunizieren.
- ❑ Meist ist nur die Kommunikation zwischen benachbarten Schichten erlaubt.



Architekturstil »Schichtenarchitektur«

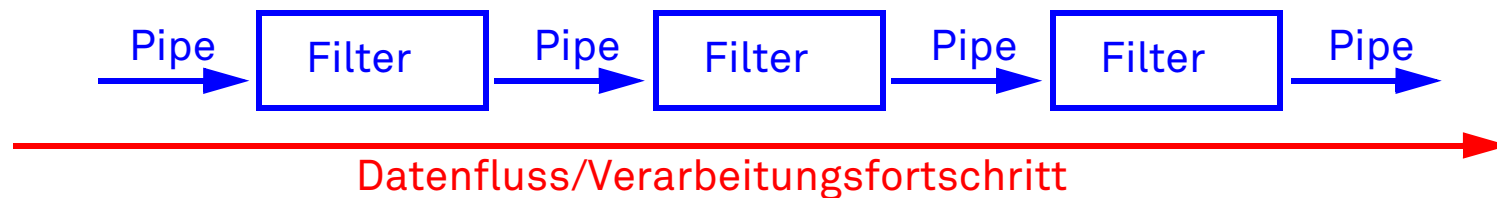
(Fortsetzung)

- ❑ Die Komponenten werden auf semantisch zusammenhängende Schichten verteilt. (Statt Schicht wird auch von *Ebene* (engl. *tier* oder *layer*) gesprochen.)
- ❑ Innerhalb einer Schicht können die Komponenten beliebig kommunizieren.
- ❑ Meist ist nur die Kommunikation zwischen benachbarten Schichten erlaubt.
- ❑ Bei vielschichtigen Architekturen lässt sich diese strikte Begrenzung der Kommunikation nicht immer durchhalten (z.B. beim Auftreten von Performanzproblemen).



Architekturstil »Pipes-and-Filters-Architektur«

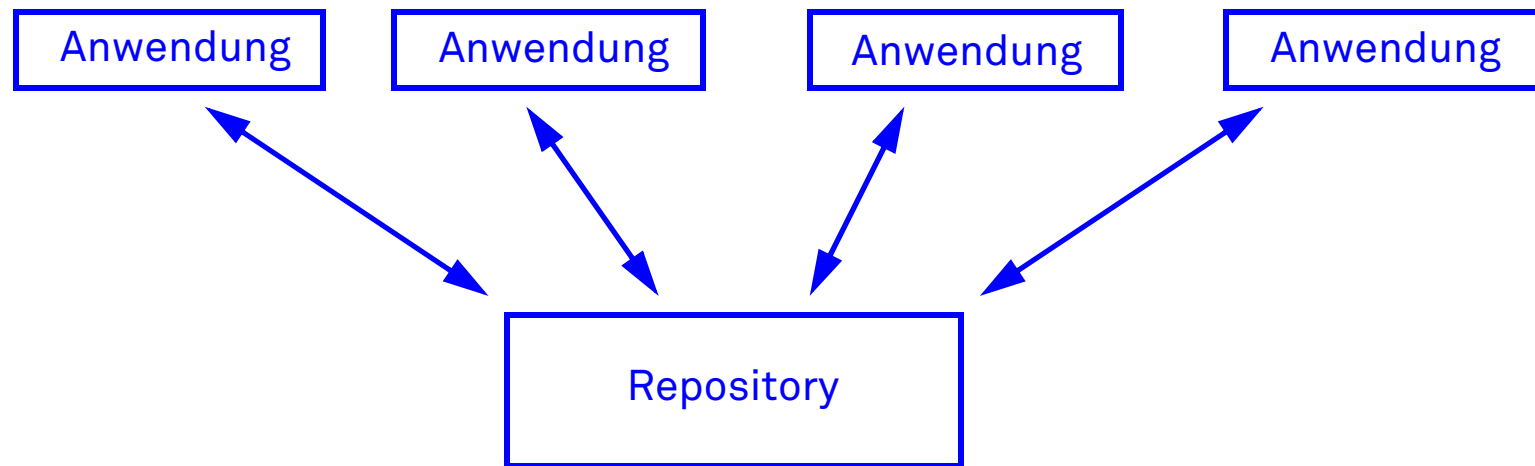
- ❑ Filter (= Komponente)
 - erfüllt einen Verarbeitungsschritt: Eingabe wird in Ausgabe umgewandelt,
 - kennt weder Vorgänger noch Nachfolger,
 - kooperiert dementsprechend nicht mit den benachbarten Komponenten.
- ❑ Pipe (= Konnektor)
 - verbindet zwei Filter, so dass nur ein sequentieller Ablauf möglich ist.



- ❑ Vorteile: Das System lässt sich leicht anpassen, ergänzen oder umkonfigurieren.
- ❑ Variationen:
 - ohne oder mit Verzweigungen,
 - ohne oder mit Zyklen (wiederholtes Durchlaufen des gleichen Filters),
- ❑ Hinweis: Pipes-and-Filters ist der Architekturstil in der *funktionalen Programmierung*.

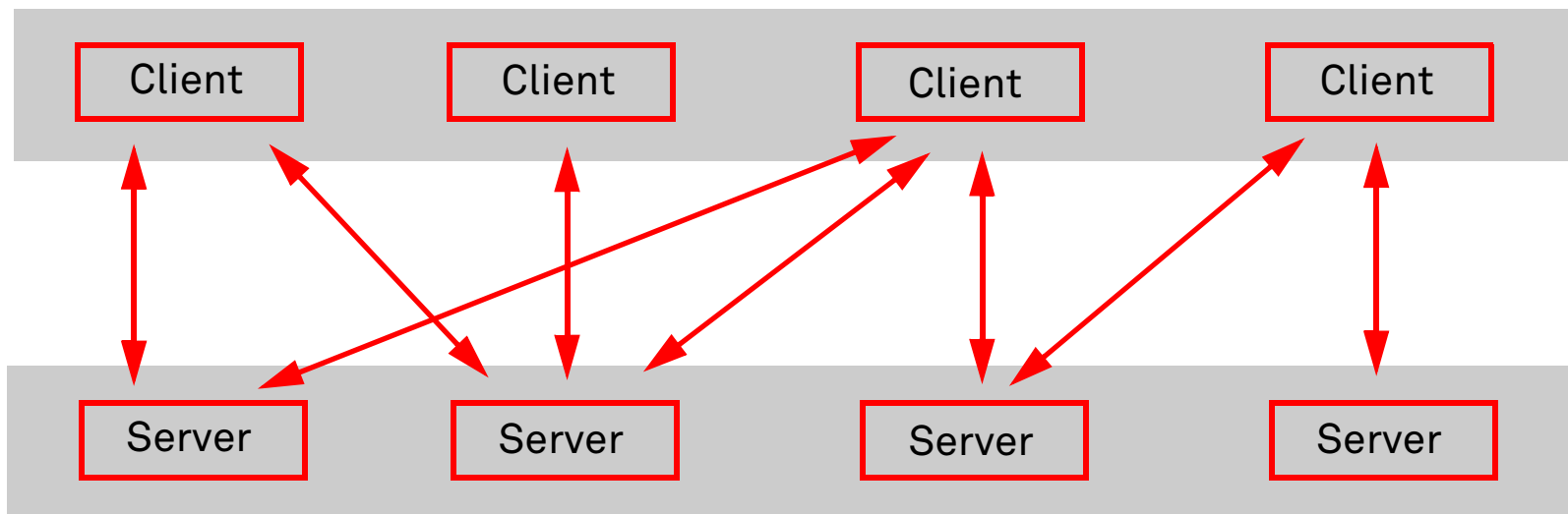
Architekturstil »Repository-basierte Architektur«

- ❑ Es werden 2 Arten von Komponenten unterschieden:
 - zentraler Datenspeicher (Repository) und die
 - auf dem Datenspeicher operierenden Anwendungen.
- ❑ Konnektoren existieren nur zwischen Datenspeicher und jeder Anwendung.
- ❑ Variation: Die Anwendungen werden in einer vorgegebenen Reihenfolge ausgeführt.



Architekturstil »Client-Server-Architektur«

- ❑ spezielle 2-Schichten-Architektur mit
 - Server-Schicht (mit möglicherweise mehreren Servern)
 - Client-Schicht (mit möglicherweise mehreren Clients)
- ❑ Konnektoren existieren nur zwischen Server und Client.
- ❑ Kommunikationsprinzip: Client fordert eine Leistung des Servers über ein Netzwerk an.
- ❑ Variation: Komponenten können gleichzeitig die Rollen *Server* und *Client* übernehmen.

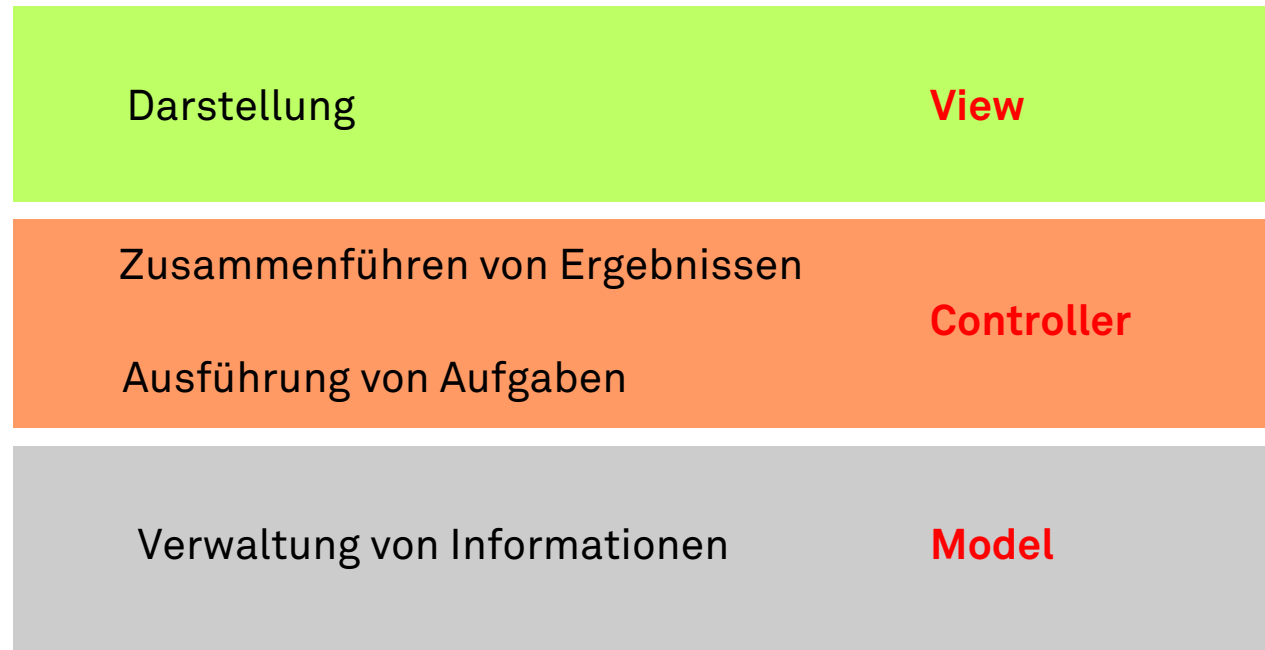


Model – View – Controller (Wiederholung Folie 20)

Beispiel:

Umsetzung einer einfachen Mehrschichtarchitektur: MVC

Model – View – Controller



Komponentendiagramm

Softwarearchitekturen können in UML durch
Komponentendiagramme veranschaulicht werden:

- ❑ Komponente
= konzeptionelle Einheit eines Softwaresystems, deren Bestandteile gekapselt sind und die Funktionalität über Schnittstellen zur Verfügung stellt.
- ❑ Der Begriff der Komponente ist nur unscharf definiert.
- ❑ Meistens wird unter einer Komponente eine Menge von Klassen verstanden.
- ❑ Eine Hierarchisierung von Komponenten ist möglich.

Komponentendiagramme werden hier nicht eingeführt.

Literatur: Seemann, Jochen; von Gudenberg, Jürgen: Software-Entwurf mit UML 2 – Objektorientierte Modellierung mit Beispielen in Java, S. 121-144
http://link.springer.com/chapter/10.1007/3-540-30950-0_8