Kapitel 3

Objektorientierte Analyse

Prof. Dr. Rolf Hennicker

23.11.2010

Ziele

- Eine Anwendungsfall-Analyse für ein zu entwickelndes System durchführen können.
- Schrittweise ein statisches Modell für ein Anwendungsproblem erstellen können.
- Interaktionsdiagramme für kommunizierende Objekte erstellen können.
- Zustands- und Aktivitätsdiagramme (aus dem Interaktionsmodell) herleiten können.

Die vorgestellte Methode orientiert sich an

- OMT (Rumbaugh et al.)
- "Uses Cases" nach OOSE (Jacobson et al.)

Ziel: Präzise und verständliche Beschreibung der Anforderungen.

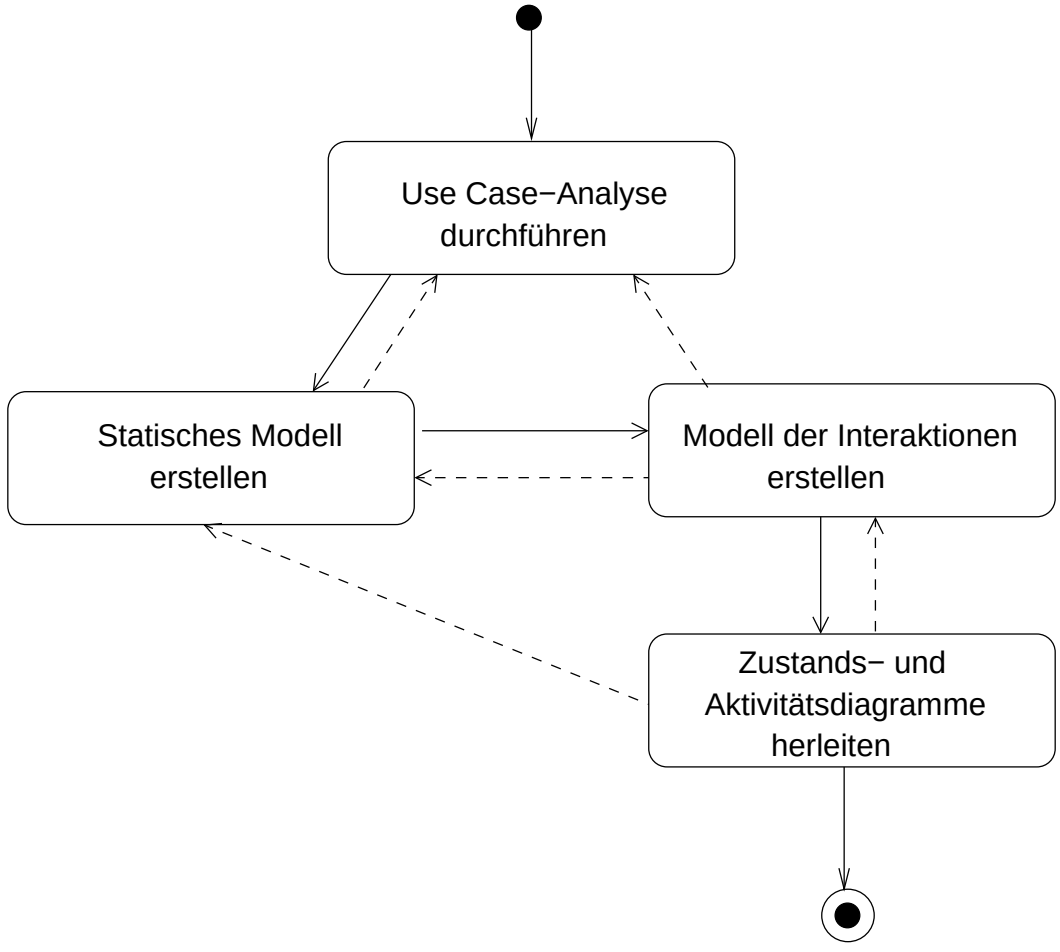
Schritte dazu:

1. "Use Case"-Analyse
2. Entwicklung eines statischen Modells in Form von Klassendiagrammen
3. Entwicklung eines dynamischen Modells in Form von
   (a) Interaktionsdiagrammen
   (b) Zustands- und Aktivitätsdiagrammen

Bemerkungen

- Die obigen Schritte werden ergänzt durch Validieren, Überarbeiten und Erweitern der Modelle (in mehreren Iterationen).
- Während der Analysephase kann zusätzlich
  - eine Grobarchitektur des Systems erstellt werden,
  - die Benutzerschnittstelle skizziert werden.

Prozessmodell der objektorientierten Analyse



3.1 Anwendungsfall-Analyse

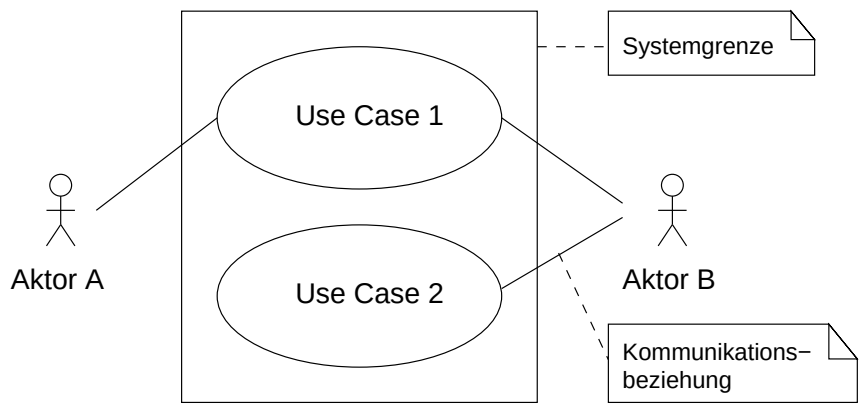
Ausgangspunkt: informelle, möglichst knappe Problembeschreibung

Ziel:
Beschreibung der gewünschten Funktionalität des zu entwickelnden Systems.

3.1.1 Use Case-Modell

- Besteht aus *Aktoren* und *Use Cases* (*Anwendungsfällen*).
- Beschreibt eine externe Sicht auf das System.

Darstellungsform:



Aktoren

- Tauschen von außen Informationen mit dem System aus (Benutzer, andere Systeme, Geräte).
- Aktoren werden durch die *Rolle*, die ein Benutzer gegenüber dem System einnimmt, charakterisiert.

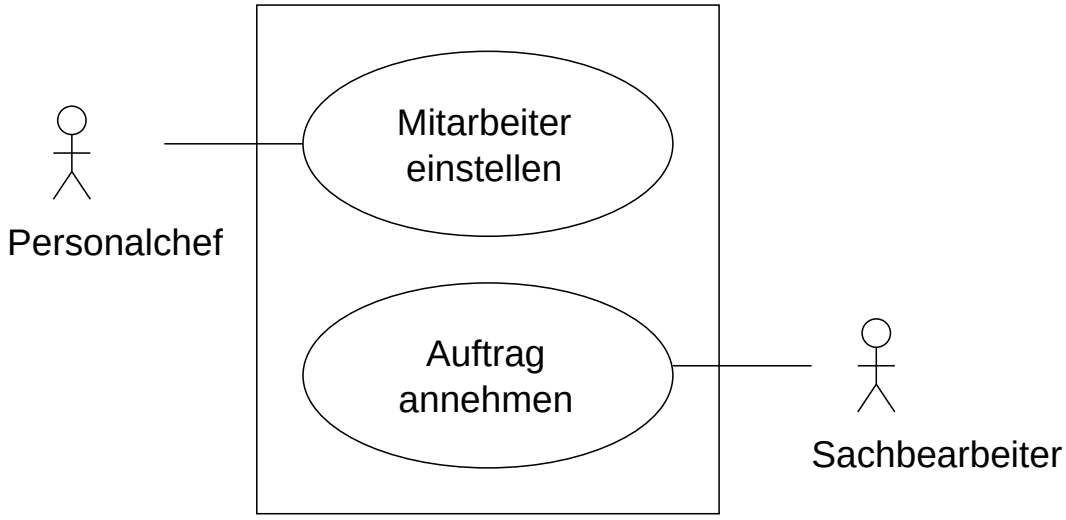
Anwendungsfall

- Beschreibt eine funktionale Anforderung an ein System.
- Beschreibt die Interaktionen zwischen einem (oder mehreren) Aktoren und dem System bei der Bearbeitung einer bestimmten, abgegrenzten Aufgabe.

Definition nach Jacobson:

Ein Anwendungsfall ist eine Menge von verhaltensverwandten Sequenzen von Transaktionen, die durch ein System ausgeführt werden und ein messbares Ergebnis liefern.

Beispiel:



Beachte:

Häufig sind Use Cases die computergestützten Teile von Geschäftsprozessen.

3.1.2 Vorgehensweise bei der Erstellung eines Use Case-Modells

1. Bestimmung der *Aktoren*, die mit dem System interagieren.
   (Wer benützt das System? Wer holt/liefert Informationen von dem/für das System?)

2. Bestimmung der *Anwendungsfälle* aufgrund der Aufgaben, die das System für jeden einzelnen Aktor erledigen soll (z.B. durch Interviews).
   Schwierigkeit: richtige Granularität finden

3. Erstellung eines *Anwendungsfall-Diagramms*, ggf. mit kurzer Beschreibung der Aktoren und Use Cases.

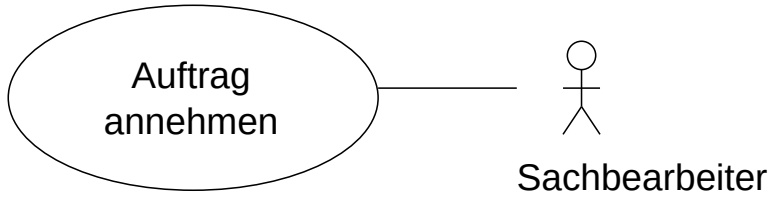
4. Beschreibung der Anwendungsfälle (iterativ).

Eine *Anwendungsfall-Beschreibung* besteht aus:

- Name des Anwendungsfalls
- Kurzbeschreibung
- Vorbedingung
  (Voraussetzung für eine erfolgreiche Ausführung des Anwendungsfalls)
- Nachbedingung
  (Zustand nach erfolgreicher Ausführung)
- einem Standardablauf (Primärszenario)
  (Schritte bzw. Interaktionen, die im Normalfall bei Ausführung des Anwendungsfalls durchlaufen werden)
- mehreren Alternativabläufen (Sekundärszenarien)
  (bei Fehlerfällen und Optionen)

Zusätzlich kann ein Aktivitätsdiagramm für den Anwendungsfall angegeben werden.

Beispiel:



Anwendungsfall: Auftrag annehmen

Kurzbeschreibung:
Ein Sachbearbeiter nimmt für einen Kunden eine Bestellung von Artikeln auf.

Vorbedingung:
Das System ist bereit einen neuen Auftrag anzunehmen.

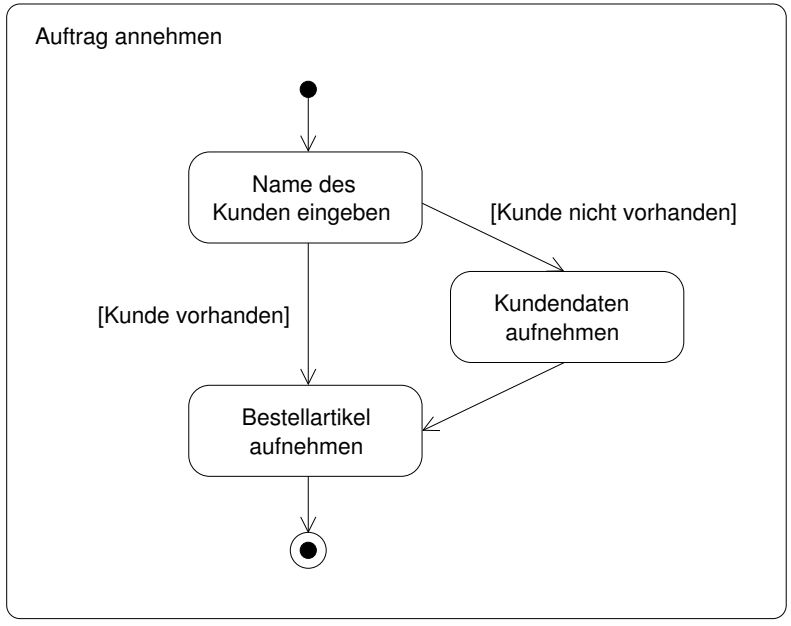
Nachbedingung:
Der Auftrag ist angenommen.

Primärszenario:

1. Der Sachbearbeiter gibt den Namen des Kunden ein.
2. Das System zeigt die Kundendaten an.
3. Der Sachbearbeiter gibt die Daten für die zu bestellenden Artikel ein.

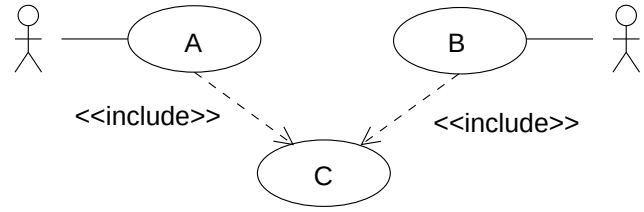
Sekundärszenarien:
Kunde nicht vorhanden

Aktivitätsdiagramm:



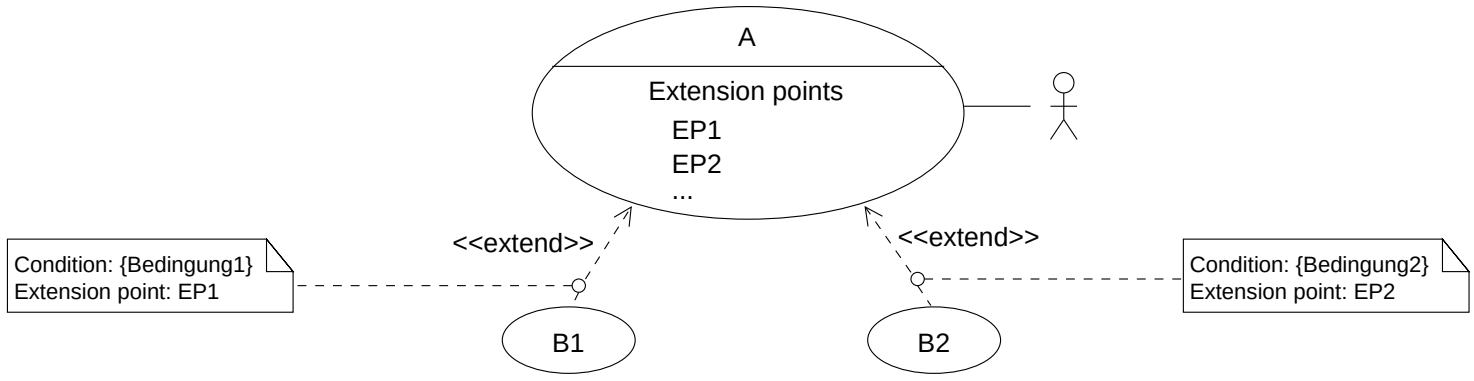
3.1.3 Beziehungen zwischen Anwendungsfällen

1. *Enthält-Beziehung*



Jeder Ablauf von A bzw. B beinhaltet als Teilablauf (notwendigerweise) einen Ablauf von C.

2. *Erweiterungsbeziehung*



Erweitert A um zusätzlich mögliches Verhalten, falls die angegebene Bedingung erfüllt ist.

3.1.4 Beispiel ATM (Automatic Teller Machine)

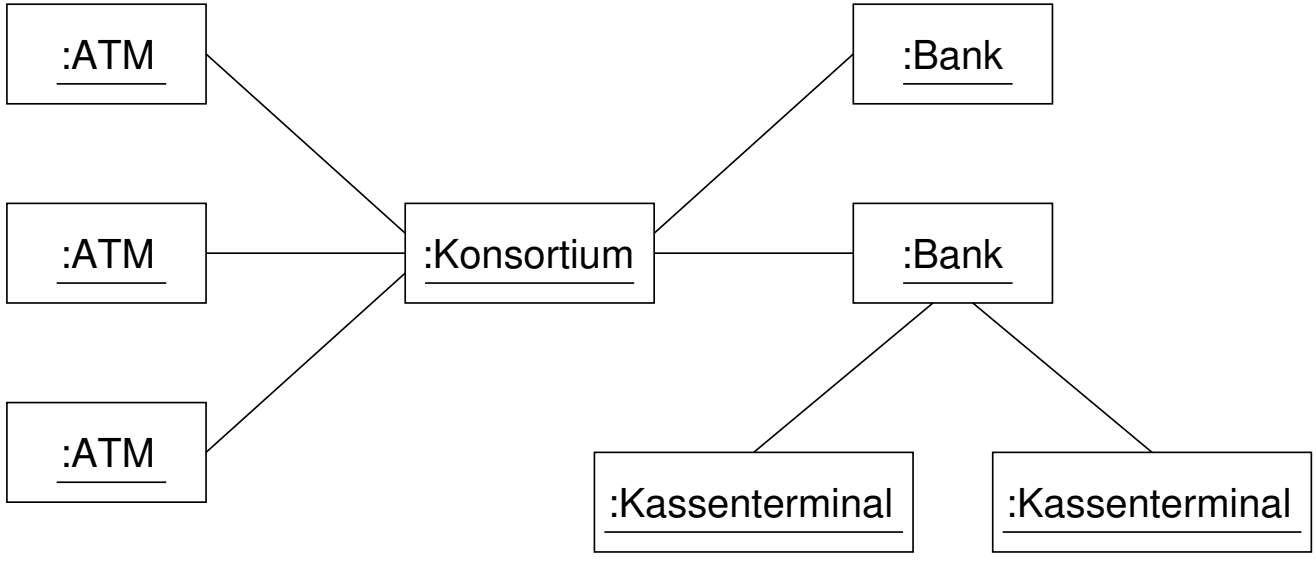
Problembeschreibung: Netzwerk von Bankautomaten
(aus Rumbaugh et al., 1993)

Entwickelt werden soll Software zur Unterstützung eines rechnergesteuerten Bankennetzwerks einschließlich Kassierern und Bankautomaten (ATMs), das sich ein Bankenkonsortium teilt. Jede Bank besitzt einen eigenen Computer, auf dem sie ihre Konten verwaltet und die Transaktionen auf Konten durchführt. Die Banken besitzen Kassenterminals, die direkt mit dem bankeigenen Computer kommunizieren.

Kassierer geben Konto- und Transaktionsdaten ein. ATMs kommunizieren mit einem Zentralrechner, der Transaktionen mit den jeweiligen Banken abklärt. Ein ATM akzeptiert eine Scheckkarte, interagiert mit dem Benutzer, kommuniziert mit dem zentralen System, um die Transaktion auszuführen, gibt Bargeld aus und druckt Belege.

Das System erfordert geeignete Aufzeichnungsmöglichkeiten und Sicherheitsmaßnahmen. Das System muss parallele Zugriffe auf das gleiche Konto korrekt abwickeln. Die Banken stellen die SW für ihre eigenen Computer selbst bereit.

Sie sollen die Software für die ATMs und das Netzwerk entwickeln. Die Kosten des gemeinsamen Systems werden nach Zahl der Scheckkarteninhaber auf die Banken umgelegt.

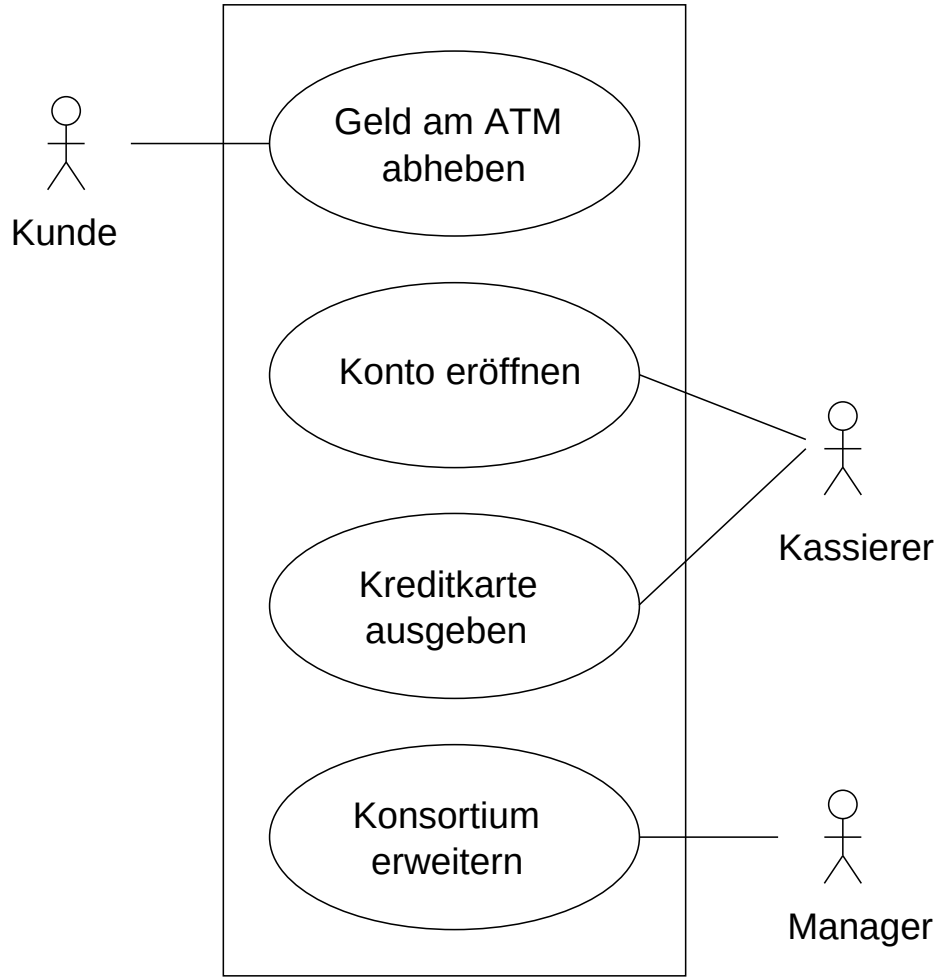


Vorgehensweise

1. *Aktoren:* Kunde, Kassierer, Manager, ...
2. *Use Cases:*
   - Geld am ATM abheben:
     Ein Benutzer hebt am Geldautomaten mit Hilfe seiner Kreditkarte Geld von seinem Konto ab.
   - Konto eröffnen:
     Ein Kassierer richtet ein neues Konto für einen Kunden ein.
   - Neue Kreditkarte ausgeben:
     Ein Kassierer gibt eine neue Kreditkarte für einen Kunden aus.
   - Konsortium erweitern:
     Ein Manager des Konsortiums nimmt eine Bank in das Konsortium auf.

   etc. für weitere Anwendungsfälle wie z.B. "Geld einzahlen", "Überweisung durchführen".

3. *Use Case-Diagramm* (Ausschnitt):



4. *Use Case Beschreibungen:*

Anwendungsfall:
Geld am ATM abheben.

Kurzbeschreibung:
Ein Benutzer hebt am Geldautomaten mit Hilfe seiner Kreditkarte Geld von seinem Konto ab.

Vorbedingung:
Das ATM ist bereit für einen Benutzer eine Transaktion durchzuführen.

Nachbedingung:
Der Benutzer hat die Kreditkarte, das Geld und den Beleg entnommen. Das ATM ist erneut bereit eine Transaktion durchzuführen.

Primärszenario (Standardablauf):

1. Der Kunde gibt seine Kreditkarte ein.
2. Das ATM liest die Kreditkarte und fordert daraufhin die Geheimzahl an.
3. Der Benutzer gibt die Geheimzahl ein.
4. Das ATM liest die Geheimzahl, überprüft sie und lässt dann die BLZ und die Kartennummer beim Konsortium überprüfen.
5. Das Konsortium überprüft die BLZ, gleicht die Kartennummer mit der Bank des Kunden ab und gibt dem ATM sein OK.
6. Das ATM fordert den Benutzer auf die Transaktionsform (Abhebung, Einzahlung, Überweisung, Kontoauszug) zu wählen.
7. Der Benutzer wählt "Abhebung", woraufhin das ATM den Betrag erfragt.
8. Der Benutzer gibt den gewünschten Betrag ein.
9. Das ATM liest den Betrag, überprüft, ob er innerhalb vordefinierter Grenzen liegt, und fordert dann das Konsortium auf die Transaktion zu verarbeiten.
10. Das Konsortium leitet die Anforderung an die Bank weiter, die den Kontostand aktualisiert und die Ausführung bestätigt.
11. Das Konsortium teilt dem ATM den erfolgreichen Abschluss der Transaktion mit.
12. Das ATM gibt den gewünschten Betrag Bargeld aus und fordert den Benutzer auf es zu entnehmen.

13. Der Benutzer entnimmt das Bargeld, woraufhin das ATM fragt, ob der Benutzer eine weitere Transaktion durchführen will.
14. Der Benutzer verneint.
15. Das ATM druckt einen Beleg, gibt die Karte aus und fordert den Benutzer auf sie zu entnehmen.
16. Der Benutzer entnimmt den Beleg und die Karte.
17. Das ATM fordert den nächsten Benutzer auf eine Kreditkarte einzugeben.

Sekundärszenarien (abweichende Fälle):

Karte gesperrt
In Schritt 5, wenn die Bank feststellt, dass die Karte gesperrt ist, teilt sie dies dem Konsortium mit. Das Konsortium leitet die Nachricht an das ATM weiter. Das ATM meldet dem Benutzer den Fehler. Der Use Case wird dann bei Schritt 15 fortgesetzt.

Transaktion gescheitert
In Schritt 10, wenn die Bank feststellt, dass der Kreditrahmen überschritten wird, teilt sie dem Konsortium mit, dass die Banktransaktion gescheitert ist. Das Konsortium leitet die Nachricht an das ATM weiter. Das ATM meldet dem Benutzer das Scheitern der Transaktion. Der Use Case wird ab Schritt 6 wiederholt.

Abbruch durch den Benutzer

Karte nicht lesbar

Falsche Geheimzahl

Falsche Bankleitzahl

Grenzen überschritten

Karte abgelaufen

Kein Geld im ATM

Netzwerk unterbrochen

Zusammenfassung von Abschnitt 3.1

- Ein Use Case-Modell besteht aus Anwendungsfällen und Aktoren.
- Ein Aktor tauscht Informationen mit dem System aus.
- Ein Anwendungsfall beschreibt eine Aufgabe, die das System für einen oder mehrere Aktoren durchführen soll.
- Eine Anwendungsfall-Beschreibung beinhaltet u.a. ein Primärszenario und mehrere Sekundärszenarien.
- Anwendungsfälle können mit <<include>> und <<extend>>-Beziehungen strukturiert werden.
- Ein Use Case-Modell wird schrittweise erstellt.

3.2 Entwicklung eines statischen Modells

Input

- Use Case-Modell
- Problembeschreibung
- Expertenwissen über Anwendungsbereich
- Allgemeinwissen

Ziel: Erstellung eines Klassendiagramms (noch ohne Operationen!)

Vorgehensweise (nach OMT)

1. Klassen identifizieren
2. Assoziationen bestimmen
3. Attribute identifizieren
4. Vererbung einführen
5. Modell überarbeiten

3.2.1 Klassen identifizieren

Kandidaten sind

- Personen bzw. Rollen (z.B. Student, Angestellter, Kunde, ...)
- Organisationen (z.B. Firma, Abteilung, Uni, ...)
- Gegenstände (z.B. Artikel, Flugzeug, Immobilie, ...)
- begriffliche Konzepte (z.B. Bestellung, Vertrag, Vorlesung, ...)

1. Schritt: Kandidaten notieren

Möglichkeit: Durchsuche die Use Case-Beschreibungen nach Substantiven.

2. Schritt: Ungeeignete Klassen streichen

Streichungskriterien:

- redundante Klassen
- irrelevante Klassen
- vage Klassen
- Attribute oder Attributwerte
- Operationen (Tätigkeiten)

3. Schritt: Fehlende relevante Klassen hinzunehmen

Hinweise für fehlende Klassen:

- Attribute aus Schritt 2, zu denen es bisher noch keine Klassen gibt
- Problembereichswissen

Beispiel ATM:

1. Schritt:

Substantive im (Primärszenario des) Use Case “Geld am ATM abheben” notieren

Kunde

Kreditkarte

ATM

Geheimzahl

Benutzer

Bankleitzahl

Kartennummer

Konsortium

Bank

Transaktionsform

Abhebung, Einzahlung, Überweisung, Kontoauszug

Betrag

Grenzen

Transaktion

Abschluss

Ausführung

Anforderung

Kontostand

Bargeld

Beleg

Karte

2. Schritt: Ungeeignete Klassen streichen

Kunde

Kreditkarte

ATM

Geheimzahl *(Attribut)*

Benutzer *(redundant)*

Bankleitzahl *(Attribut)*

Kartennummer *(Attribut)*

Konsortium

Bank

Transaktionsform *(Attribut)*

Betrag *(Attribut)*

Grenzen *(Attribut)*

Transaktion

Abschluss *(Tätigkeit)*

Ausführung *(Tätigkeit)*

Anforderung *(Tätigkeit)*

Kontostand *(Attribut)*

Bargeld *(irrelevant)*

Beleg *(vage)*

Karte *(redundant)*

Abhebung, Einzahlung, Überweisung, Kontoauszug *(Attributwert von Transaktionsform)*

3. Schritt: Fehlende Klassen hinzunehmen

- Konto (*folgt aus dem Attribut Kontostand*)
- Kassierer
- Kassenterminal
- Kassiertransaktion
- Aussentransaktion (*statt der o.g. Transaktion*)

3.2.2 Assoziationen identifizieren

Kandidaten sind physische oder logische Verbindungen mit einer bestimmten Dauer, wie

- konzeptionelle Verbindungen (arbeitet für, ist Kunde von, ...)
- Besitz (hat, ist Teil von, ...)
- (häufige) Zusammenarbeit von Objekten zur Lösung einer Aufgabe

1. Schritt: Kandidaten notieren

Möglichkeit: Überprüfe

- Verben
- Substantive im Genitiv (z.B. die Kreditkarte des Kunden)
- Possesivpronomen (z.B. seine Kreditkarte)

Beachte:

- Verben bezeichnen häufig Aktivitäten und keine Beziehungen (z.B. ATM überprüft die Geheimzahl).
- Falls jedoch in einer Aktivität mehrere Objekte gebraucht werden, kann dies ein Hinweis auf eine häufige Zusammenarbeit sein, die eine Assoziation voraussetzt (z.B. ATM lässt Bankleitzahl beim Konsortium überprüfen).
- Assoziationen sind i.a. schwieriger zu bestimmen als Klassen. Wissen über den Problembereich (und Allgemeinwissen) ist wichtig!

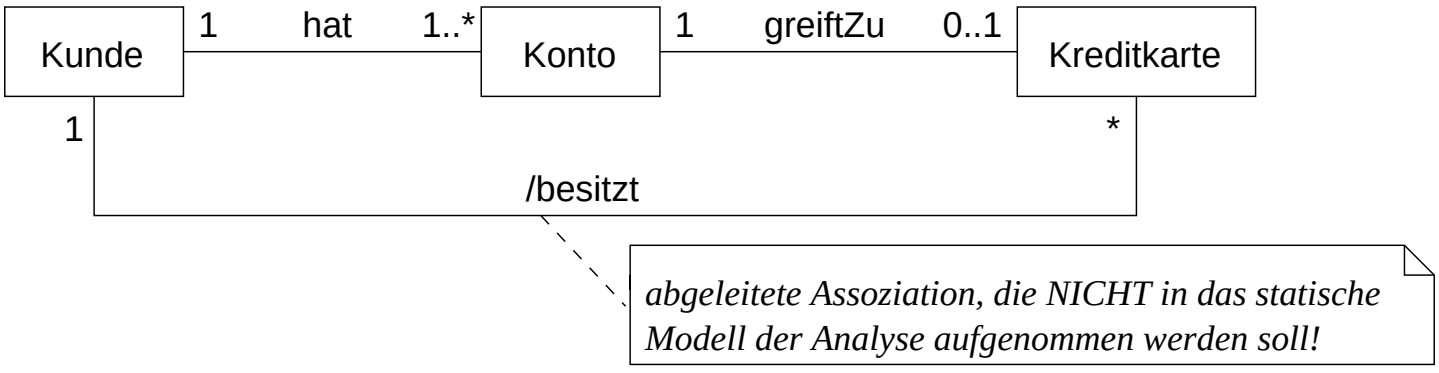
Beispiel ATM:

- Kunde besitzt Kreditkarte (... seine Kreditkarte ...)
- Konsortium besitzt ATM (... lässt überprüfen ...)
- Konsortium besteht aus Banken (... gleicht ab ...)
- Bank führt Konten (Wissen über den Problembereich)
- Kunde hat Konten (Wissen über den Problembereich)

2. Schritt: Ungeeignete Assoziationen streichen

Kriterien wie bei Klassen, inbesondere auf redundante (abgeleitete) Assoziationen möglichst verzichten.

Beispiel ATM:



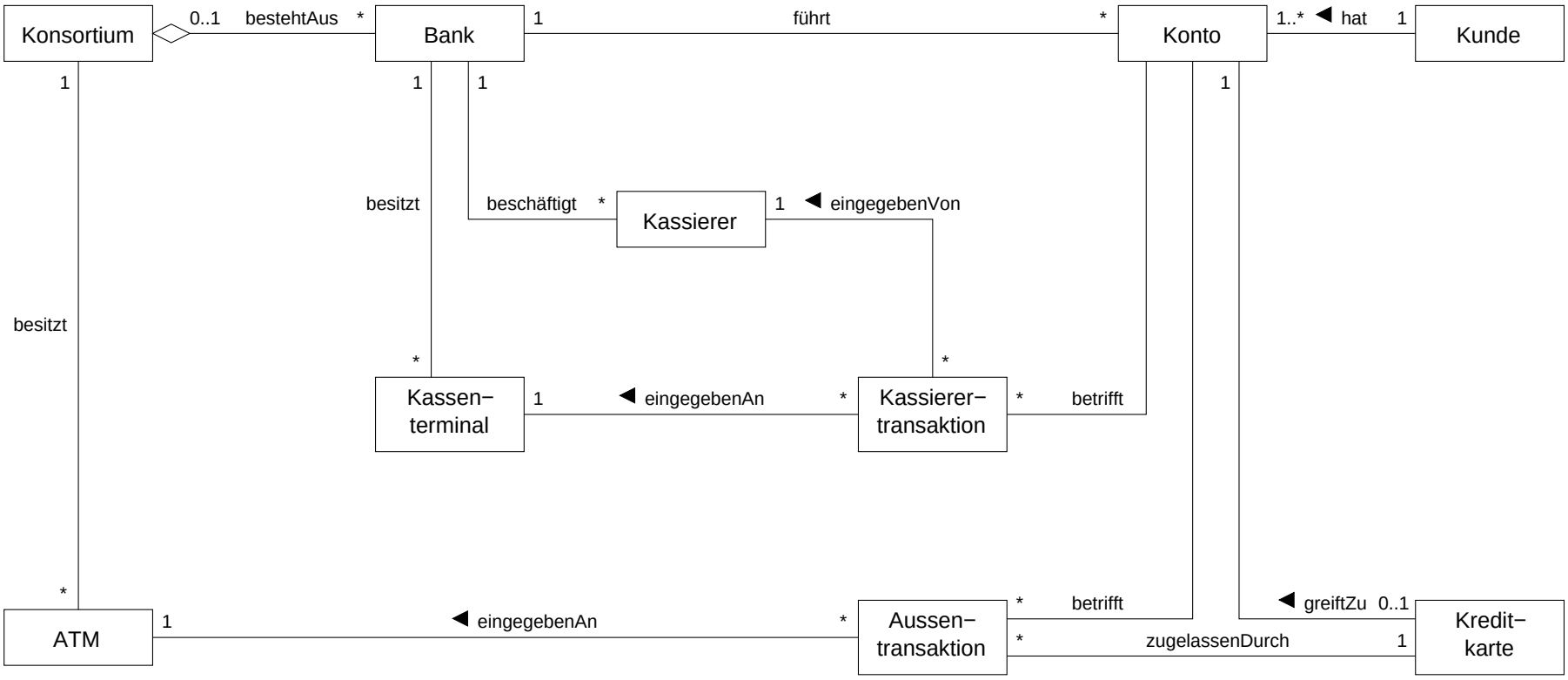
3. Schritt: Fehlende relevante Assoziationen hinzunehmen

4. Schritt: Multiplizitäten und ggf. explizite Rollennamen hinzufügen

Bemerkung

Multiplizitäten können in zweifelhaften Fällen noch weggelassen werden und erst bei der Überarbeitung des Modells bestimmt werden.

Beispiel ATM: Erstes Klassendiagramm



3.2.3 Attribute identifizieren

Richtlinien

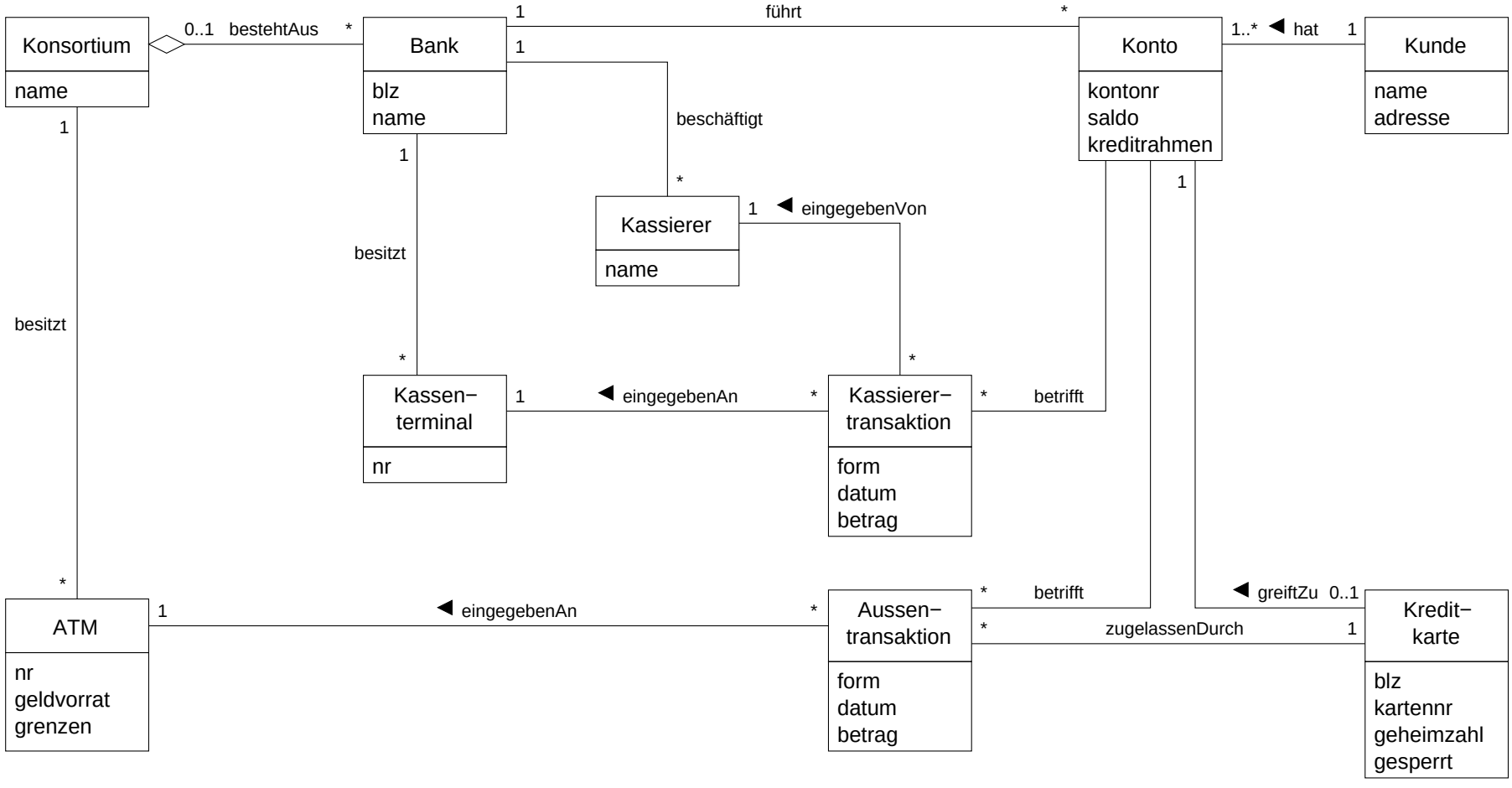
- nur Attribute, die für die Anwendung relevant sind (Problembereichswissen wichtig!)
- keine Attribute, deren Werte Objekte sind (dafür Assoziationen verwenden!)
- Attributtypen können noch weggelassen werden

Beispiel ATM:

Aus dem Primärszenario des Use Case "Geld am ATM abheben" ergeben sich die folgenden Attribute:

- blz ist Attribut von Bank und von Kreditkarte
- kartennr, geheimzahl sind Attribute von Kreditkarte
- form, betrag sind Attribute von Kassierertransaktion und Aussentransaktion
- grenzen ist Attribut von ATM
- saldo (Kontostand) ist Attribut von Konto

Beispiel ATM: Klassendiagramm mit Attributen



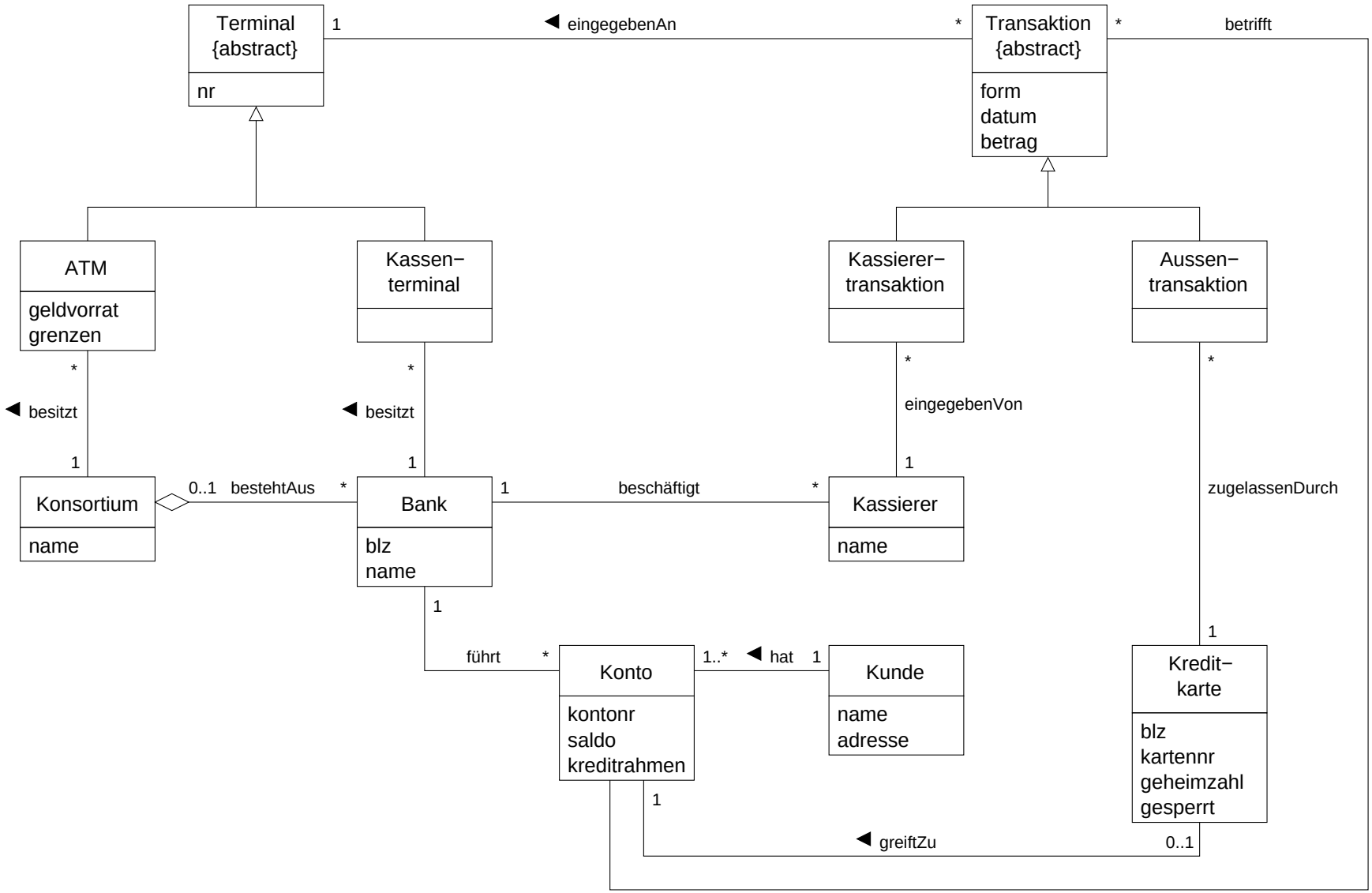
3.2.4 Vererbung einführen

- Vor allem **Generalisierung**!
- Zusammenfassen gemeinsamer Merkmale vorhandener Klassen (Attribute, Assoziationen) in einer Oberklasse.
- Bestimmen, welche Klassen abstrakt sind.

Beispiel ATM:

- Aussentransaktion und Kassierertransaktion haben alle Attribute gemeinsam und eine gemeinsame Assoziation zu Konto
  $\Rightarrow$ Generalisierung zur (abstrakten) Klasse Transaktion
- ATM und Kassenterminal können zur (abstrakten) Klasse Terminal generalisiert werden.

Beispiel ATM: Klassendiagramm mit Vererbung



3.2.5 Modell überarbeiten

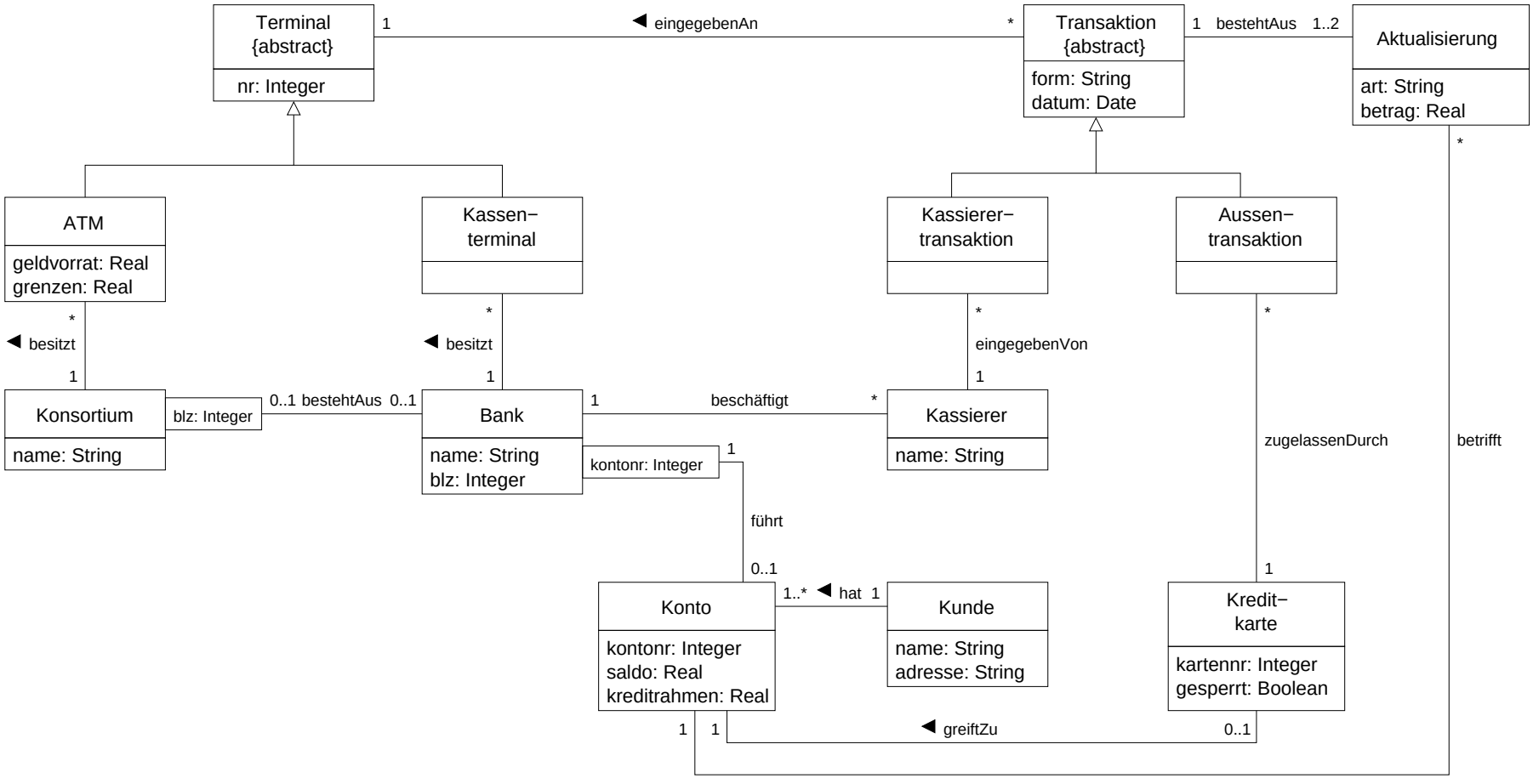
Fragen

- Fehlende Klassen oder Assoziationen?
- Unnötige Klassen oder Assoziationen?
- Falsche Platzierung von Assoziationen und Attributen?
- Qualifizierer für Assoziationen?
- Typen für Attribute?
- Fehlende Multiplizitäten?

Beispiel ATM:

- Eine Überweisungstransaktion betrifft zwei Konten
  $\Rightarrow$ Einführung der Klasse "Aktualisierung".
- Eine Kreditkarte ist ein Stück Plastik! In dem zugehörigen Software-Objekt ist die Geheimzahl *nicht* gespeichert. Ebensowenig die Bankleitzahl (wäre redundant).
- Qualifizierer für die Assoziationen zwischen Konsortium und Bank und zwischen Bank und Konto verwenden.

Beispiel ATM: Klassendiagramm nach Überarbeitung



Zusammenfassung von Abschnitt 3.2

- Das statische Modell beschreibt die strukturellen und datenorientierten Aspekte eines Systems.
- Das statische Modell wird durch Klassendiagramme (ggf. ergänzt um Objektdiagramme) dargestellt.
- Schritte bei der Entwicklung des statischen Modells sind:
  - Klassen identifizieren
  - Assoziationen bestimmen
  - Attribute identifizieren
  - Vererbung einführen
  - Modell überarbeiten

3.3 Modellierung von Interaktionen

Interaktion = spezifisches Muster der Zusammenarbeit und des Nachrichtenaustauschs zwischen Objekten zur Erledigung einer bestimmten Aufgabe (z.B. eines Anwendungsfalls).

Ziel:

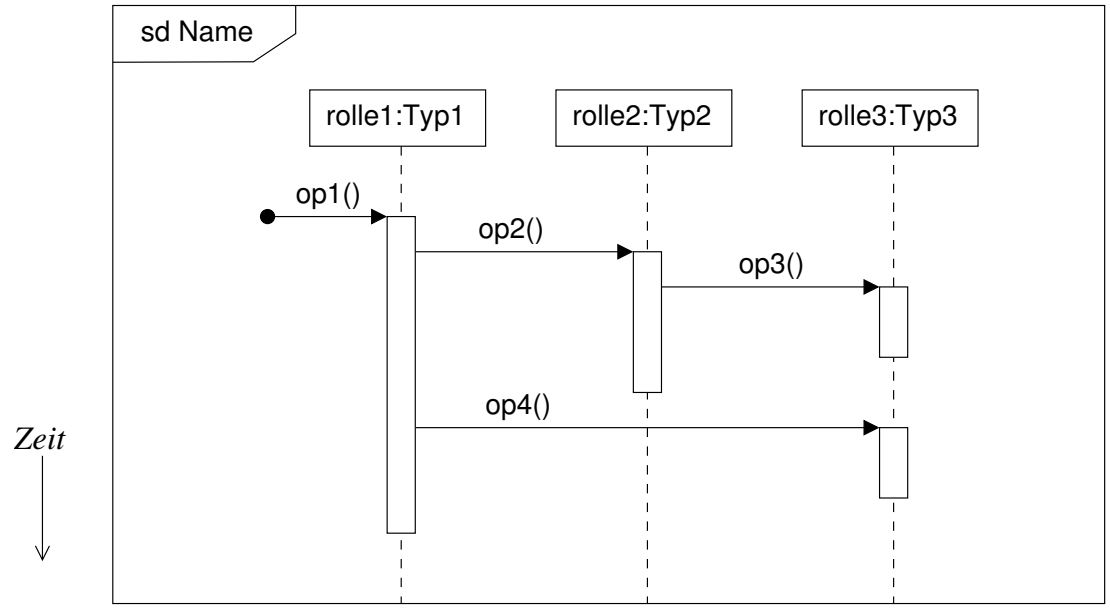
Entwurf einer Menge von *Interaktionsdiagrammen* für jeden Anwendungsfall.

Man unterscheidet zwei Arten von Interaktionsdiagrammen:

- Sequenzdiagramme
- Kommunikationsdiagramme

3.3.1 Sequenzdiagramme

Heben die *zeitliche Reihenfolge* hervor, in der Nachrichten zwischen Objekten ausgetauscht (d.h. gesendet und empfangen) werden.



Beachte:

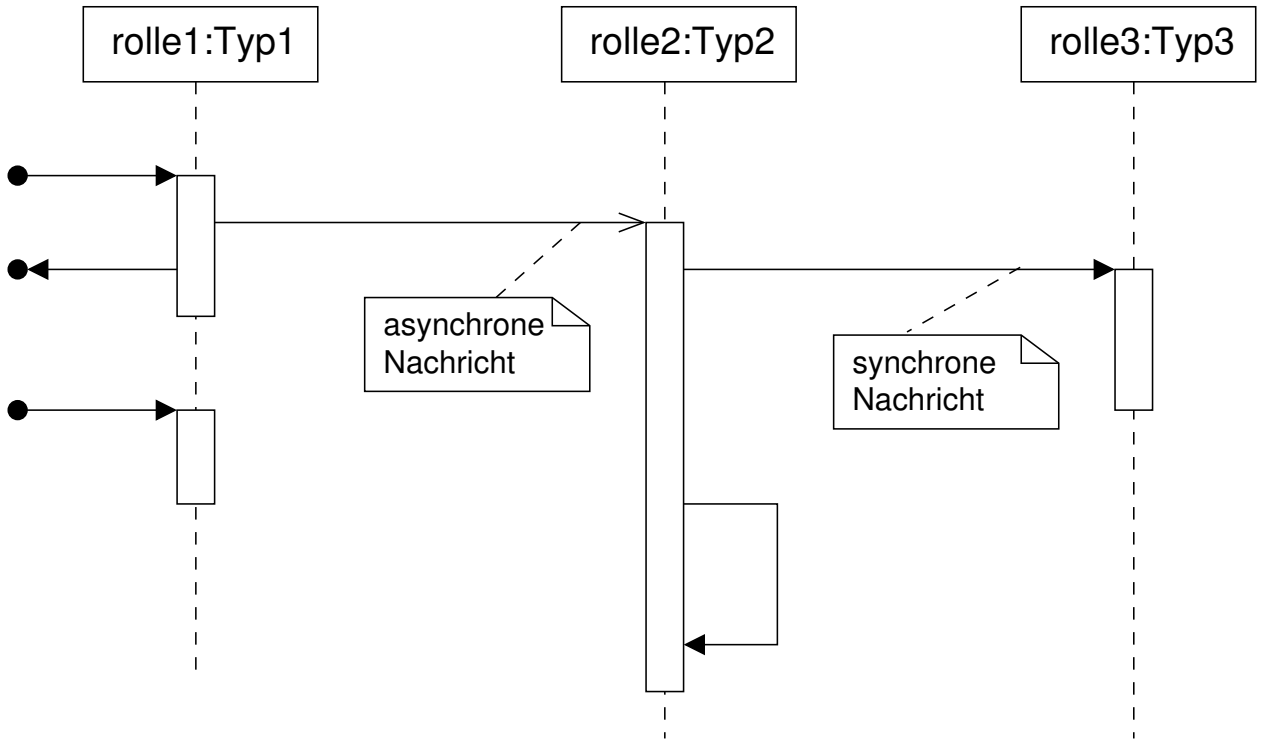
Empfangen einer Nachricht ist ein **Ereignis**.
Senden einer Nachricht ist eine **Aktion**.

Aktivierung

Zeitspanne, innerhalb der ein Objekt aktiv ist, weil es

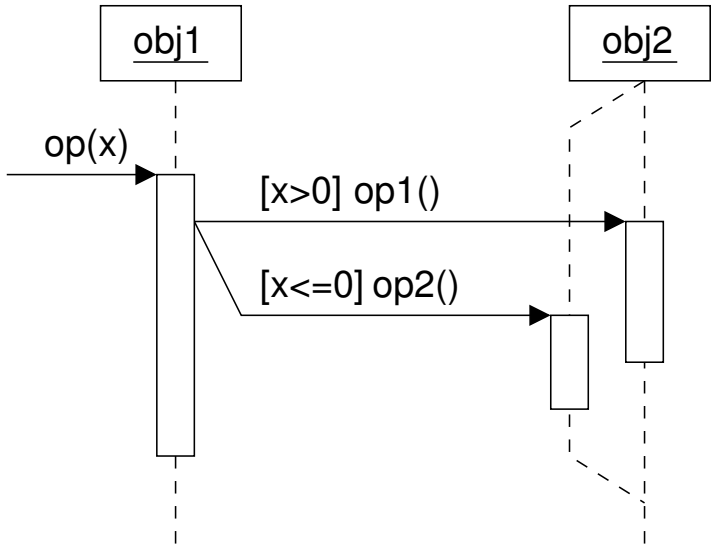
- gerade selbst tätig ist, oder
- auf die Beendigung einer Aktivierung eines (anderen) Objekts wartet, dem es eine (synchrone) Nachricht gesendet hat ("Rückgabe des Steuerungsfokus")

Asynchrone Nachrichten und parallele Ausführung

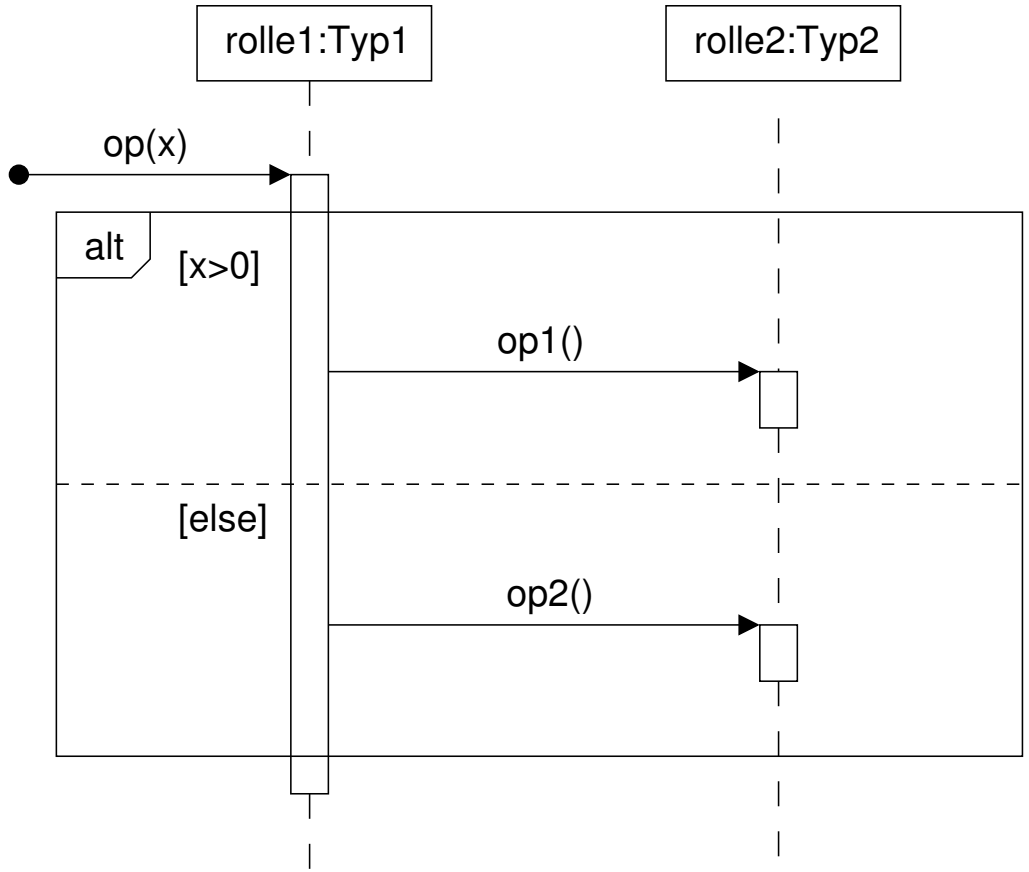


Das sendende Objekt setzt sofort nach dem Senden einer asynchronen Nachricht seine Tätigkeit fort (parallel zur Tätigkeit des Empfängers).

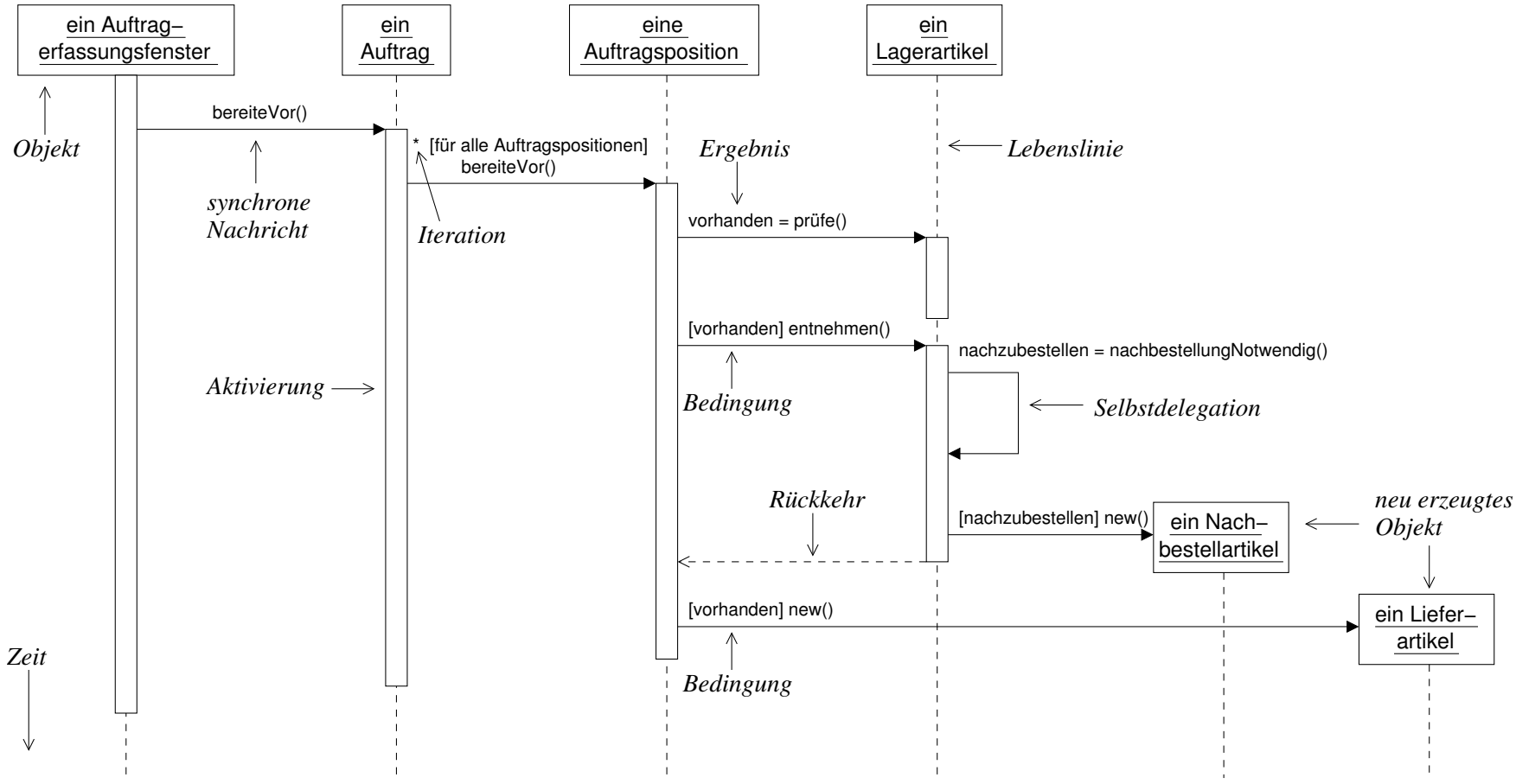
Alternativen (UML 1.X)



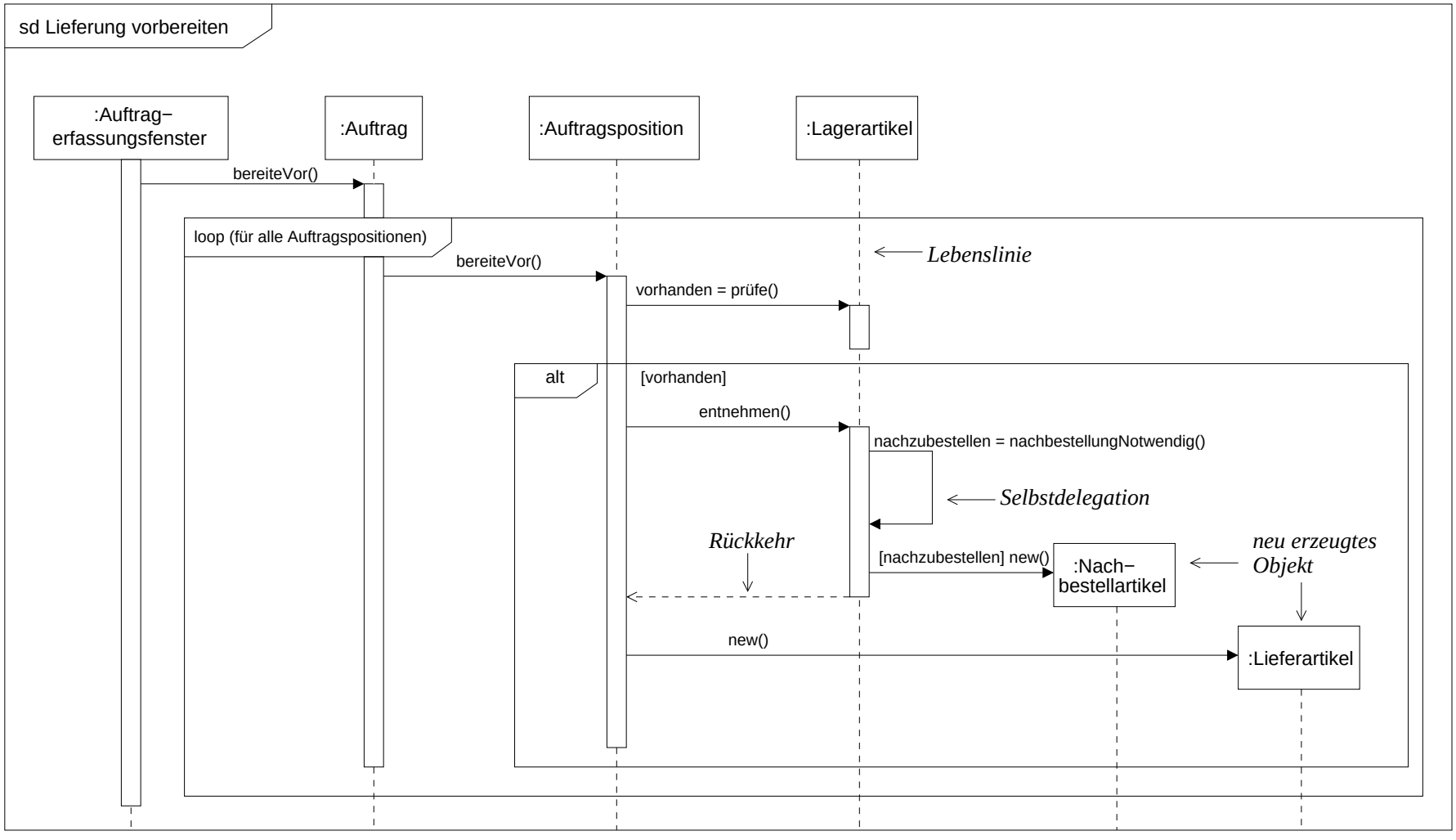
Alternativen (UML 2.0)



Beispiel (UML 1.X): Sequenzdiagramm für "Lieferung vorbereiten"

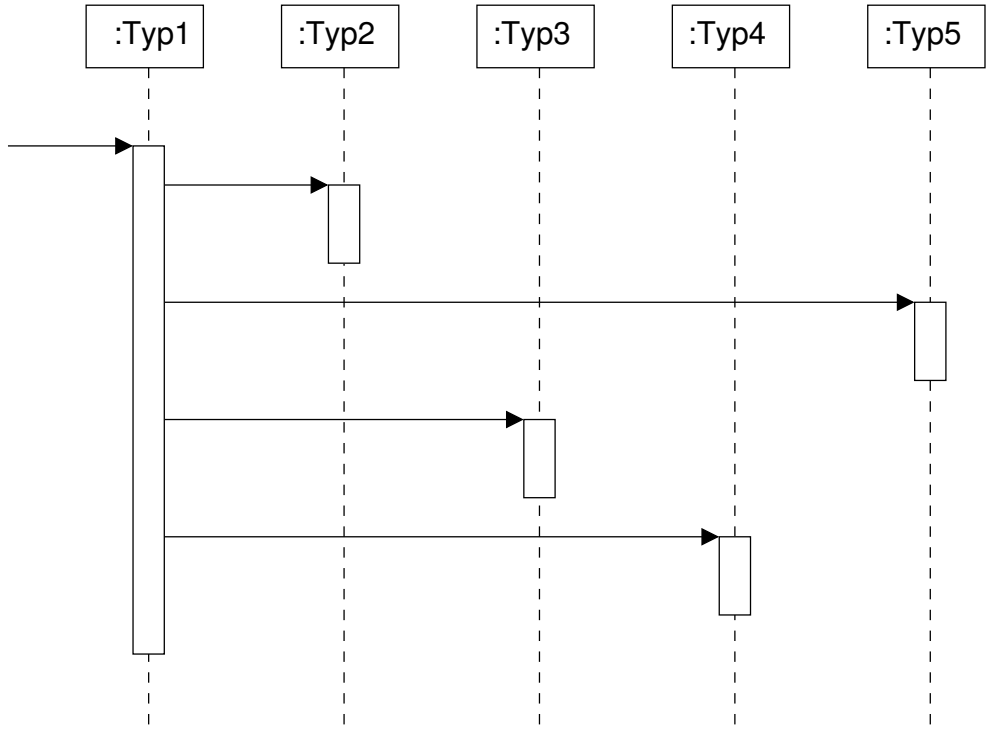


UML 2.0


sd Lieferung vorbereiten
:Auftrag−
erfassungsfenster
:Auftrag
:Auftragsposition
:Lagerartikel
bereiteVor()
loop (für alle Auftragspositionen)
bereiteVor()
Lebenslinie
vorhanden = prüfe()
alt
[vorhanden]
entnehmen()
nachzubestellen = nachbestellungNotwendig()
Selbstdelegation
Rückkehr
[nachzubestellen] new()
:Nach−
bestellartikel
neu erzeugtes
Objekt
new()
:Lieferartikel

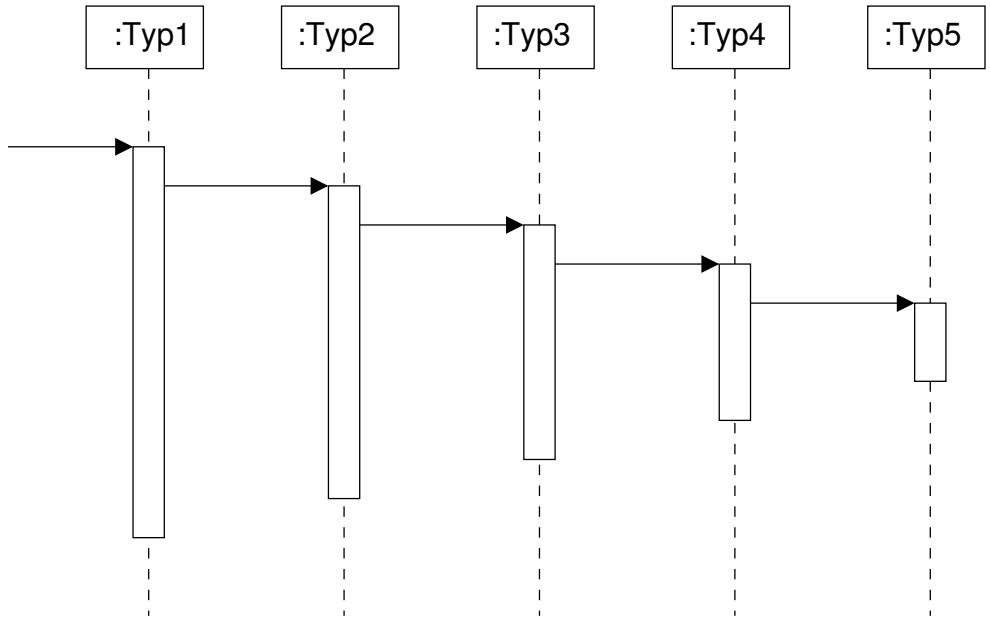
Typische Strukturen von Sequenzdiagrammen

Zentralisierte Struktur ("Fork")



Ein Objekt kontrolliert die anderen Objekte (besitzt die Verantwortung für die erfolgreiche Ausführung).

Dezentralisierte Struktur ("Stair")

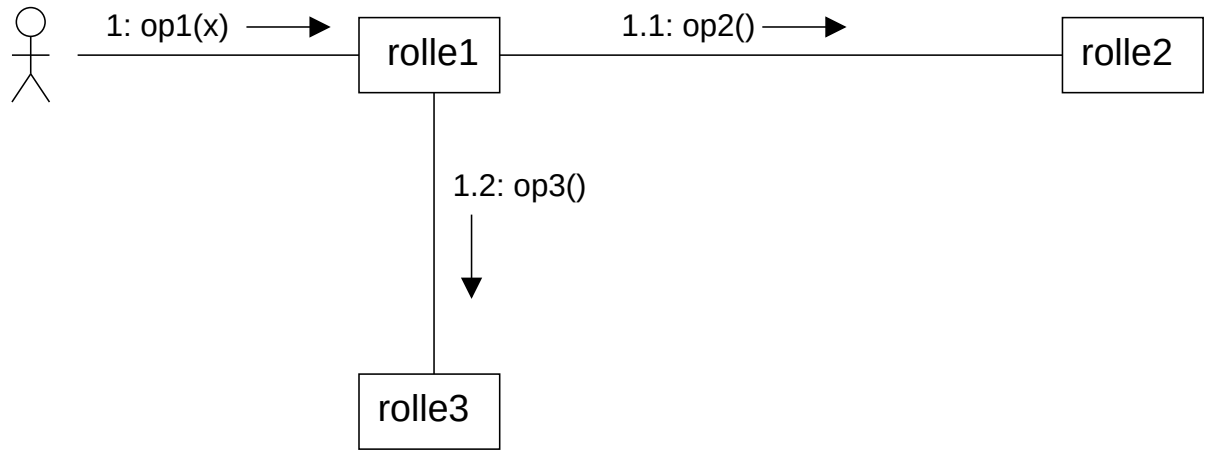


Jedes Objekt trägt eine eigene Verantwortung für die erfolgreiche Ausführung.

3.3.2 Kommunikationsdiagramme

Heben die *strukturellen Beziehungen* (dauerhafte und temporäre) aller an einer Interaktion beteiligten Objekte hervor.

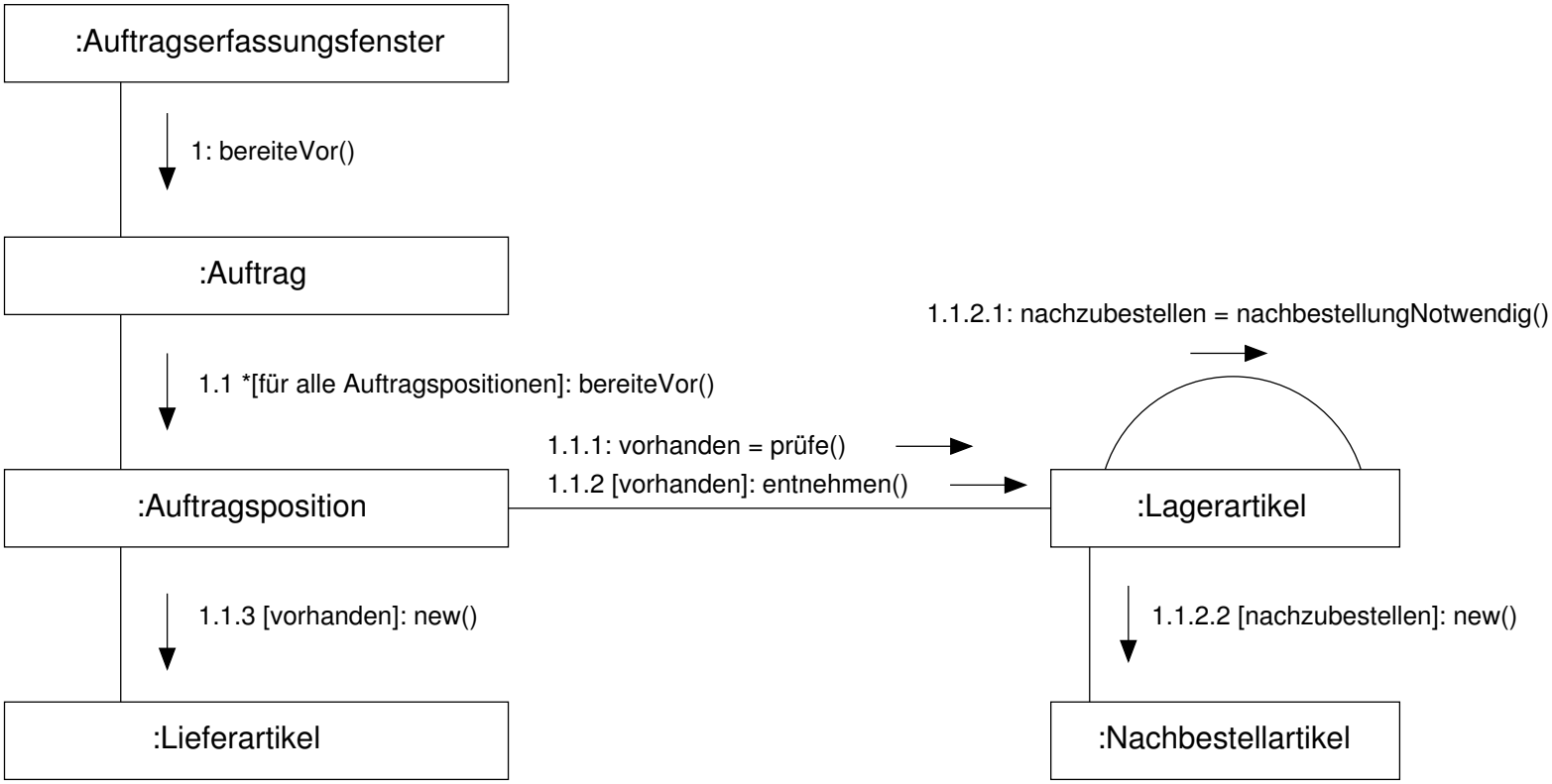
Die zeitliche Reihenfolge wird durch Nummerierung der Nachrichten dargestellt (mit dezimalen Nummern 1.1, 1.2 etc. für geschachtelte Nachrichten).



Bemerkung

Die Links in Kommunikationsdiagrammen sind Instanzen von Assoziationen oder temporäre Beziehungen.

Beispiel: Kommunikationsdiagramm für "Lieferung vorbereiten"



Bemerkung

Sequenz- und Kommunikationsdiagramme stellen im wesentlichen dieselbe Information in verschiedener Form dar.

3.3.3 Entwurf von Interaktionsdiagrammen

Input

- Use Case-Beschreibungen (Szenarien!)
- statisches Modell

Ziel

Modellierung der Zusammenarbeit von Objekten innerhalb eines Anwendungsfalls durch Interaktionsdiagramme.

Vorgehensweise

1. Identifiziere die Nachrichten, die innerhalb eines Anwendungsfalls ausgetauscht werden und die Objekte, die die Nachrichten senden und empfangen.
2. Konstruiere Interaktionsdiagramme für jeden Anwendungsfall.

Möglichkeiten dazu:

(A) Für jedes Szenario eines Use Case ein eigenes Interaktionsdiagramm.

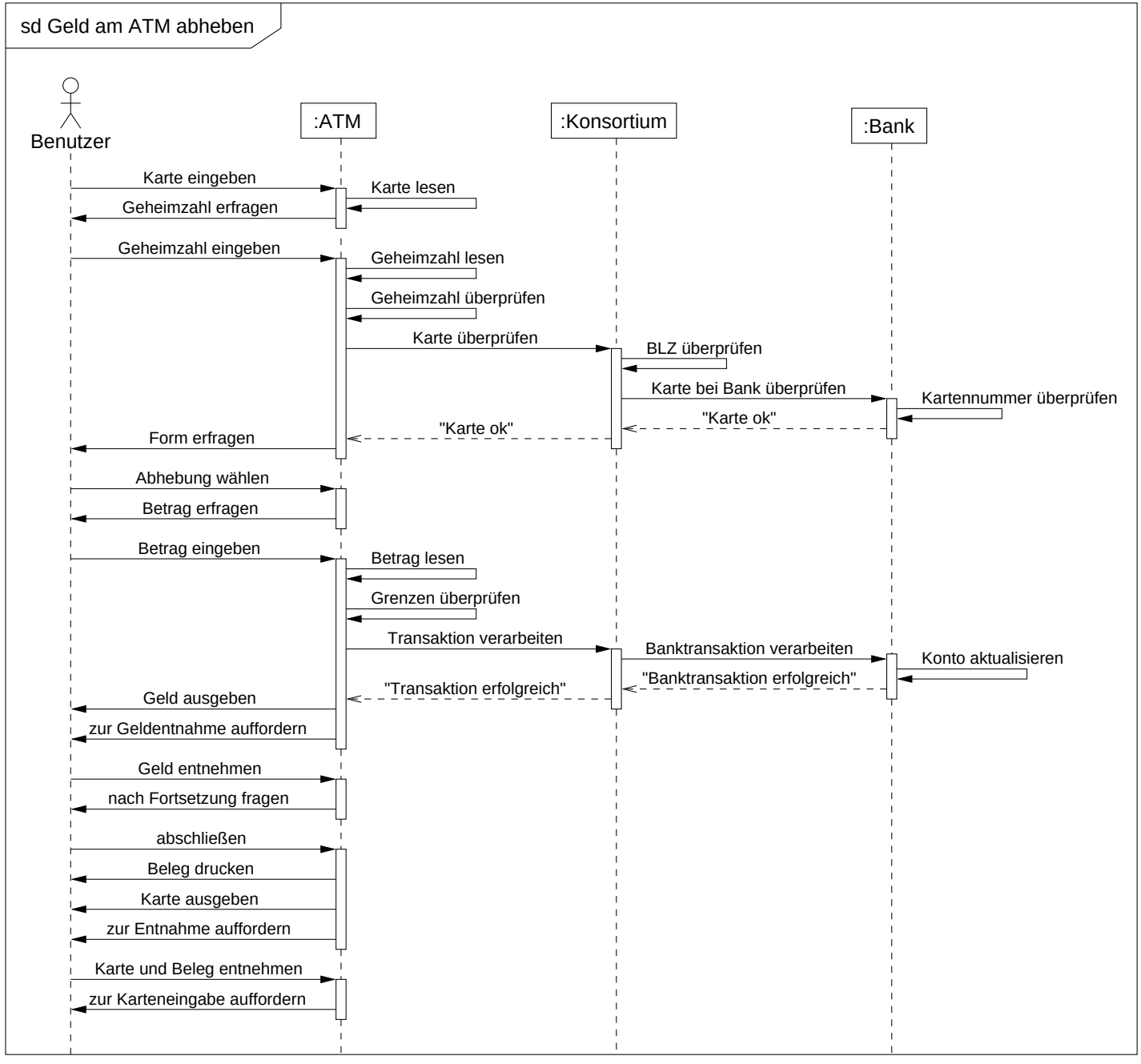
(B) Ein komplexes Interaktionsdiagramm (mit Alternativen, Iterationen (loops), etc.), das alle Szenarien eines Use Case subsummiert.
Nachteil: Wird schnell unübersichtlich!

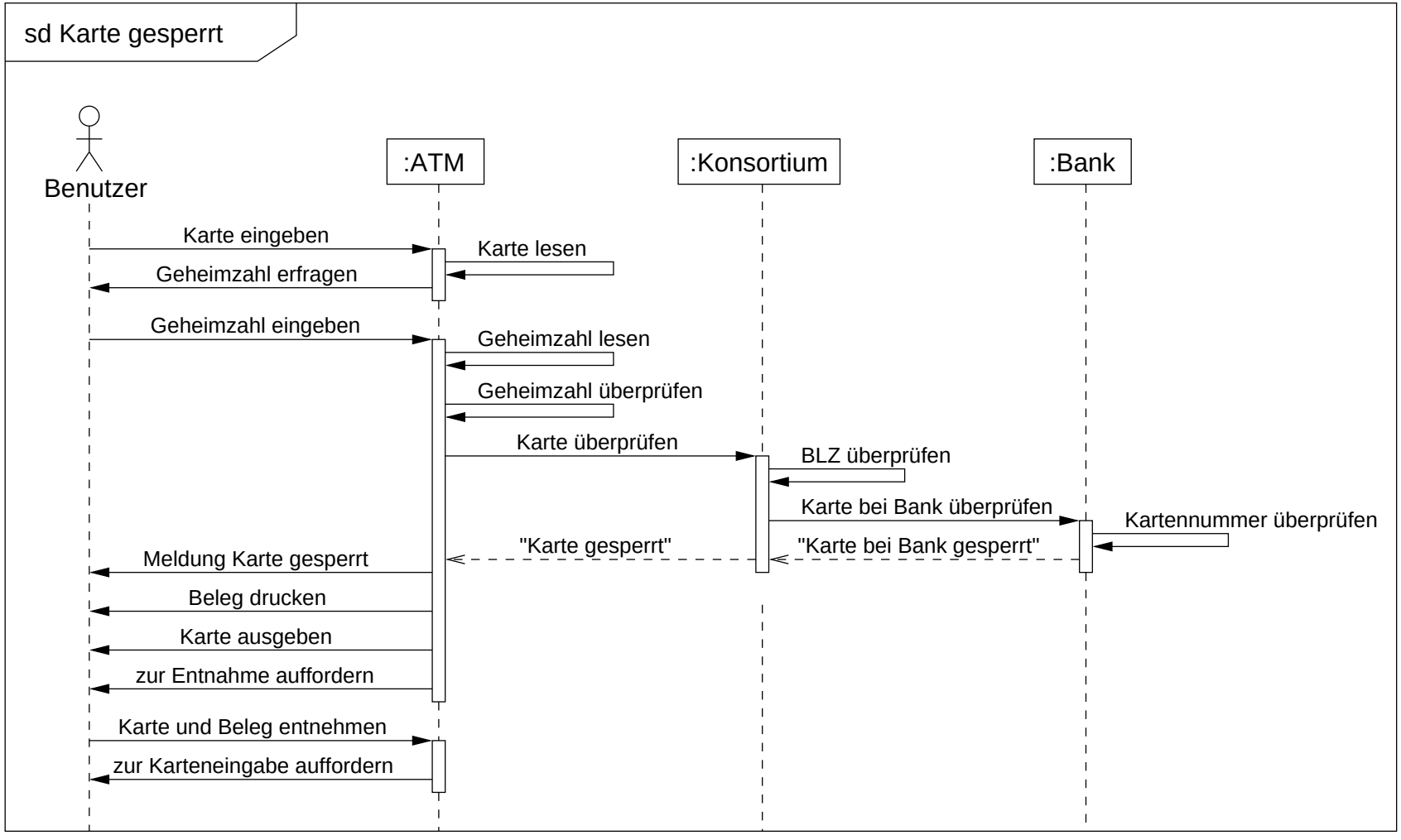
Wir orientieren uns an Möglichkeit (A) und verwenden Sequenzdiagramme (SDs).

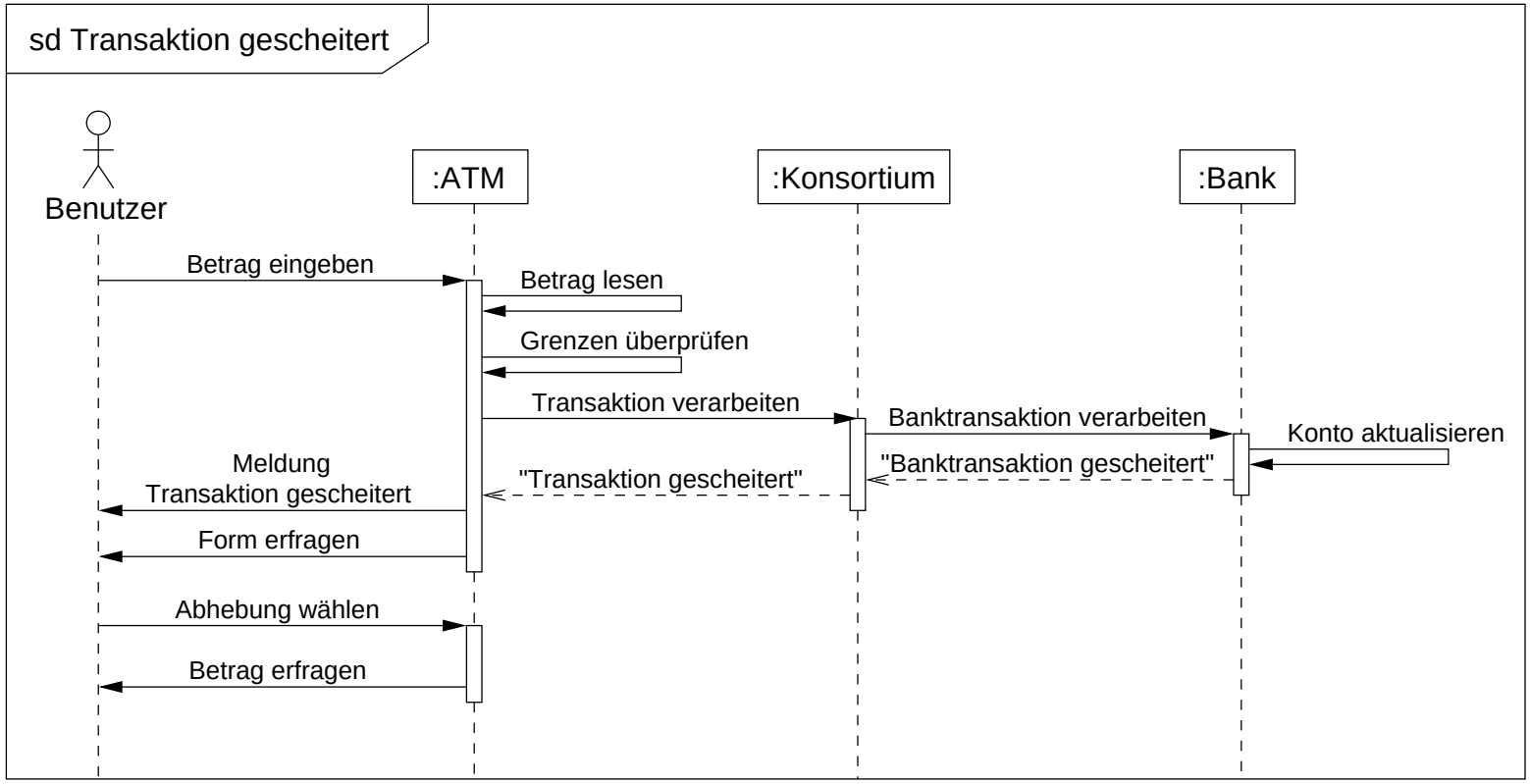
Beispiel: Use Case "Geld am ATM abheben"

Wir konstruieren:

- ein SD für das Primärszenario
- ein SD für das Sekundärszenario "Karte gesperrt"
- ein SD für das Sekundärszenario "Transaktion gescheitert"


sd Geld am ATM abheben
Benutzer
:ATM
:Konsortium
:Bank
Karte eingeben
Karte lesen
Geheimzahl erfragen
Geheimzahl eingeben
Geheimzahl lesen
Geheimzahl überprüfen
Karte überprüfen
BLZ überprüfen
Karte bei Bank überprüfen
Kartennummer überprüfen
"Karte ok"
"Karte ok"
Form erfragen
Abhebung wählen
Betrag erfragen
Betrag eingeben
Betrag lesen
Grenzen überprüfen
Transaktion verarbeiten
Banktransaktion verarbeiten
Konto aktualisieren
"Banktransaktion erfolgreich"
"Transaktion erfolgreich"
Geld ausgeben
zur Geldentnahme auffordern
Geld entnehmen
nach Fortsetzung fragen
abschließen
Beleg drucken
Karte ausgeben
zur Entnahme auffordern
Karte und Beleg entnehmen
zur Karteneingabe auffordern


sd Karte gesperrt
Benutzer
:ATM
:Konsortium
:Bank
Karte eingeben
Karte lesen
Geheimzahl erfragen
Geheimzahl eingeben
Geheimzahl lesen
Geheimzahl überprüfen
Karte überprüfen
BLZ überprüfen
Karte bei Bank überprüfen
Kartennummer überprüfen
"Karte gesperrt"
"Karte bei Bank gesperrt"
Meldung Karte gesperrt
Beleg drucken
Karte ausgeben
zur Entnahme auffordern
Karte und Beleg entnehmen
zur Karteneingabe auffordern


sd Transaktion gescheitert
Benutzer
:ATM
:Konsortium
:Bank
Betrag eingeben
Betrag lesen
Grenzen überprüfen
Transaktion verarbeiten
Banktransaktion verarbeiten
Konto aktualisieren
Meldung
Transaktion gescheitert
"Transaktion gescheitert"
"Banktransaktion gescheitert"
Form erfragen
Abhebung wählen
Betrag erfragen

Zusammenfassung von Abschnitt 3.3

- Interaktionsdiagramme beschreiben die Zusammenarbeit und den Nachrichtenaustausch zwischen *mehreren* Objekten.
- Wir unterscheiden Sequenzdiagramme (heben die zeitliche Reihenfolge hervor) und Kommunikationsdiagramme (heben die strukturellen Beziehungen hervor).
- Ein Modell der Interaktionen basiert auf den Anwendungsfall-Beschreibungen und dem statischen Modell.
- Pro Anwendungsfall werden i.a. mehrere Interaktionsdiagramme erstellt (z.B. für das Primärszenario und für jedes Sekundärszenario ein Sequenzdiagramm.)

3.4 Entwicklung von Zustands- und Aktivitätsdiagrammen

Ausgangspunkt: Menge von Sequenzdiagrammen (SDs) zu den Use Cases.

Ziel

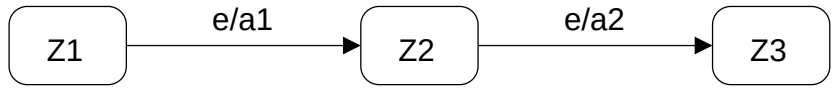
- Je ein Zustandsdiagramm für jede Klasse mit "interessantem Verhalten" (d.h. die Objekte der Klasse haben einen nicht-trivialen Lebenszyklus).
- Aktivitätsdiagramme zur Beschreibung der Abläufe von Operationen.

Bemerkung

- Zustands- und Aktivitätsdiagramme erfassen das (vollständige) Verhalten *eines jeden* Objekts einer Klasse über viele Szenarien hinweg.
- Auf Aktivitätsdiagramme kann verzichtet werden, wenn das Ablauf-Verhalten einer Operation schon vollständig in *einem* Interaktionsdiagramm beschrieben ist.

Kriterien für "interessantes Verhalten" eines Objekts

- Es gibt mindestens ein Ereignis, das in Abhängigkeit vom Objektzustand *unterschiedliche Reaktionen* auslösen kann.



Beispiel: Stellen einer Digitaluhr

- Mindestens ein Ereignis wird in bestimmten Zuständen ignoriert (bzw. kann in bestimmten Zuständen nicht auftreten).

Beispiel: ATM

Typische zustandsabhängige Objekte sind:

- Kontrollobjekte für Anwendungsfälle und Benutzerschnittstellen
- Geräte (z.B. Videorecorder, Digitaluhr, Thermostat, ...)
- Objekte mit einem beschränkten Fassungsvermögen (voll, leer, ...)

Klassifizierung von Zuständen

- Einen Zustand, dem eine Aktivität zugeordnet ist und der **nur** durch ein (evtl. bedingtes) Completion Event verlassen werden kann, nennen wir *Aktivitätszustand*.
- Einen Zustand, dem keine Aktivität zugeordnet ist, nennen wir *stabilen Zustand* (oder *inaktiven Zustand*).

Vorgehensweise bei der Konstruktion eines Zustandsdiagramms

1. Wähle ein SD, das eine typische Interaktion für ein Objekt der betrachteten Klasse zeigt (üblicherweise das SD für das Primärszenario).
2. Betrachte die Projektion des SD auf die Lebenslinie dieses Objekts.
3. Bilde der Lebenslinie des Objekts folgend eine Kette von Zuständen und Transitionen, so dass
   - die Phasen in denen das Objekt nicht aktiv ist, durch stabile Zustände modelliert werden,
   - Aktivierungsphasen durch Aktivitätszustände modelliert werden (in denen lokale Operationen zur Durchführung der Aktivität aufgerufen werden),
   - eintreffende Ereignisse durch entsprechend markierte Transitionen von stabilen Zuständen in Aktivitätszustände modelliert werden,
   - die Beendigung einer Aktivität durch eine Transition mit einem Completion Event von einem Aktivitätszustand in einen stabilen Zustand modelliert wird.
4. Bilde Zyklen für Folgen, die wiederholt werden können.

{ein SD verarbeitet, noch ohne Detaillierung der Aktivierungen}

5. Solange es weitere SDs für Objekte der betrachteten Klasse gibt,
   - wähle ein solches SD und projiziere es auf die Lebenslinie des Objekts,
   - finde den Aktivitätszustand, wo die Sequenz von dem bisher beschriebenen Verhalten abweicht,
   - hänge die neue Folge als Alternative an diesen Zustand an,
   - finde (wenn möglich) einen stabilen Zustand, in dem die alternative Folge mit dem bestehenden Zustandsdiagramm vereinigt werden kann.

{alle SDs verarbeitet, noch ohne Detaillierung der Aktivierungen}

6. Konstruiere Aktivitätsdiagramme für lokale Operationen (die in Aktivitätszuständen aufgerufen werden; vgl. Schritt 3).

7. Verfeinere das bestehende Zustandsdiagramm ggf. durch Einfügen von Bedingungen bei den von den Aktivitätszuständen ausgehenden Transitionen.

{alle SDs mit allen Aktivierungen verarbeitet}

8. Integriere (soweit noch nötig) alle angegebenen Sekundärszenarien, für die kein SD vorhanden ist.

Bemerkung

- Der nach den Schritten 1-5 erhaltene Entwurf eines Zustandsdiagramms ist i.a. nicht-deterministisch. Dieser Entwurf wird in Schritt 7 zu einem deterministischen Zustandsdiagramm verfeinert.
- Eine genaue Formalisierung der angegebenen Methodik ist zu finden in:

  R. Hennicker, A. Knapp: Activity-Driven Synthesis of State Machines. Konferenzband FASE 2007, Fundamental Approaches to Software Engineering, Springer Lecture Notes in Computer Science 4422, 87-101, 2007.

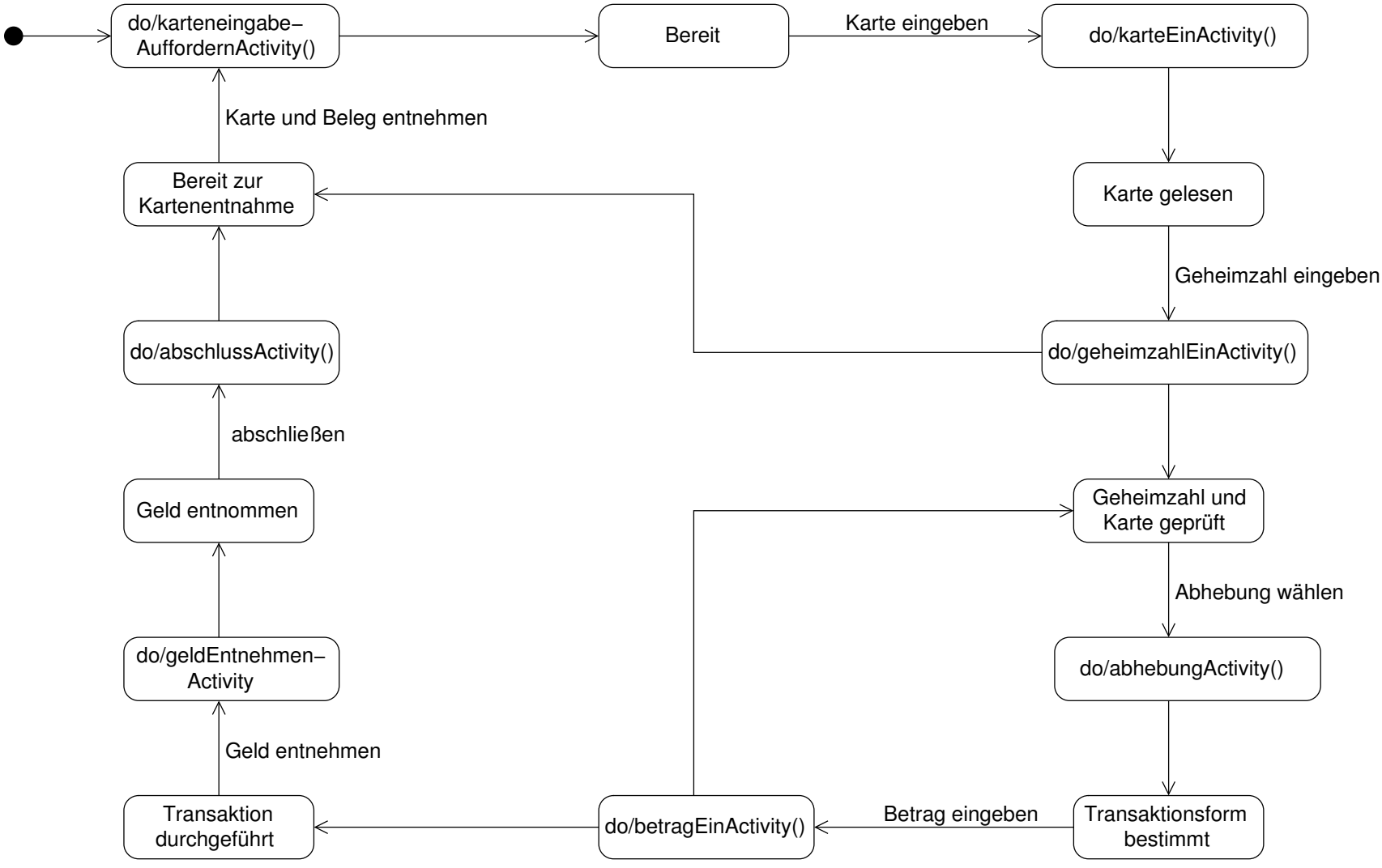
Beispiel ATM:

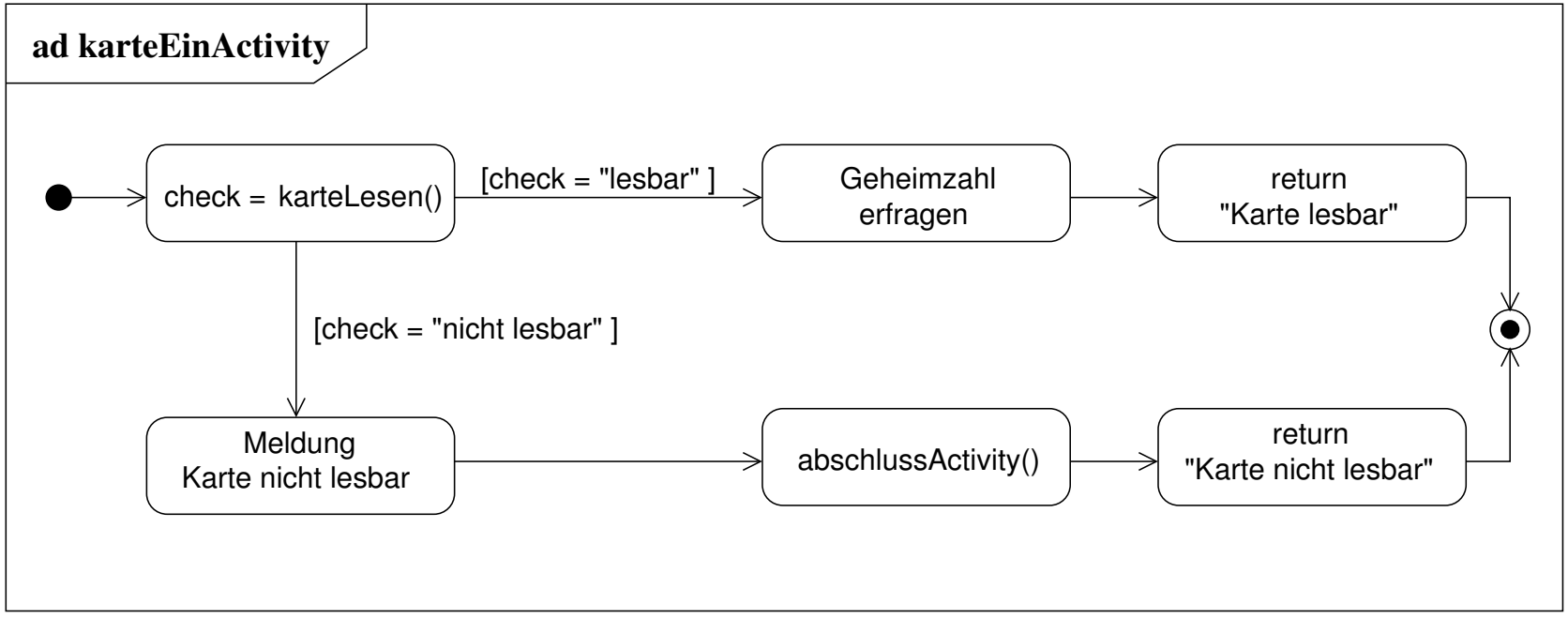
- "interessantes Verhalten": ATM (*Zustandsdiagramm*)
- "Operationen": (*Aktivitätsdiagramme*)
  - für ATM:
    - lokale Operationen
  - für Konsortium:
    - Karte überprüfen
    - Transaktion verarbeiten
  - für Bank:
    - Karte bei Bank überprüfen
    - Banktransaktion verarbeiten

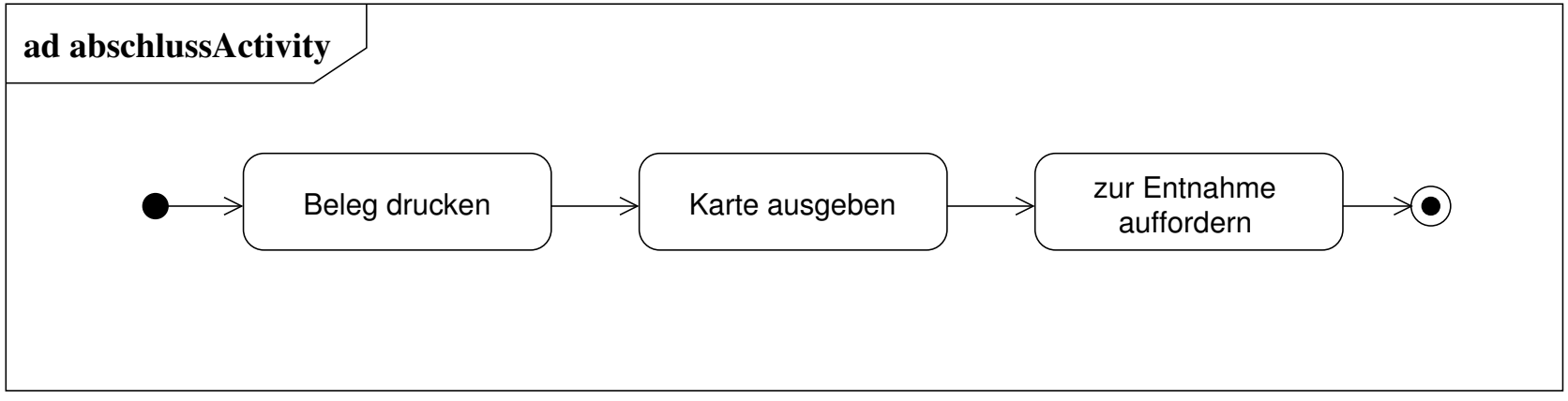
Zustandsdiagramm für ATM konstruieren

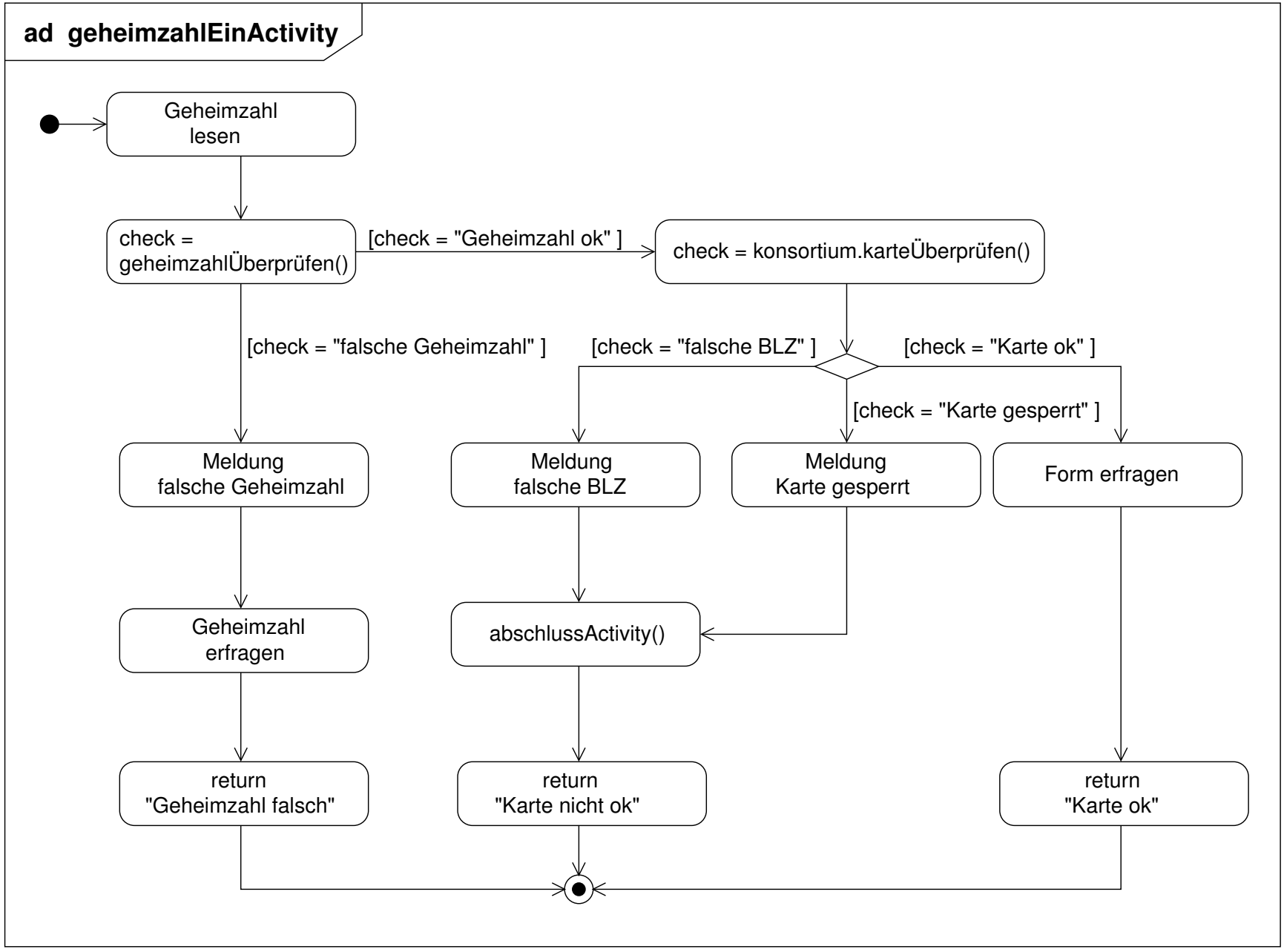
1. Wähle SD für das Primärszenario von "Geld am ATM abheben"
2. Betrachte die Lebenslinie von ":ATM"
3. Kette von Zuständen und Transitionen bilden mit einem Aktivitätszustand nach jeder Benutzerinteraktion
4. Schleife bei Zustand "Bereit"
5. SD für "Karte gesperrt" integrieren,
   SD für "Transaktion gescheitert" integrieren
6. Aktivitätsdiagramme für die in den Aktivitätszuständen aufgerufenen lokalen Operationen konstruieren
7. Bedingungen einfügen bei den von o.g. Aktivitätszuständen ausgehenden Transitionen und das bestehende Zustandsdiagramm verfeinern
8. Noch nicht berücksichtigte Sekundärszenarien integrieren
   ("Abbruch", etc.)

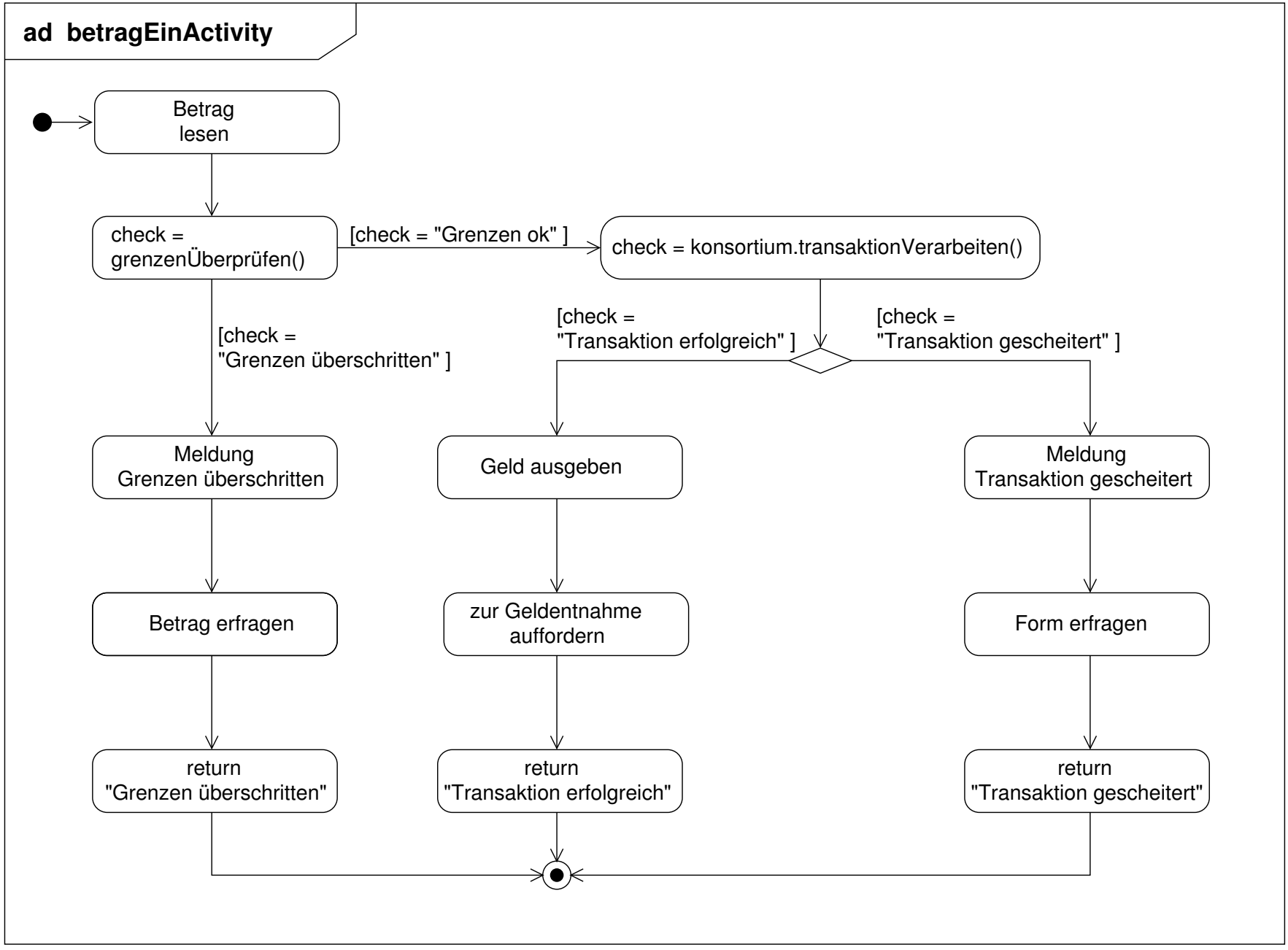
Zustandsdiagramm für ATM (nach den Schritten 1-5)

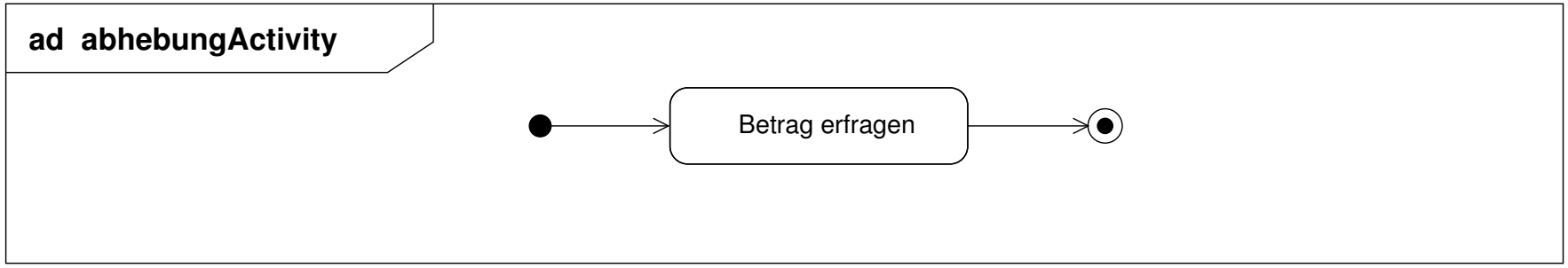


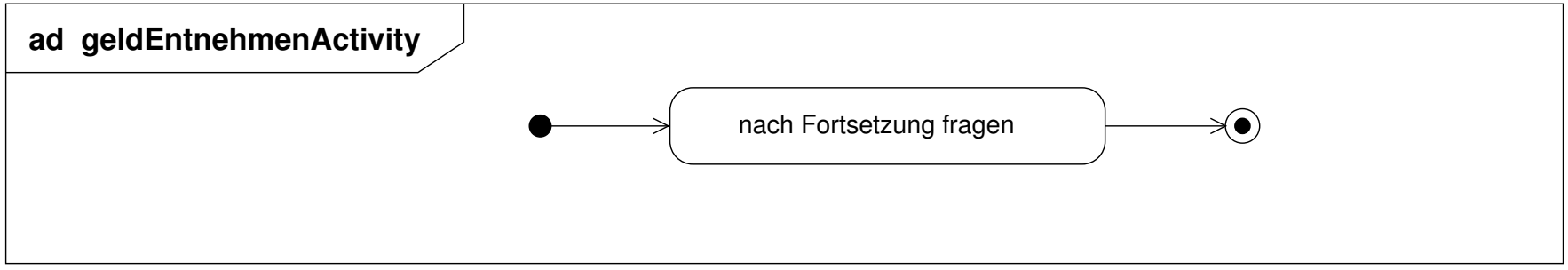

ad karteEinActivity
check = karteLesen()
[check = "lesbar"]
Geheimzahl erfragen
return "Karte lesbar"
[check = "nicht lesbar"]
Meldung Karte nicht lesbar
abschlussActivity()
return "Karte nicht lesbar"

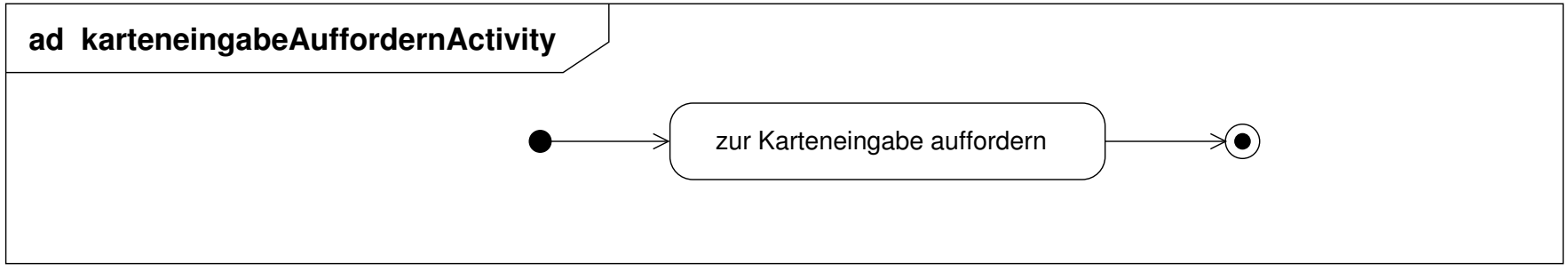

ad abschlussActivity
Beleg drucken
Karte ausgeben
zur Entnahme auffordern


ad geheimzahlEinActivity
Geheimzahl lesen
check = geheimzahlÜberprüfen()
[check = "Geheimzahl ok"]
check = konsortium.karteÜberprüfen()
[check = "falsche Geheimzahl"]
[check = "falsche BLZ"]
[check = "Karte ok"]
[check = "Karte gesperrt"]
Meldung falsche Geheimzahl
Meldung falsche BLZ
Meldung Karte gesperrt
Form erfragen
Geheimzahl erfragen
abschlussActivity()
return "Geheimzahl falsch"
return "Karte nicht ok"
return "Karte ok"

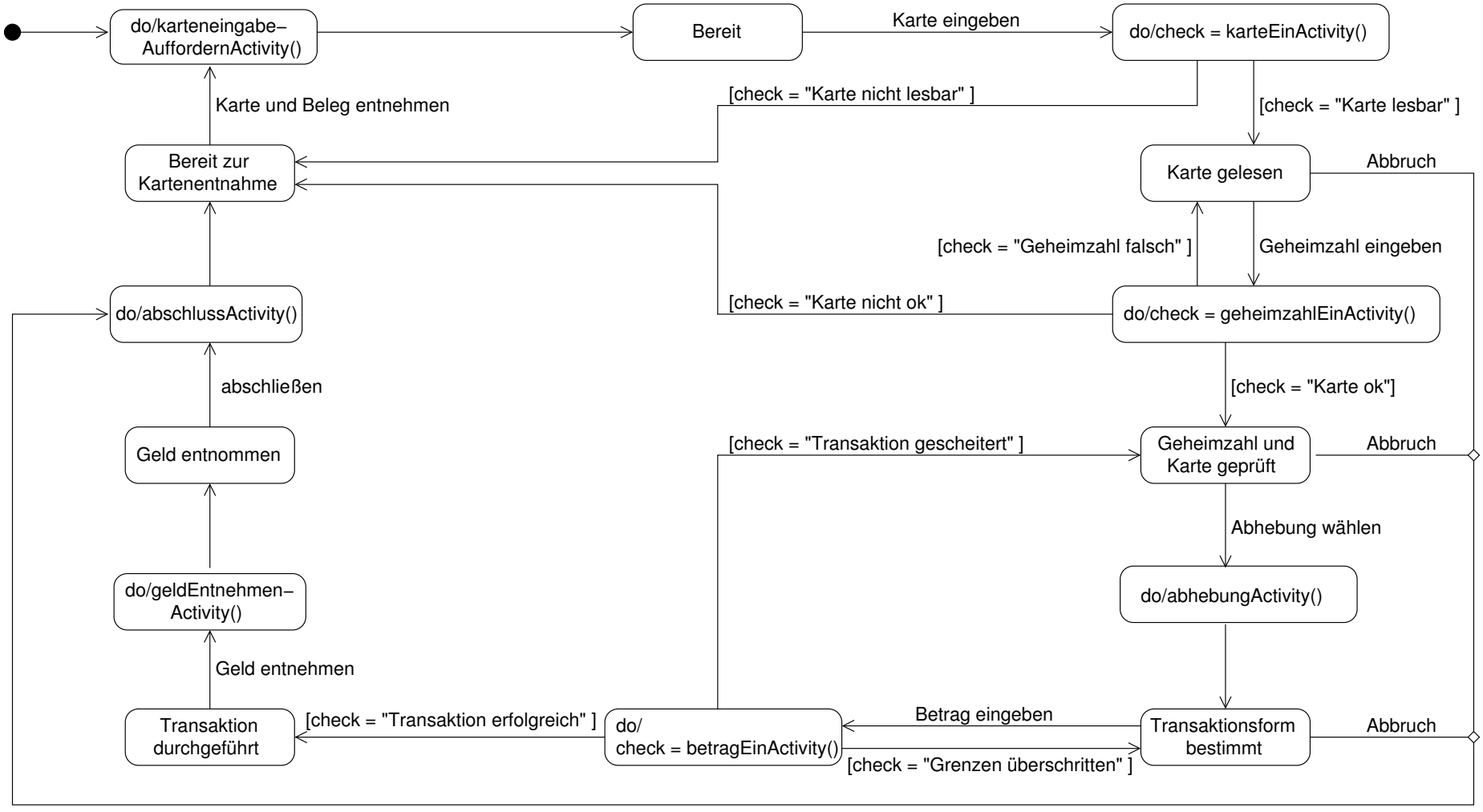

ad betragEinActivity
Betrag lesen
check = grenzenÜberprüfen()
[check = "Grenzen ok"]
check = konsortium.transaktionVerarbeiten()
[check = "Grenzen überschritten"]
[check = "Transaktion erfolgreich"]
[check = "Transaktion gescheitert"]
Meldung Grenzen überschritten
Geld ausgeben
Meldung Transaktion gescheitert
Betrag erfragen
zur Geldentnahme auffordern
Form erfragen
return "Grenzen überschritten"
return "Transaktion erfolgreich"
return "Transaktion gescheitert"


ad abhebungActivity
Betrag erfragen

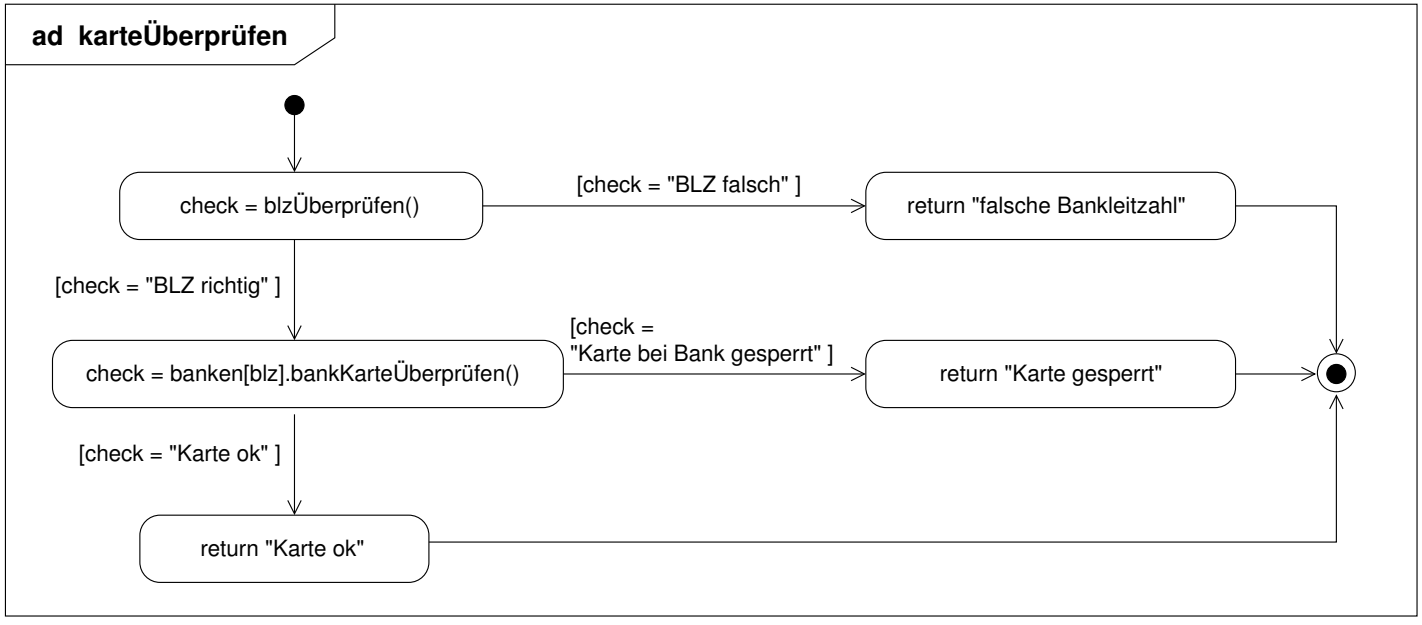

ad geldEntnehmenActivity
nach Fortsetzung fragen

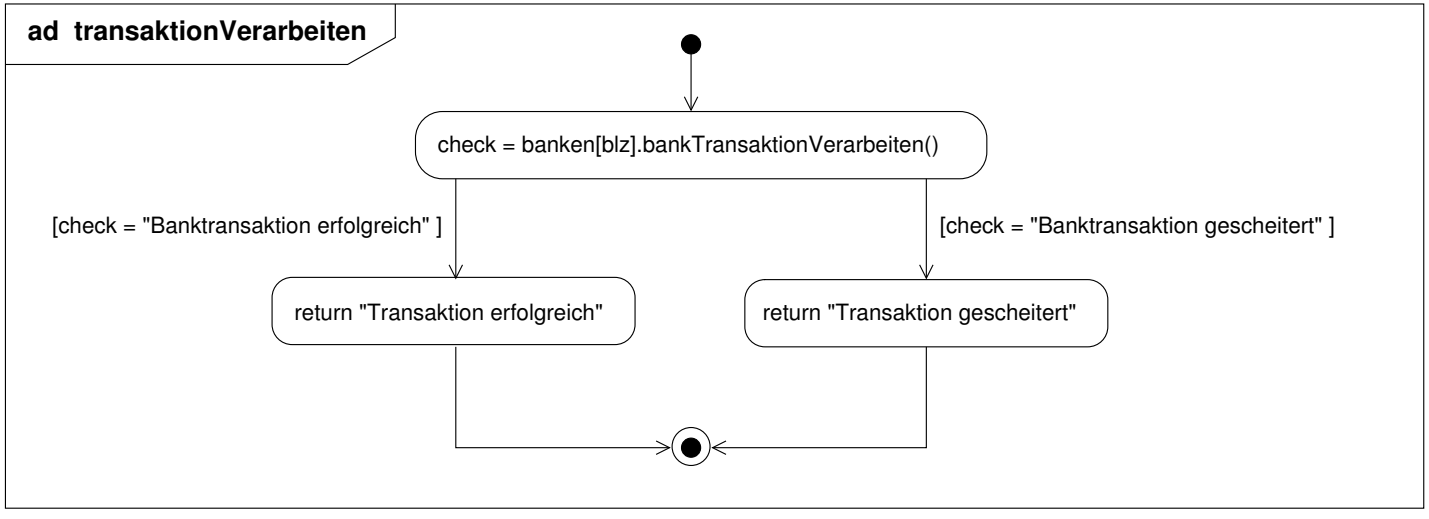

ad karteneingabeAuffordernActivity
zur Karteneingabe auffordern

Zustandsdiagramm für ATM (nach den Schritten 6-8)

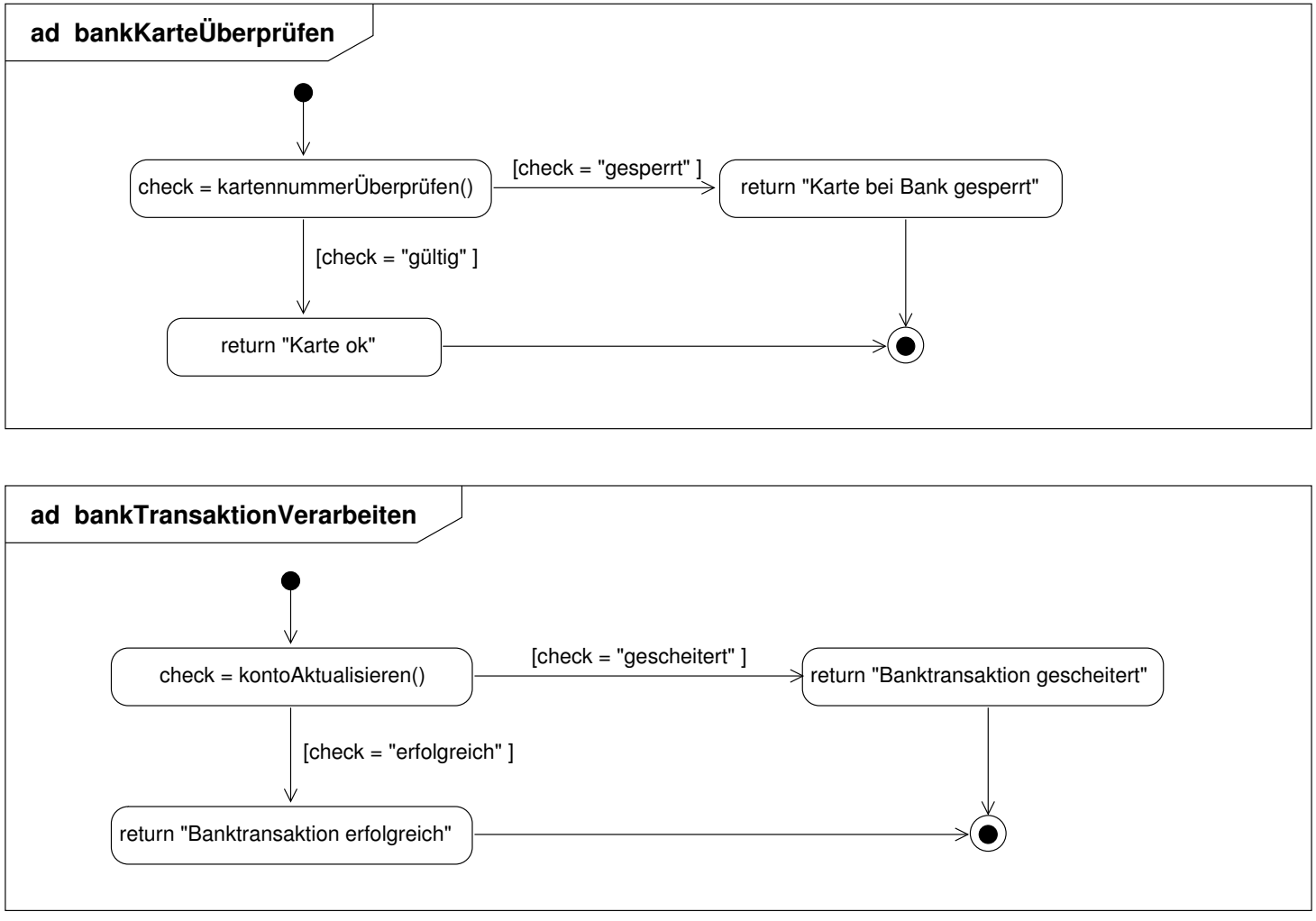


Aktivitätsdiagramme für Operationen der Klasse Konsortium





Aktivitätsdiagramme für Operationen der Klasse Bank



Bemerkung

Die Konsistenz der Diagramme untereinander kann leicht überprüft werden.

Zusammenfassung von Abschnitt 3.4

- Zustands- und Aktivitätsdiagramme können ausgehend von dem Modell der Interaktionen systematisch entwickelt werden.
- Zustandsdiagramme werden für Klassen, deren Objekte einen nicht-trivialen Lebenszyklus haben, entwickelt.
- Bei der schrittweisen Entwicklung von Zustandsdiagrammen werden stabile Zustände und Aktivitätszustände unterschieden.
- Zur Beschreibung von Operationen werden Aktivitätsdiagramme erstellt.