

Objektorientierte Software-Entwicklung

Priv.- Doz Dr. Rolf Hennicker

04.10.2002



Kapitel 3

Objektorientierte Analyse

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 Interaktionsdiagrammen, Zustands- und Aktivitätsdiagrammen)
4. Validieren, überarbeiten und erweitern der Modelle (in mehreren Iterationen)

Bemerkung

Während der Analysephase kann auch eine Grobarchitektur des Systems erstellt werden.

3.1 Anwendungsfall-Analyse

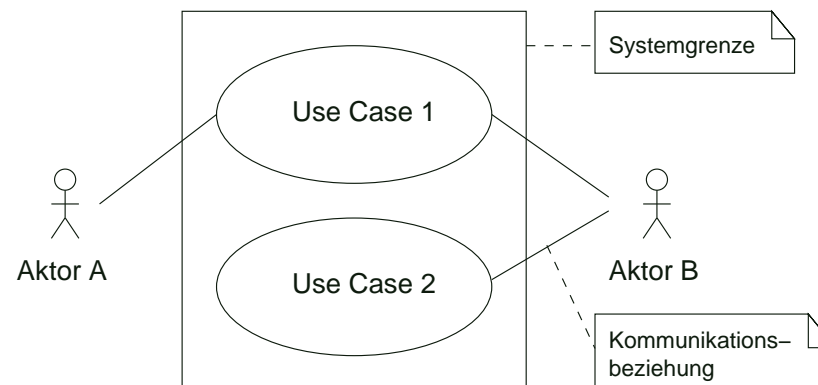
Ausgangspunkt: informelle, knappe Problembeschreibung

Ziel: Beschreibung der gewünschten Funktionalität des zu entwickelnden Systems.

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.

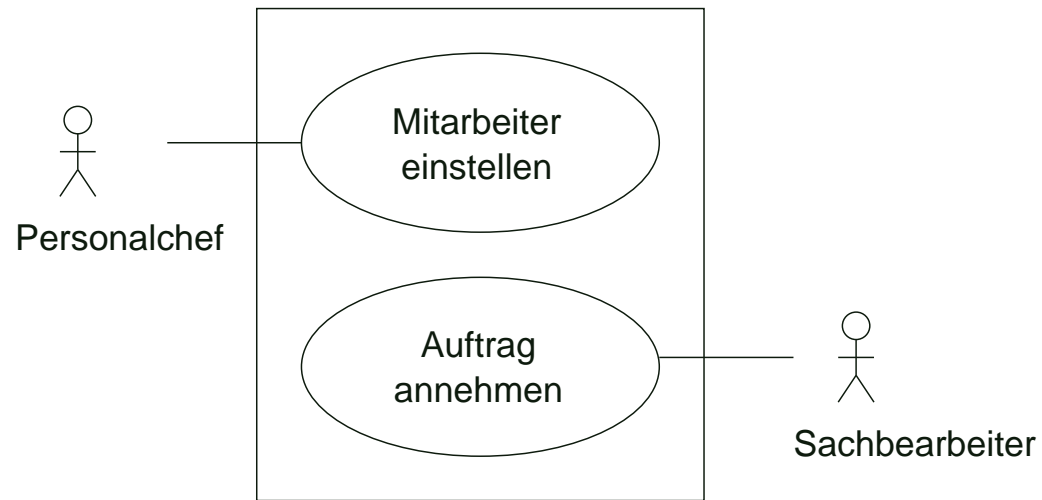
Anwendungsfall

- 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.

Vorgehensweise bei der Erstellung eines Use Case-Modells

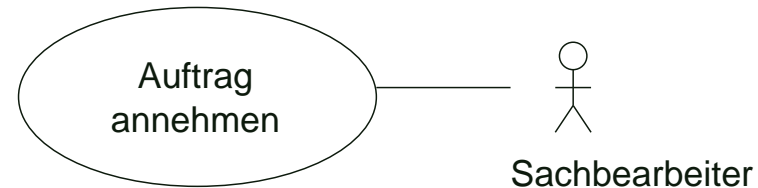
1. Bestimmung der *Aktoren*, die mit dem System interagieren.
(Wer benutzt das System? Wer holt/liefert Informationen von/für das System?)
2. Bestimmung der *Anwendungsfälle* aufgrund der Aufgaben, die das System für jeden einzelnen Aktor erledigen soll (z.B. durch Interviews).
Schwierigkeit: richtige 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)
- Standardablauf (Primärszenario)
(Schritte bzw. Interaktionen, die im Normalfall bei Ausführung des Anwendungsfalls durchlaufen werden)
- Alternativabläufe (Sekundärszenarien)
(bei Fehlern und selteneren Fällen)

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:

Die Bestellung ist registriert.

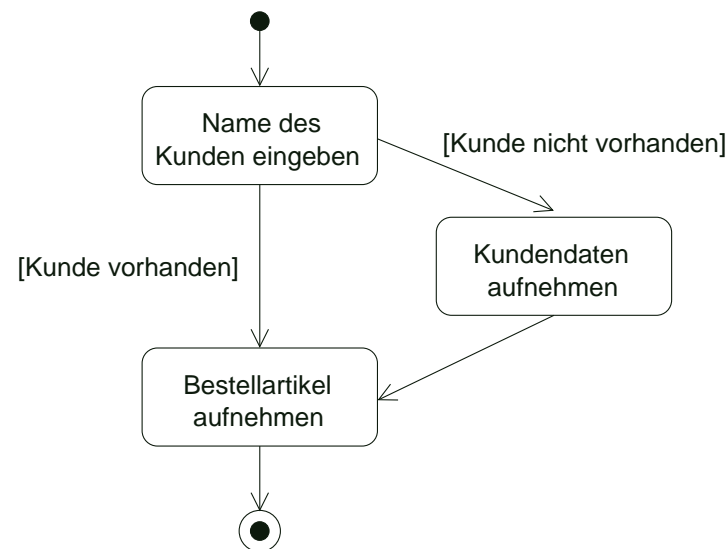
Primärszenario:

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:

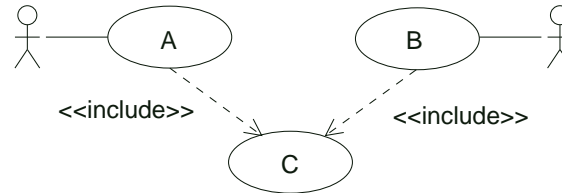
Kunde nicht vorhanden

Aktivitätsdiagramm:



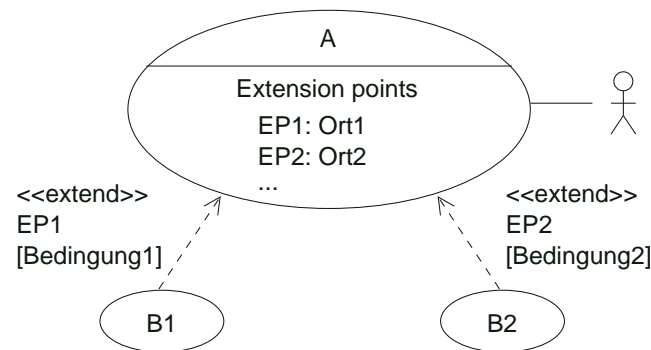
Beziehungen zwischen Anwendungsfällen

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.

Beispiel: ATM (automatic teller machine)

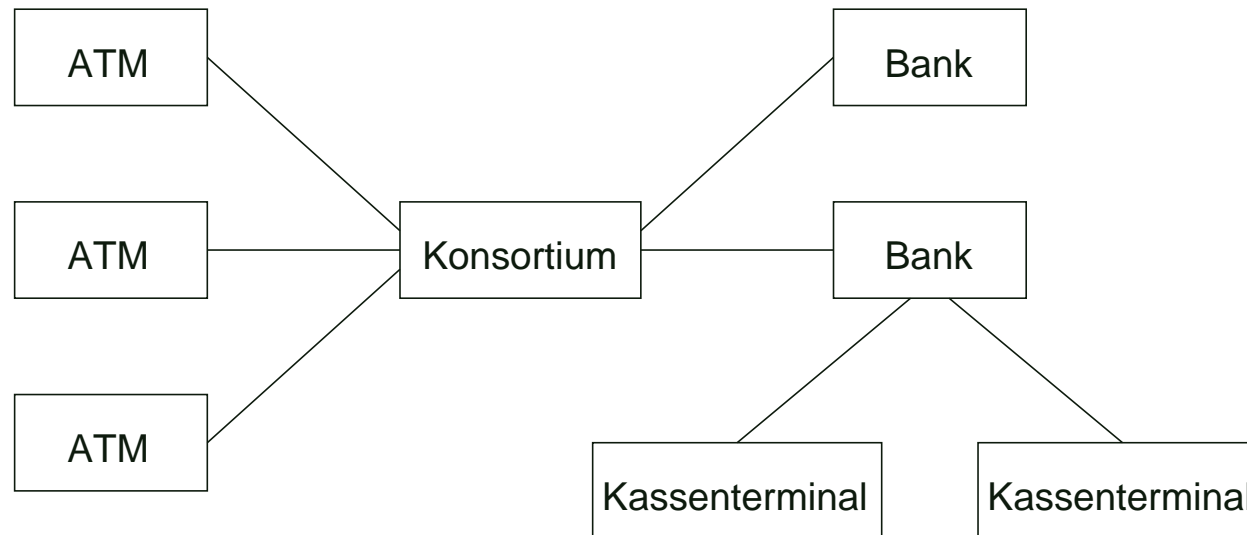
Problembeschreibung: Netzwerk von Bankautomaten
(aus Rumbaugh et al., 1993)

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.

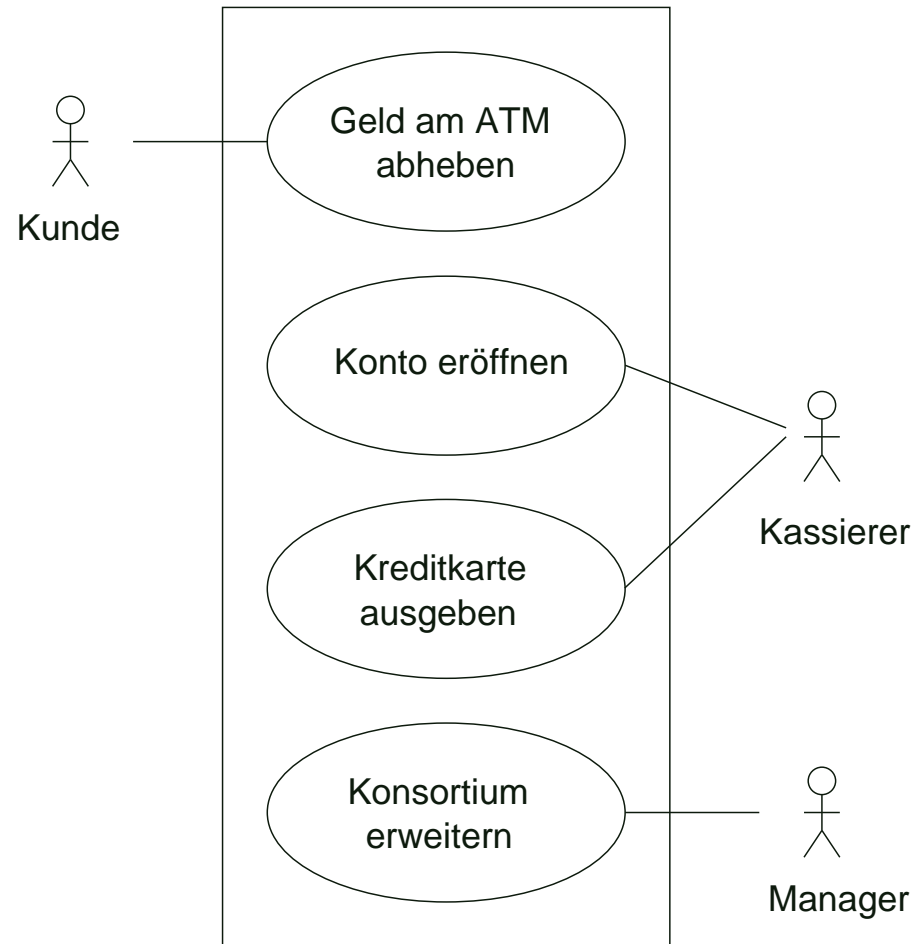
Sie sollen die Software für die ATMs und das Netzwerk entwickeln. Die Kosten des gemeinsamen Systems werden nach Zahl der Scheckkarteninhaber auf die Banken umgelegt.



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 zeigt den Hauptbildschirm und fordert den Benutzer auf eine Kreditkarte einzugeben.

Nachbedingung:

Der Benutzer hat die Kreditkarte, das Geld und den Beleg entnommen. Das ATM zeigt den Hauptbildschirm und fordert den nächsten Benutzer auf eine Kreditkarte einzuschieben.

Primärszenario:

1. Der Kunde schiebt seine Kreditkarte ein.
2. Das ATM akzeptiert die Kreditkarte, liest die BLZ und die Kartenummer und fordert daraufhin die Geheimzahl an.
3. Der Benutzer gibt die Geheimzahl ein.
4. Das ATM überprüft die Geheimzahl und lässt die BLZ und die Kartenummer beim Konsortium überprüfen.
5. Das Konsortium überprüft die BLZ, gleicht die Kartenummer mit der Bank des Kunden ab und gibt dem ATM sein OK.
6. Das ATM fordert den Benutzer auf die Transaktionsform (Abhebung, Einzahlung, Überweisung, Anfrage) zu wählen.
7. Der Benutzer wählt "Abhebung", woraufhin das ATM den Betrag erfragt.
8. Der Benutzer gibt den gewünschten Betrag ein.
9. Das ATM überprüft, ob der Betrag innerhalb vordefinierter Grenzen liegt und fordert 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.

Sekundärszenarien:

Falsche Geheimzahl

In Schritt 4, wenn das ATM feststellt, dass die Geheimzahl falsch ist, fordert es den Benutzer erneut auf die Geheimzahl einzugeben. Der Use Case wird ab Schritt 3 wiederholt.

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

Der Kunde kann immer, wenn eine Eingabe von ihm erwartet wird, den Vorgang abbrechen. Das ATM meldet den Abbruch. Der Use Case wird ab Schritt 15 fortgesetzt.

Karte nicht lesbar

Falsche Bankleitzahl

Grenzen überschritten

Karte abgelaufen

Kein Geld im ATM

Netzwerk unterbrochen

Bemerkung:

Beschreibungen der übrigen Anwendungsfälle können entweder sofort oder iterativ nach Durchführung weiterer Entwicklungszyklen erfolgen.

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 (einschl. Aggregationen) bestimmen
3. Attribute identifizieren
4. Vererbung einführen
5. Modell überarbeiten

1. Klassen identifizieren

Kandidaten sind

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

1. Schritt: Kandidaten notieren

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	Betrag
Kreditkarte	Grenzen
ATM	Transaktion
Bankleitzahl	Anforderung
Kartenummer	Kontostand
Benutzer	Ausführung
Geheimzahl	Abschluss
Konsortium	Bargeld
Bank	Beleg
Transaktionsform	Karte
Abhebung, Einzahlung, Überweisung, Anfrage	

2. Schritt: Ungeeignete Klassen streichen

Kunde

Kreditkarte

ATM

Bankleitzahl (*Attribut*)

Kartenummer (*Attribut*)

Benutzer (*redundant*)

Geheimzahl (*Attribut*)

Konsortium

Bank

Transaktionsform (*Attribut*)

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

Betrag (*Attribut*)

Grenzen (*Attribut*)

Transaktion

Anforderung (*Tätigkeit*)

Kontostand (*Attribut*)

Ausführung (*Tätigkeit*)

Abschluss (*Tätigkeit*)

Bargeld (*irrelevant*)

Beleg (*vage*)

Karte (*redundant*)

3. *Schritt*: Fehlende Klassen hinzunehmen

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

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, ...)
- Zusammenarbeit von Objekten zur Lösung einer Aufgabe

1. Schritt: Kandidaten notieren

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 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 ist wichtig!

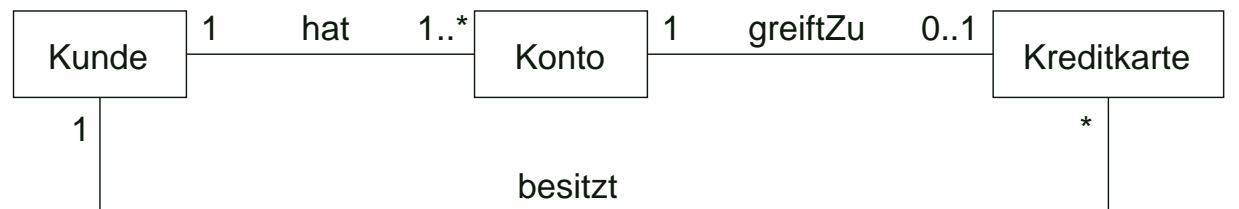
Beispiel ATM:

- 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, insbesondere auf redundante (abgeleitete) Assoziationen möglichst verzichten.

Beispiel ATM:



redundante Assoziation, die NICHT in das statische Modell der Analyse aufgenommen werden soll!

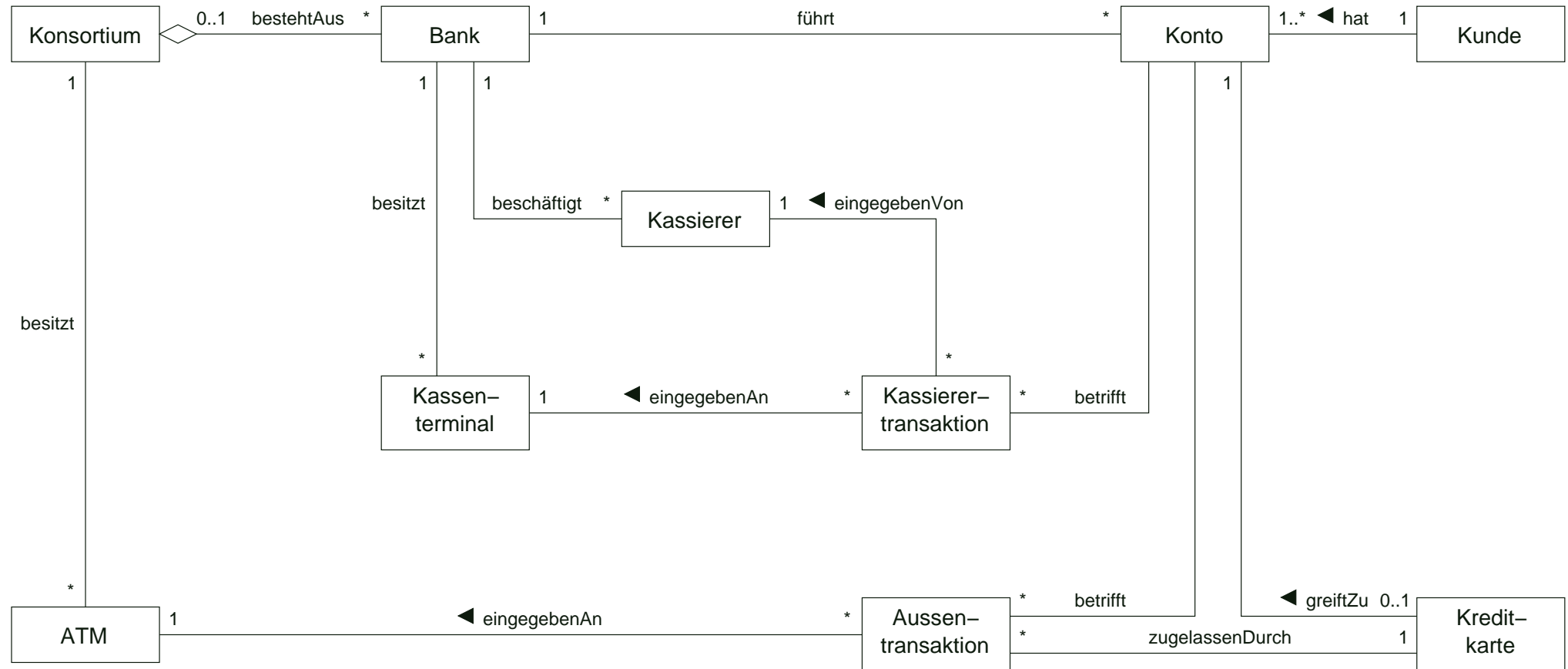
3. Schritt: Fehlende relevante Assoziationen hinzunehmen

4. Schritt: Multiplizitäten und ggf. explizite Rollennamen hinzufügen

Bemerkung

Multiplizitäten und Rollennamen können auch erst bei der Überarbeitung des Modells bestimmt werden.

Beispiel ATM: Erstes Klassendiagramm



3. Attribute identifizieren

Richtlinien

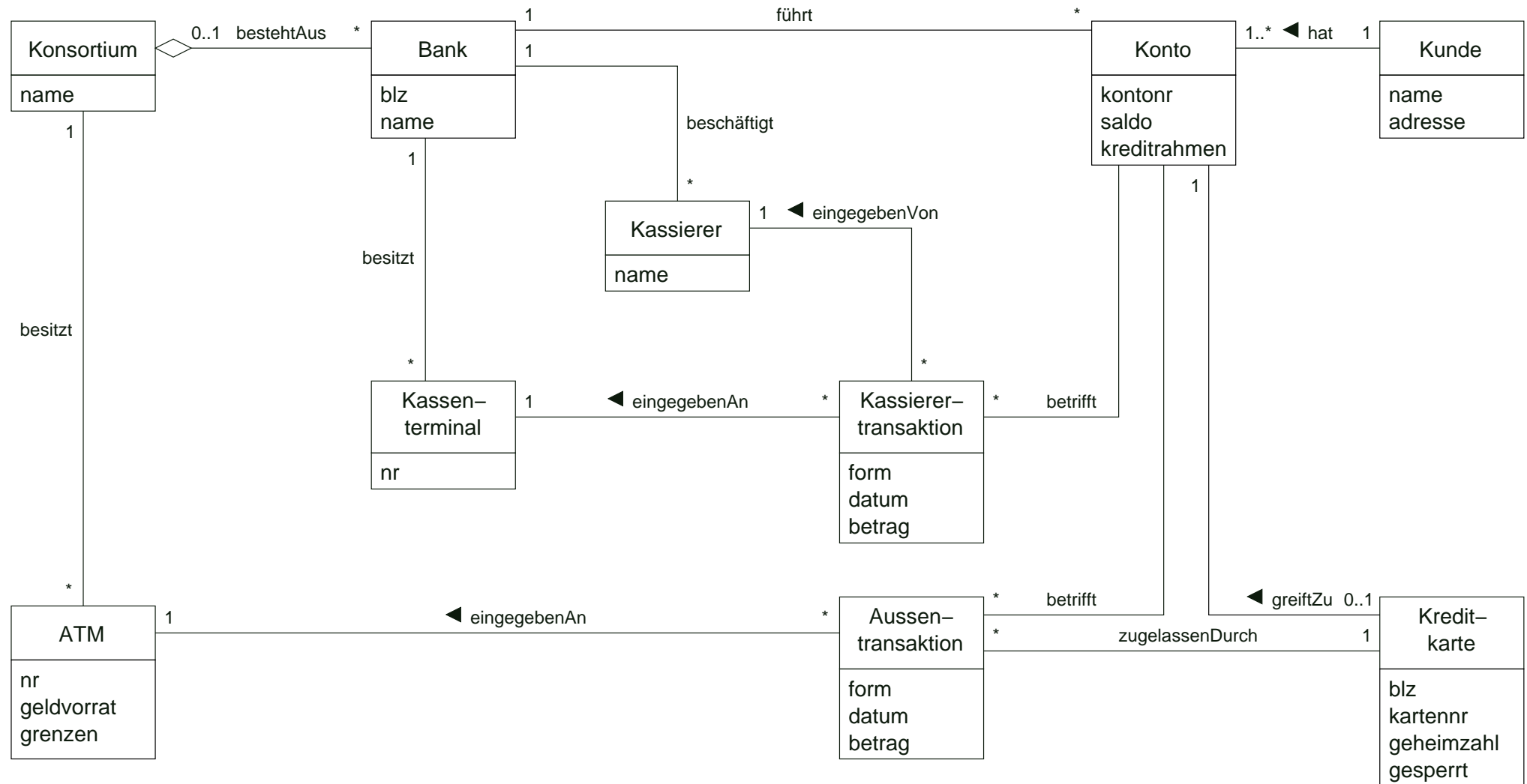
- nur Attribute, die für die Anwendung relevant sind (Problembereichswissen wichtig!)
- keine Attribute, deren Werte Objekte sind (dafür Assoziationen verwenden!)

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



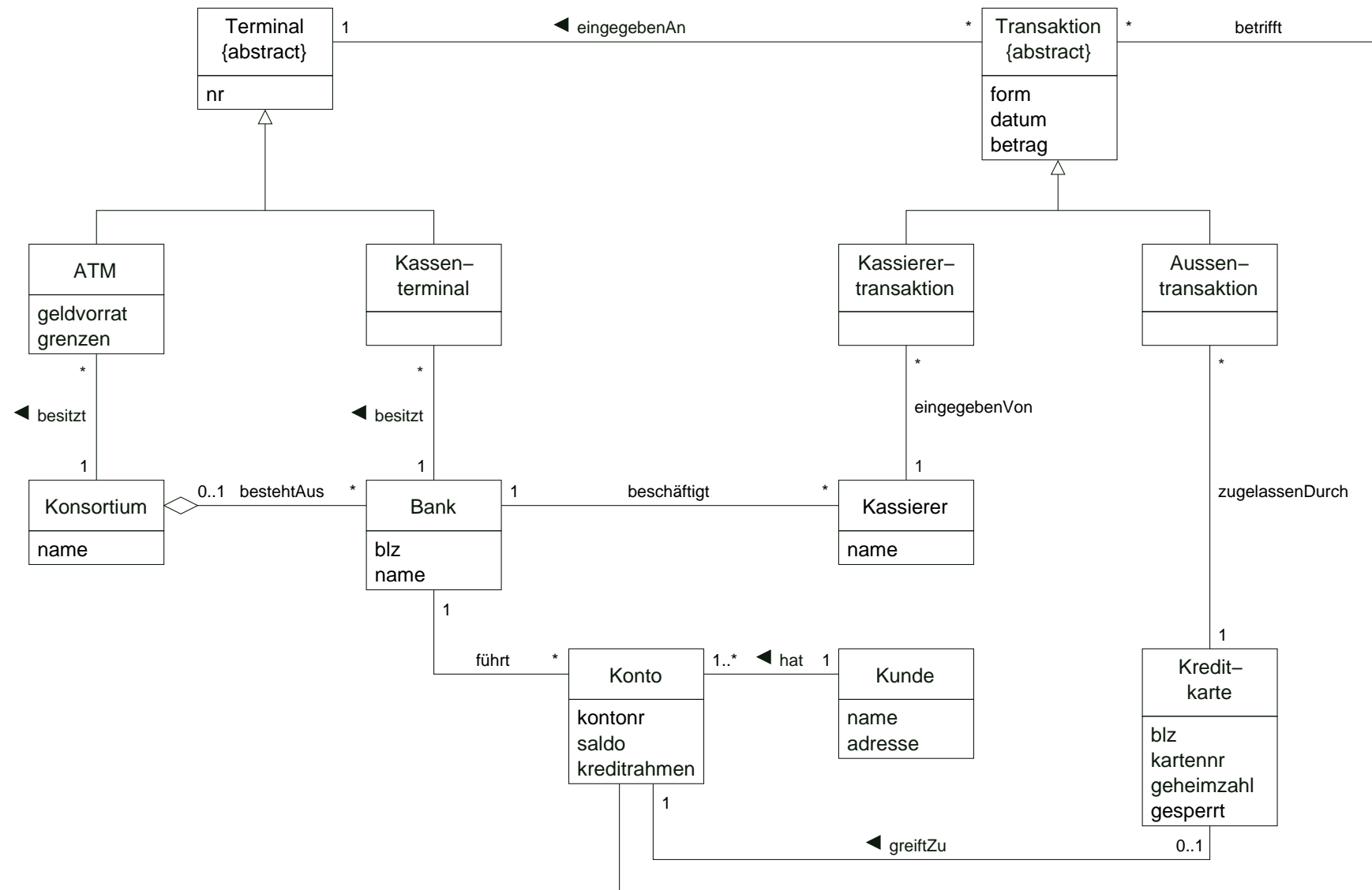
4. Vererbung einführen

- Vor allem **Generalisierung**
- Zusammenfassen gemeinsamer Merkmale vorhandener Klassen (Attribute, Assoziationen) in einer Oberklasse.

Beispiel ATM:

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

Beispiel ATM: Klassendiagramm mit Vererbung



Modell überarbeiten

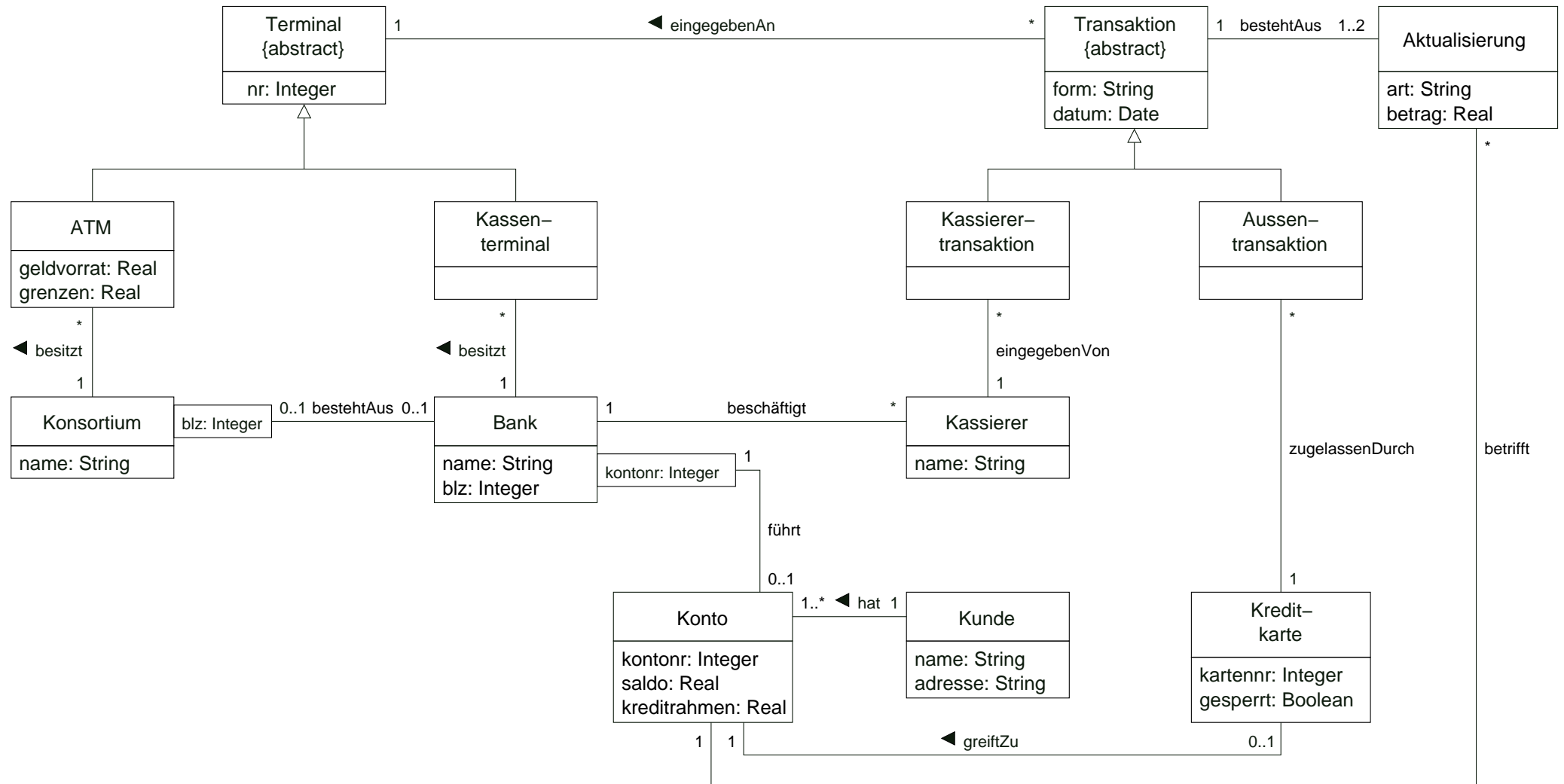
Fragen

- Fehlende Klassen oder Assoziationen?
- Unnötige Klassen oder Assoziationen?
- Falsche Platzierung von Assoziationen und Attributen?

Beispiel ATM:

- Eine Überweisungstransaktion betrifft zwei Konten
⇒ Einführung der Klasse "Aktualisierung".
- Eine Kreditkarte ist ein Stück Plastik! In dem zugehörigen Software-Objekt ist die Geheimzahl *nicht* gespeichert. Ebenso wenig die Bankleitzahl (wäre redundant).

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:

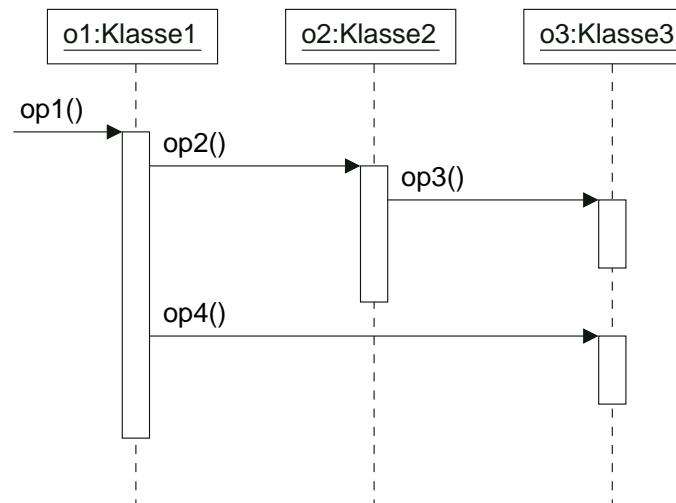
Entwurf einer Menge von *Interaktionsdiagrammen* für jeden Anwendungsfall.

Man unterscheidet zwei Arten von Interaktionsdiagrammen:

- Sequenzdiagramme
- Kollaborationsdiagramme

Sequenzdiagramme

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

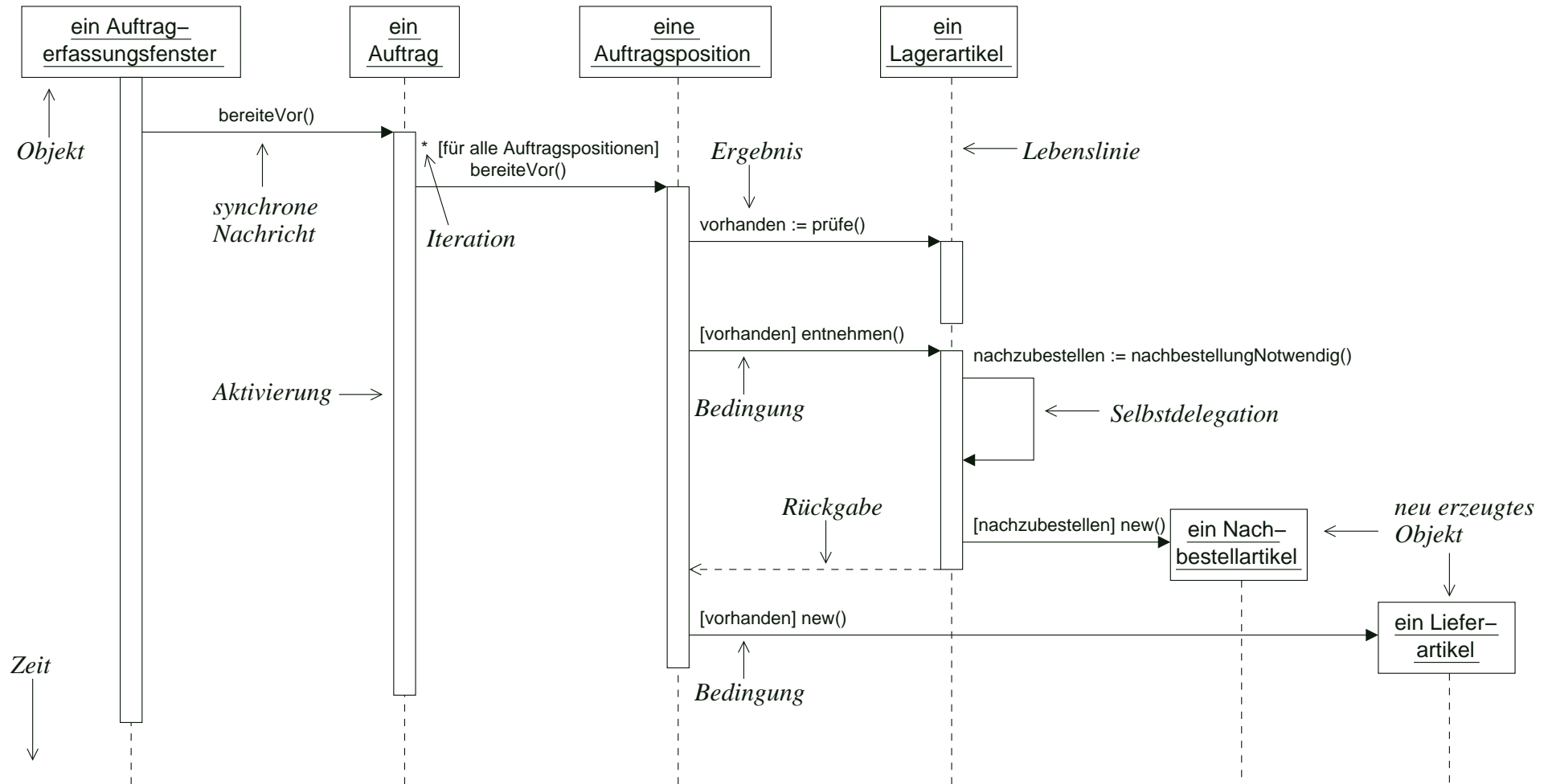


Beachte:

Empfangen einer Nachricht ist ein **Ereignis**.

Senden einer Nachricht ist eine **Aktion**.

Beispiel: Sequenzdiagramm für "Lieferung vorbereiten"

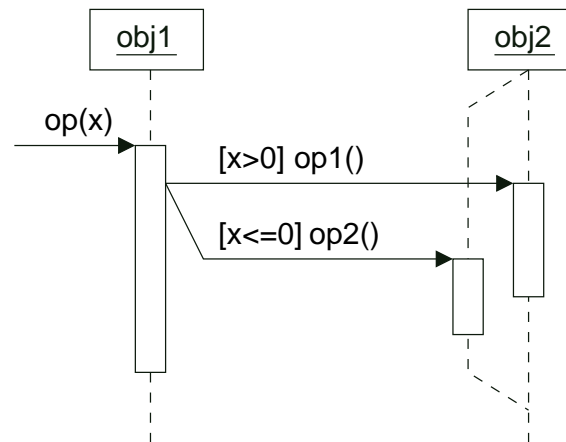


Aktivierung

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")

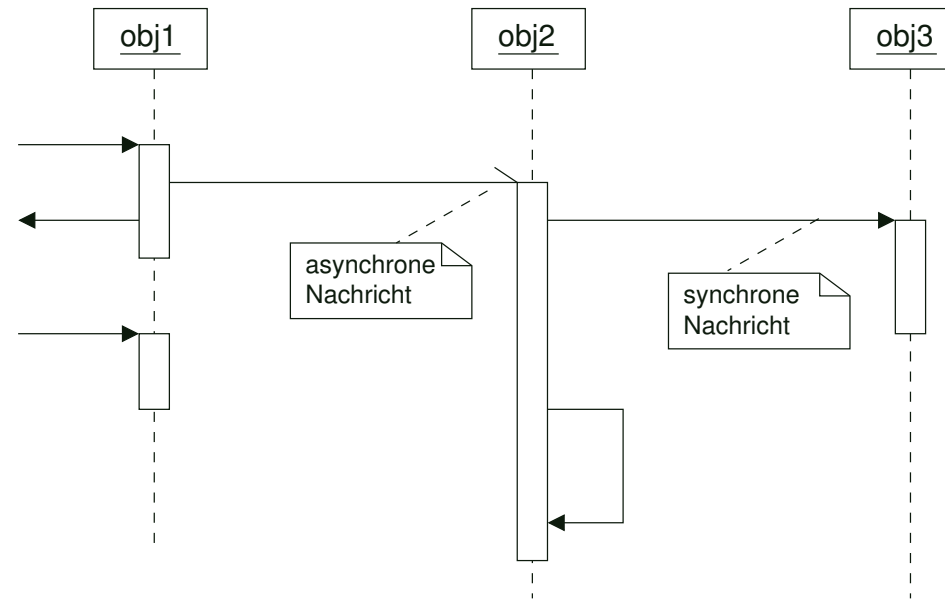
Verzweigungen



Beachte:

Mit Verzweigungen werden Sequenzdiagramme schnell unübersichtlich.

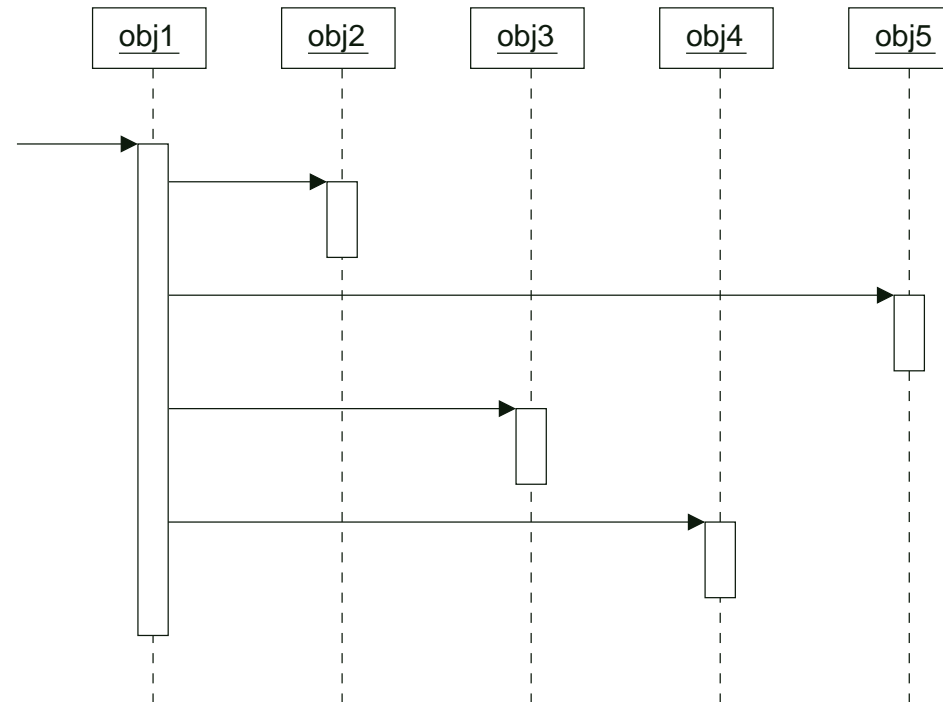
Asynchrone Nachrichten und parallele Ausführung



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

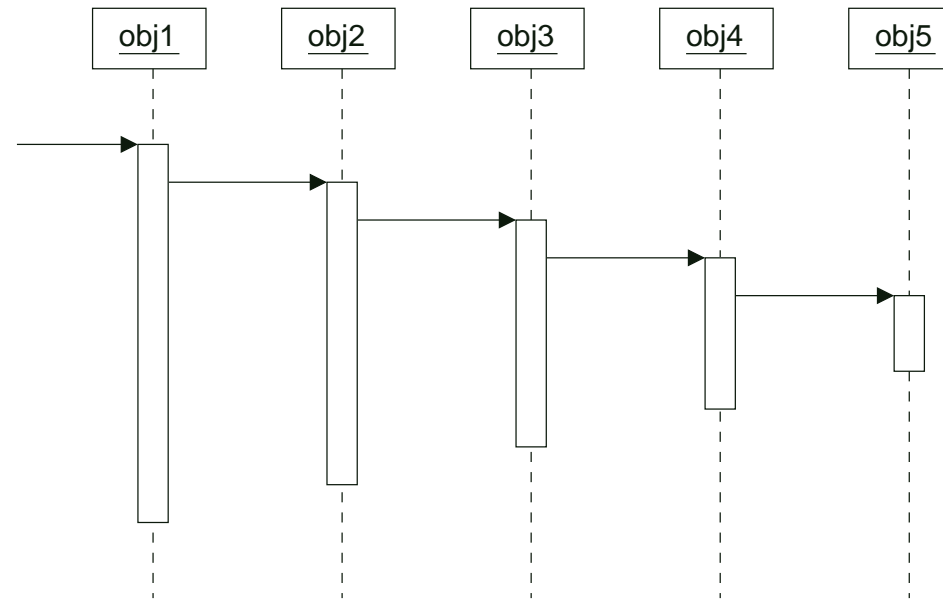
Formen von Sequenzdiagrammen

Zentralisierte Struktur



Ein Objekt (obj1) kontrolliert die anderen Objekte (besitzt die Verantwortung für die erfolgreiche Ausführung).

Dezentralisierte Struktur

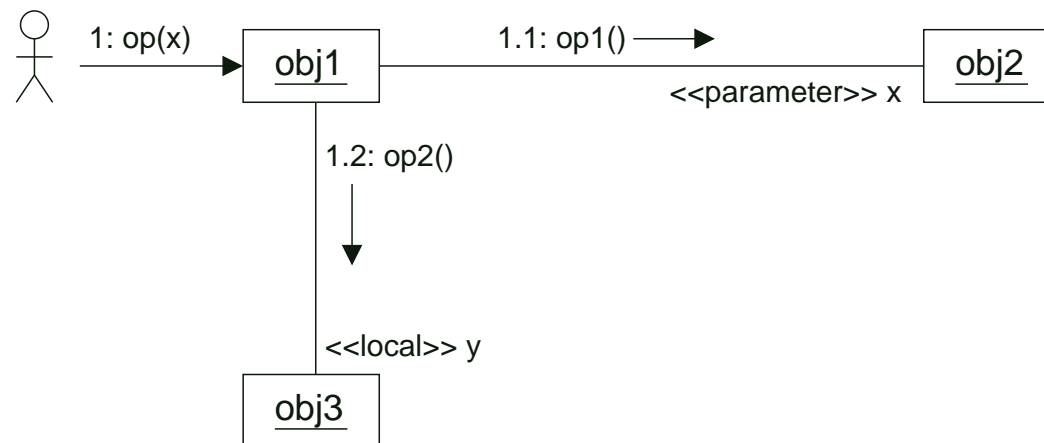


Jedes Objekt trägt eine eigene Verantwortung für die erfolgreiche Ausführung.

Kollaborationsdiagramme

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

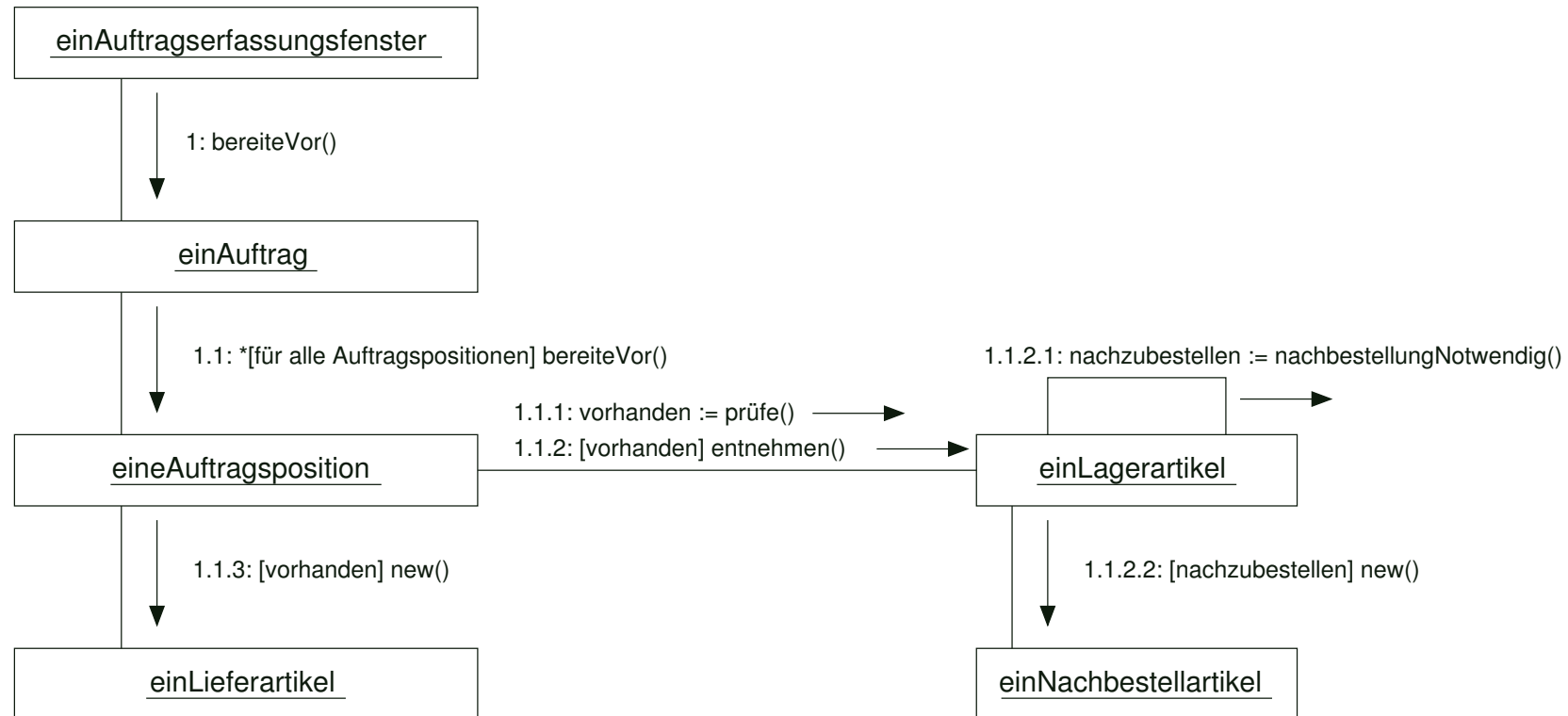
Die zeitliche Reihenfolge wird durch Nummerierung der Nachrichten dargestellt (mit dezimalen Nummern 1.1, 1.2 etc. für geschachtelte Nachrichten).



Bemerkung

Die Links in Kollaborationsdiagrammen sind häufig Instanzen von Assoziationen.

Beispiel: Kollaborationsdiagramm für "Lieferung vorbereiten"



Bemerkung

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

Entwurf von Interaktionsdiagrammen

Input

- Use Case-Beschreibungen
- 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.

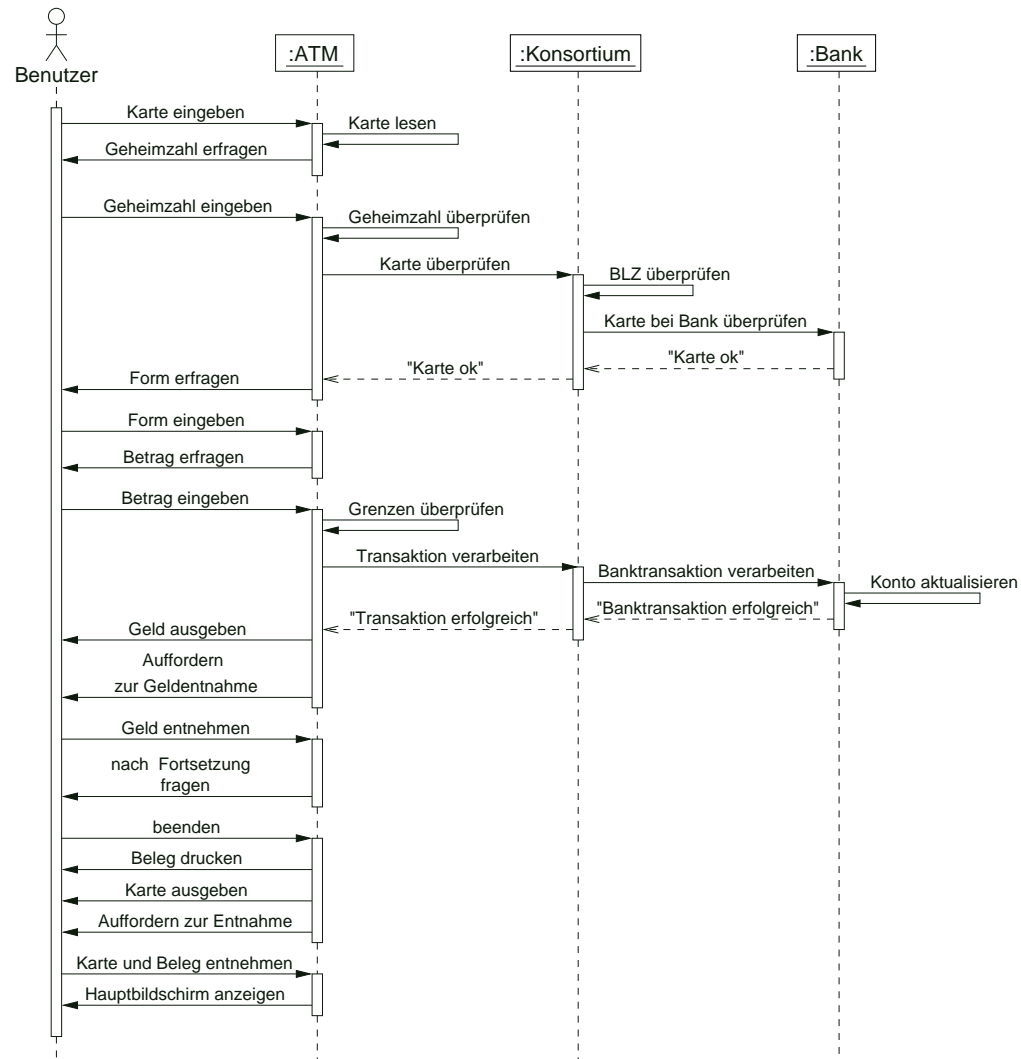
Möglichkeiten dazu:

- A. Für jedes Szenario eines Use Case ein eigenes Interaktionsdiagramm.
- B. Ein komplexes Interaktionsdiagramm (mit Verzweigungen, Iterationen ...), das alle Szenarien eines Use Case subsummiert (Nachteil: Wird schnell unübersichtlich!).

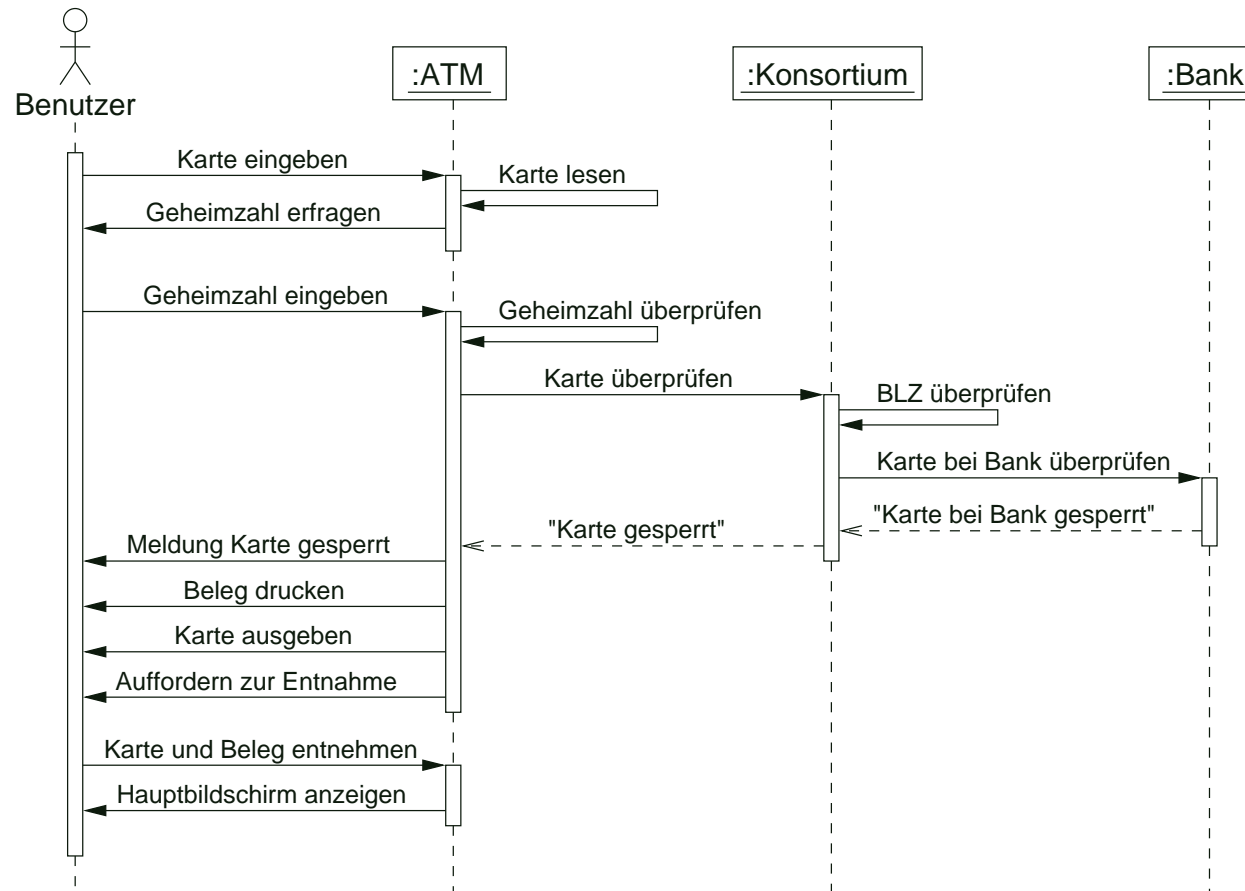
Wir orientieren uns an Möglichkeit A und verwenden Sequenzdiagramme (SDs).

Beispiel: Use Case "Geld am ATM abheben"

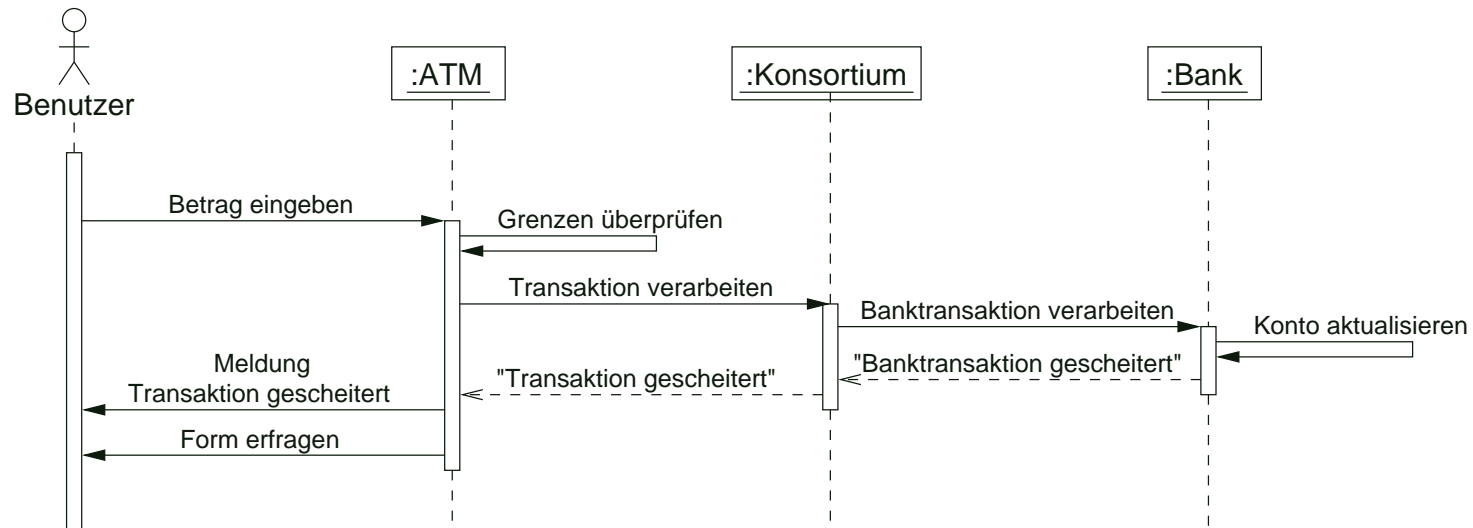
Sequenzdiagramm für das Primärszenario



Sequenzdiagramm für Sekundärszenario "Karte gesperrt"



Sequenzdiagramm für Sekundärszenario "Transaktion gescheitert"



Zusammenfassung von Abschnitt 3.3

- Interaktionsdiagramme beschreiben die Zusammenarbeit und den Nachrichtenaustausch zwischen *mehreren* Objekten.
- Wir unterscheiden Sequenzdiagramme (heben die zeitliche Reihenfolge hervor) und Kollaborationsdiagramme (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).
- Je ein Aktivitätsdiagramm für (jede) komplexe Operation, deren Verhalten nicht vollständig in einem SD erfasst ist.

Bemerkung

- Zustands- und Aktivitätsdiagramme erfassen das (vollständige) Verhalten *eines* Objektes bzw. *einer* Operation über viele Szenarien hinweg.
- Auf Aktivitätsdiagramme kann verzichtet werden, wenn das Verhalten einer Operation schon vollständig in *einem* Interaktionsdiagramm beschrieben ist (z.B. mit Fallunterscheidungen und Iterationen).

Kriterien für "interessantes Verhalten" eines Objektes:

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



Beispiel: Stellen einer Digitaluhr

- Mindestens ein Ereignis kann oder darf nur in *bestimmten Zuständen* auf ein Objekt angewendet werden.

Beispiel: ATM



Typische zustandsabhängige Objekte sind:

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

Vorgehensweise bei der Konstruktion eines Zustandsdiagramms

1. Wähle ein SD, das eine typische Interaktion für ein Objekt der betrachteten Klasse zeigt.
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 Zustände die Phasen zwischen den (eintreffenden) Ereignissen repräsentieren,
 - die Transitionen mit den Ereignissen markiert sind,
 - Reaktionen auf ein Ereignis durch einen zusätzlich eingefügten Aktions- oder Aktivitätszustand modelliert werden.
4. Bilde Zyklen für Folgen, die wiederholt werden können.
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 Zustand, wo die Sequenz von dem bisher beschriebenen Verhalten abweicht,
 - hänge die neue Folge als Alternative an diesen Zustand an,
 - finde (wenn möglich) einen Zustand, in dem die alternative Folge mit dem bestehenden Zustandsdiagramm vereinigt werden kann.

6. Konstruiere (genestete) Aktivitätsdiagramme für die Aktivitätszustände.
7. Integriere alle angegebenen Sekundärszenarien, für die kein Sequenzdiagramm vorhanden ist.
8. Verfeinere das bestehende Zustandsdiagramm durch Einfügen von Bedingungen bei den von den Aktivitätszuständen ausgehenden Transitionen.

Bemerkung

- Der nach den Schritten 1-5 erhaltene Entwurf eines Zustandsdiagramms ist i.a. nichtdeterministisch. Dieser Entwurf wird in Schritt 8 zu einem deterministischen Zustandsdiagramm verfeinert.
- Zustände, die weder Aktions- noch Aktivitätszustände sind, heißen "stabile" Zustände.

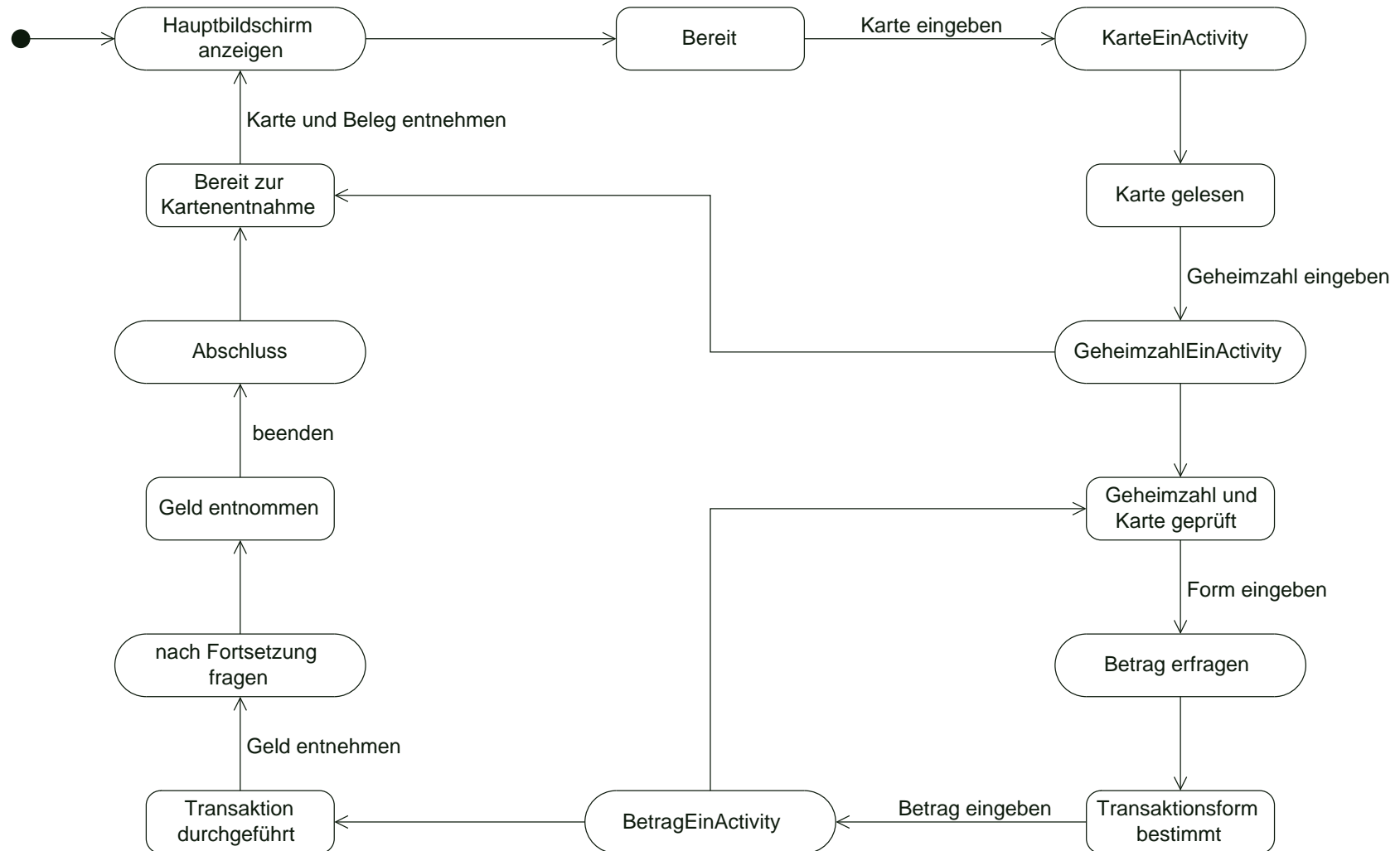
Beispiel ATM:

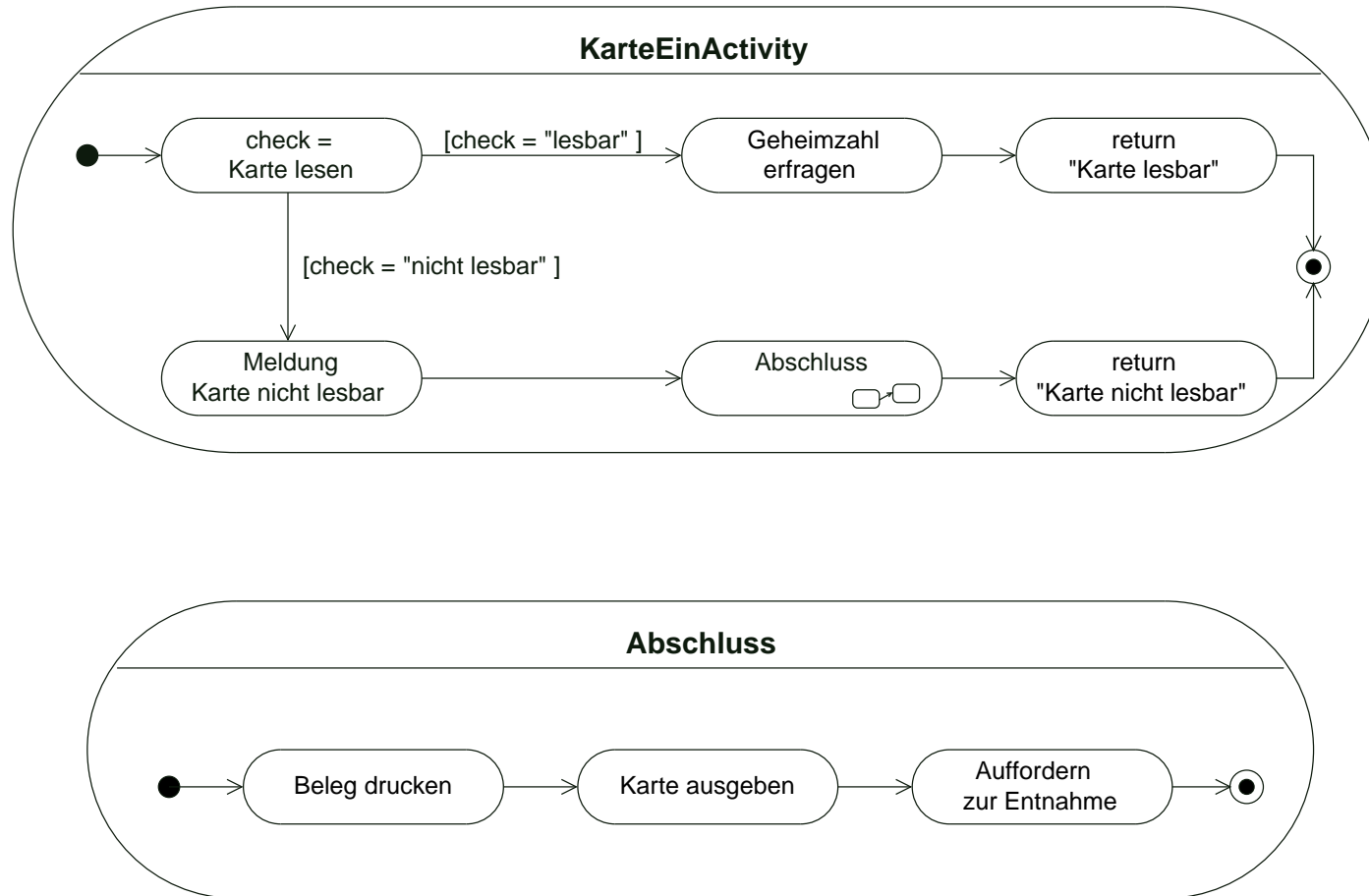
- "interessantes Verhalten": ATM (*Zustandsdiagramm*)
- "komplexe Operationen": (*Aktivitätsdiagramme*)
 - für Konsortium:
 - ▷ Karte überprüfen
 - ▷ Transaktion verarbeiten
 - für Bank:
 - ▷ Karte bei Bank überprüfen
 - ▷ Banktransaktion verarbeiten

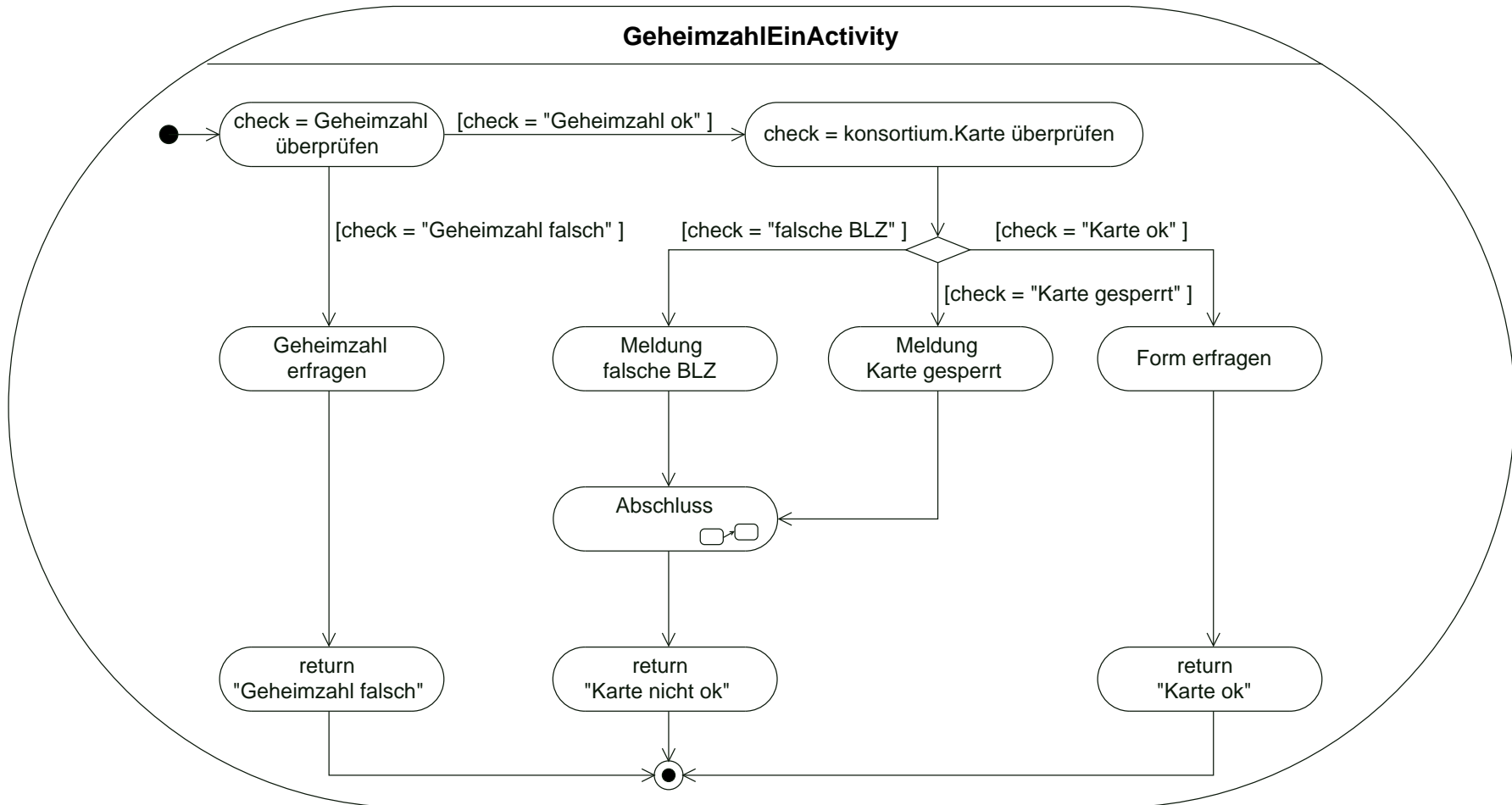
Zustandsdiagramm für ATM konstruieren

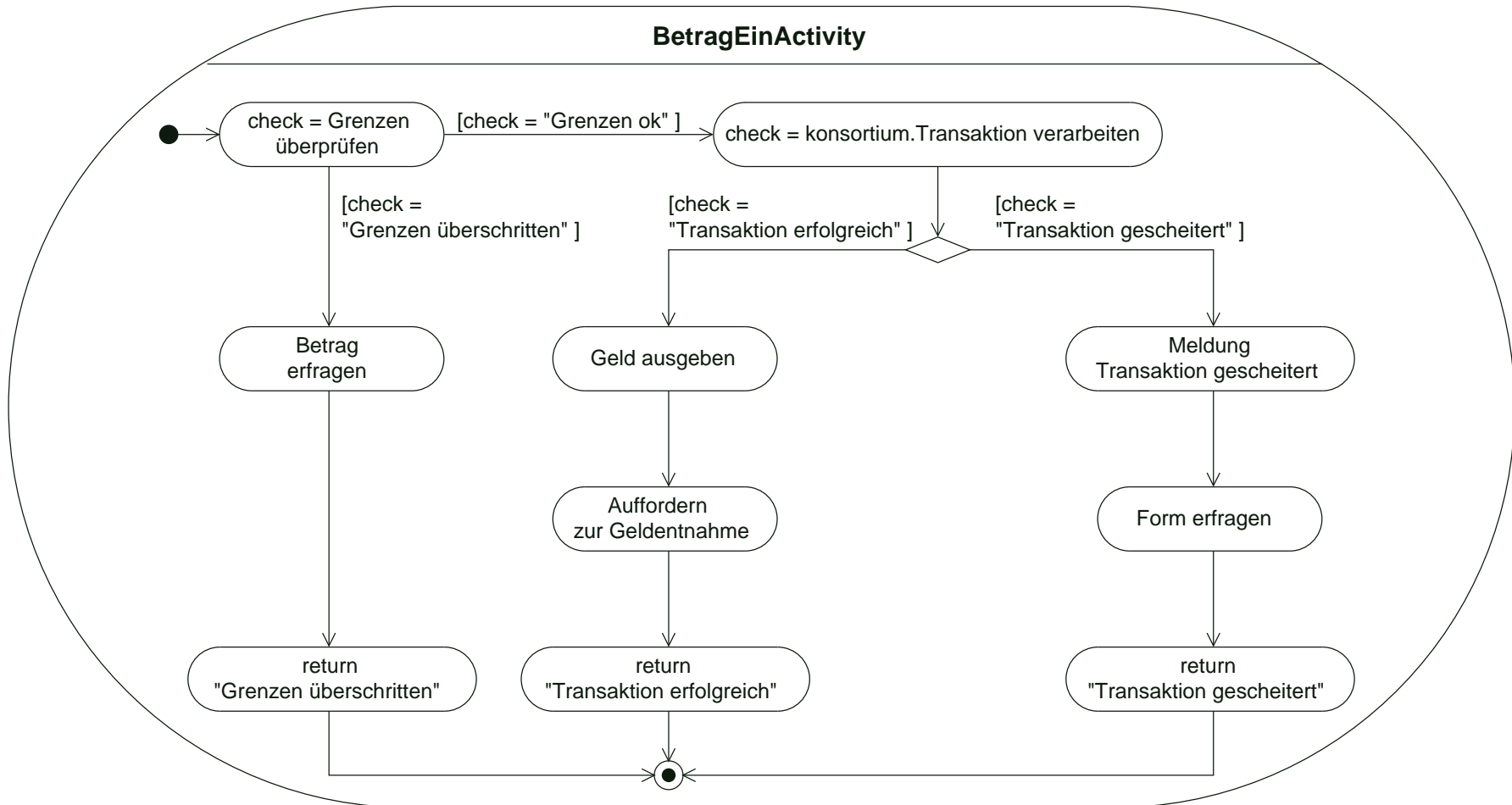
1. Wähle SD für das Primärszenario von "Geld am ATM abheben"
2. Betrachte die Lebenslinie des Objekts ":ATM"
3. Kette von Zuständen und Transitionen bilden mit einem Aktions- oder Aktivitätszustand nach jeder Benutzerinteraktion
4. Schleife bei Zustand "Bereit"
5. SD für "Karte gesperrt" integrieren,
SD für "Transaktion gescheitert" integrieren
6. Aktivitätsdiagramme für die Aktivitätszustände "KarteEinActivity",
"GeheimzahlEinActivity", "BetragEinActivity", "Abschluss" konstruieren
7. Sekundärszenarien "Karte nicht lesbar", "Falsche Geheimzahl", "Falsche Bankleitzahl", "Grenzen überschritten" integrieren
8. Bedingungen einfügen bei den von o.g. Aktivitätszuständen ausgehenden Transitionen und das bestehende Zustandsdiagramm verfeinern.

Zustandsdiagramm für ATM (nach den Schritten 1-5)

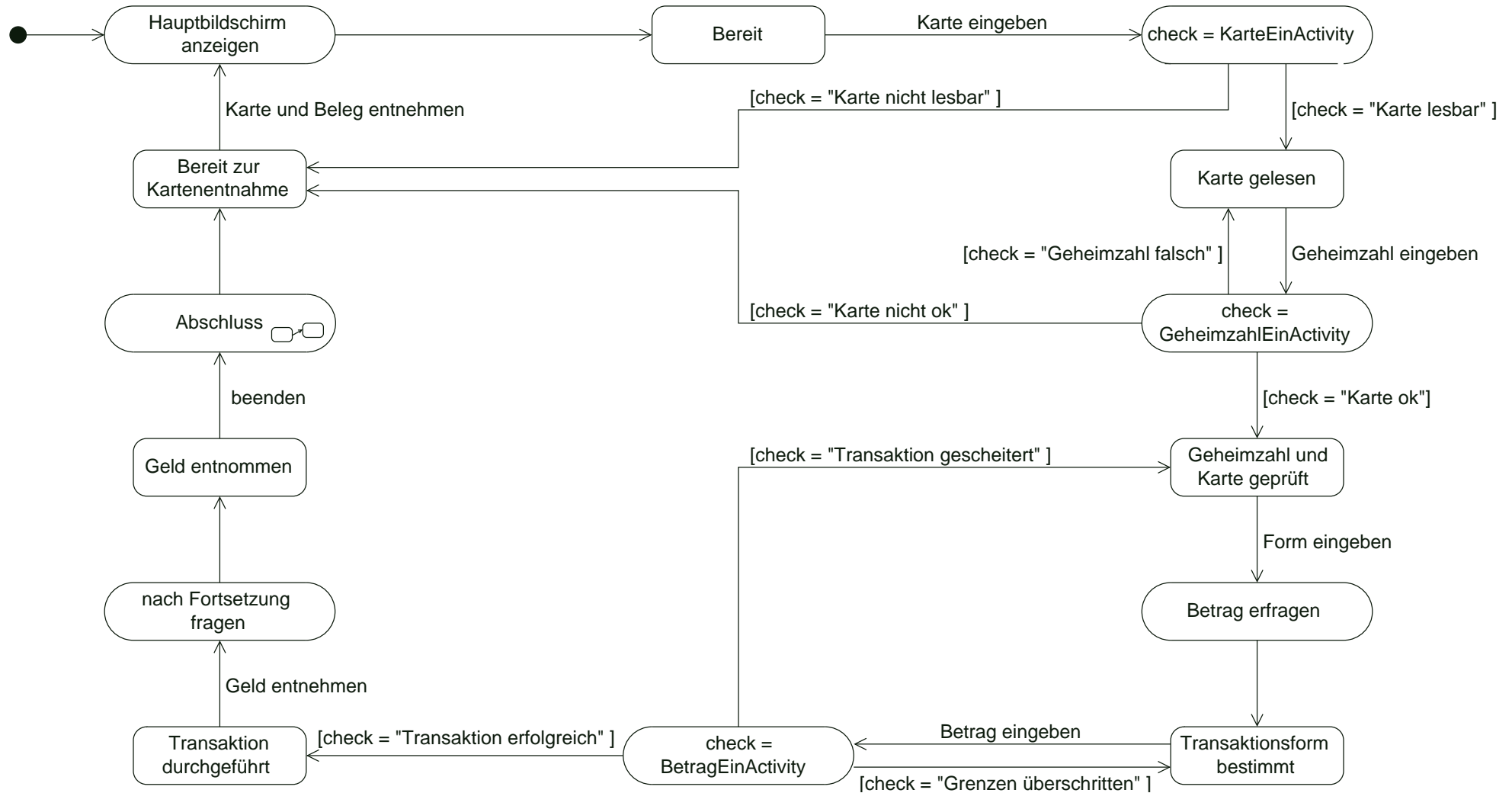






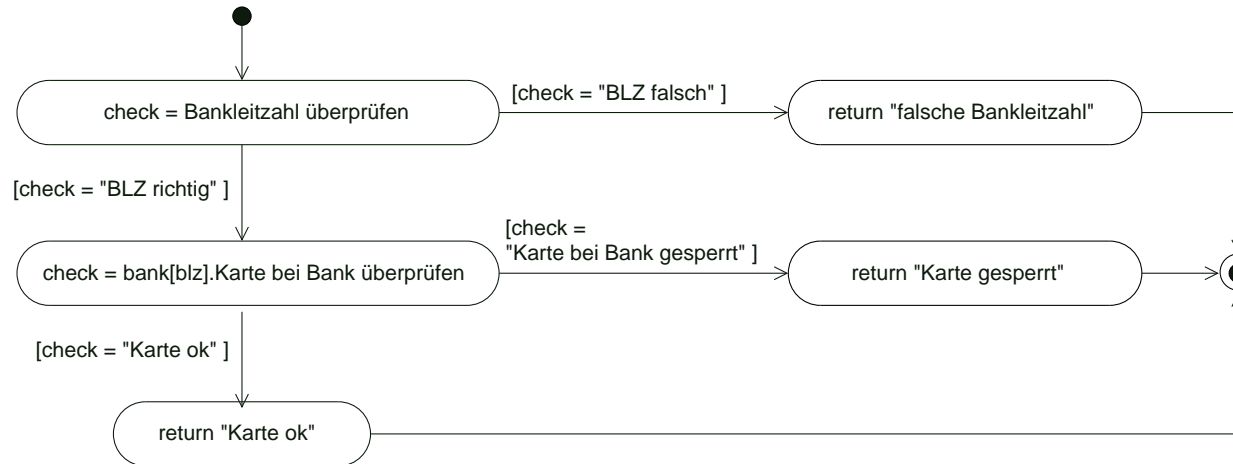


Zustandsdiagramm für ATM (nach den Schritten 6-8)

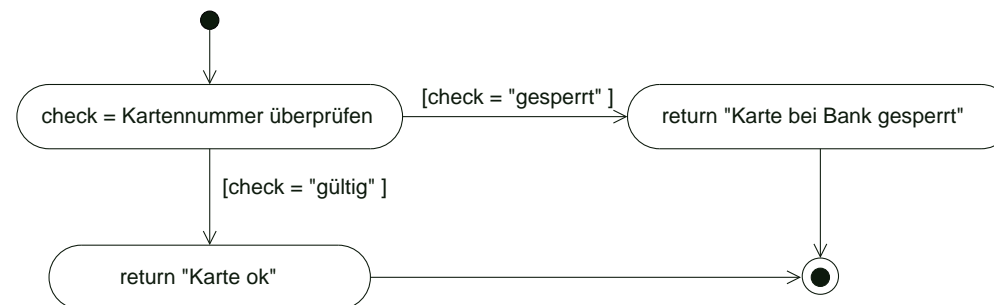


Aktivitätsdiagramme für komplexe Operationen der Klassen Konsortium und Bank

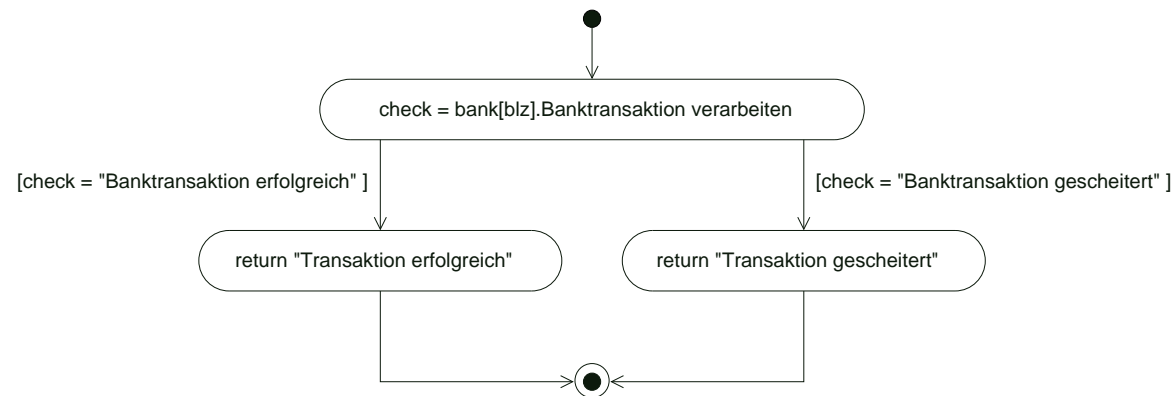
Klasse Konsortium, Aktivitätsdiagramm für "Karte überprüfen"



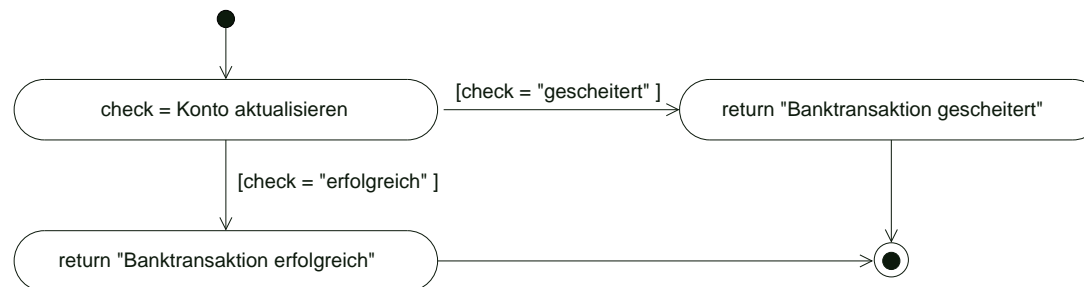
Klasse Bank, Aktivitätsdiagramm für "Karte bei Bank überprüfen"



Klasse Konsortium, Aktivitätsdiagramm für "Transaktion verarbeiten"



Klasse Bank, Aktivitätsdiagramm für "Banktransaktion verarbeiten"



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 i.a. Aktions- und Aktivitätszustände eingebunden.
- Aktivitätsdiagramme werden für komplexe Operationen erstellt.