LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Übung 1 – Einführung, Wiederholung

Methoden des Software Engineering
WS 2012/13

Christian Kroiß

Inhalt heute

- Organisatorisches & Ablauf der Übung
- Auffrischung Vorlesung Softwaretechnik:
  - Aktivitäten bei der Softwareentwicklung
  - UML
- Demo/Einführung: Papyrus UML für Eclipse

Organisatorisches

Zweiter Übungstermin

- Geplant Dienstag 16-19 Uhr
- Raum ist beantragt

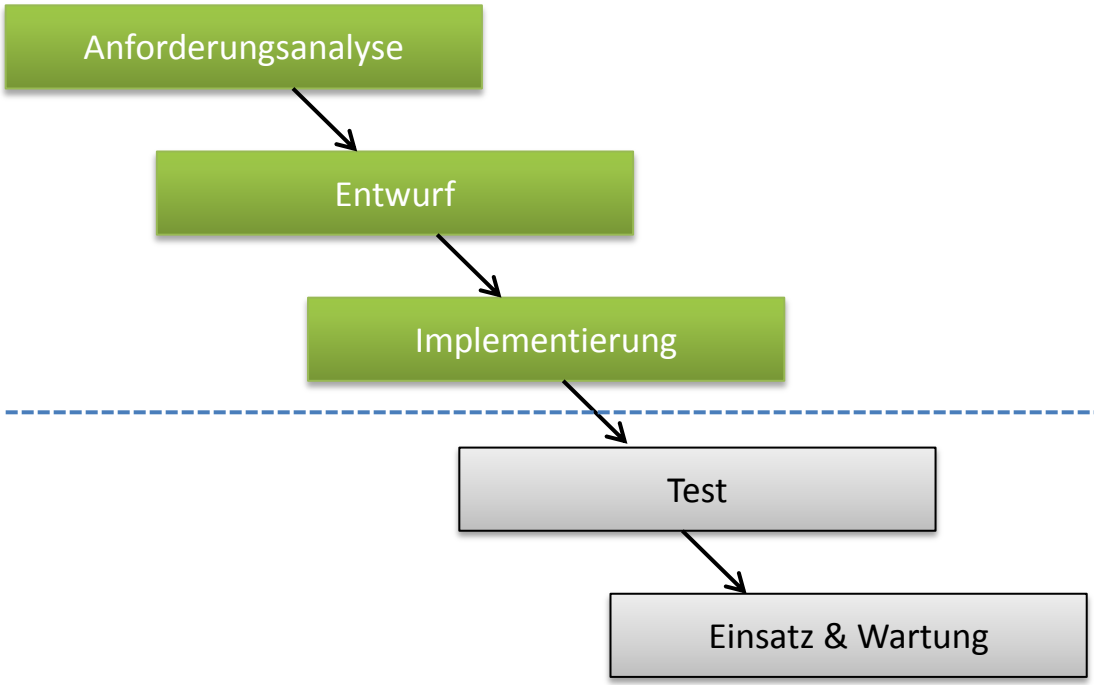
Kurztests

- 4 Kurztests – 1 nach jedem Block
  - (Vorläufige) Termine ab heute auf Homepage
  - Dauer: ca. 10 Minuten
  - Ein „Freischuss“ = nur die besten 3 zählen
  - Ergebnis der Kurztests macht 30 % der Gesamtnote aus
- Bewertete Prüfungsleistung: 90 Minuten Klausur + Kurztests
- Nachholklausur: 120 Minuten
  - Man braucht keine Kurztests, dafür eine/mehrere Aufgaben mit entsprechendem Umfang zusätzlich
  - Zuvor erreichte Ergebnisse aus den Kurztests können bei triftigem Grund für das Fehlen bei der Erstklausur eingebracht werden.

Bitte (noch mal) anschauen:

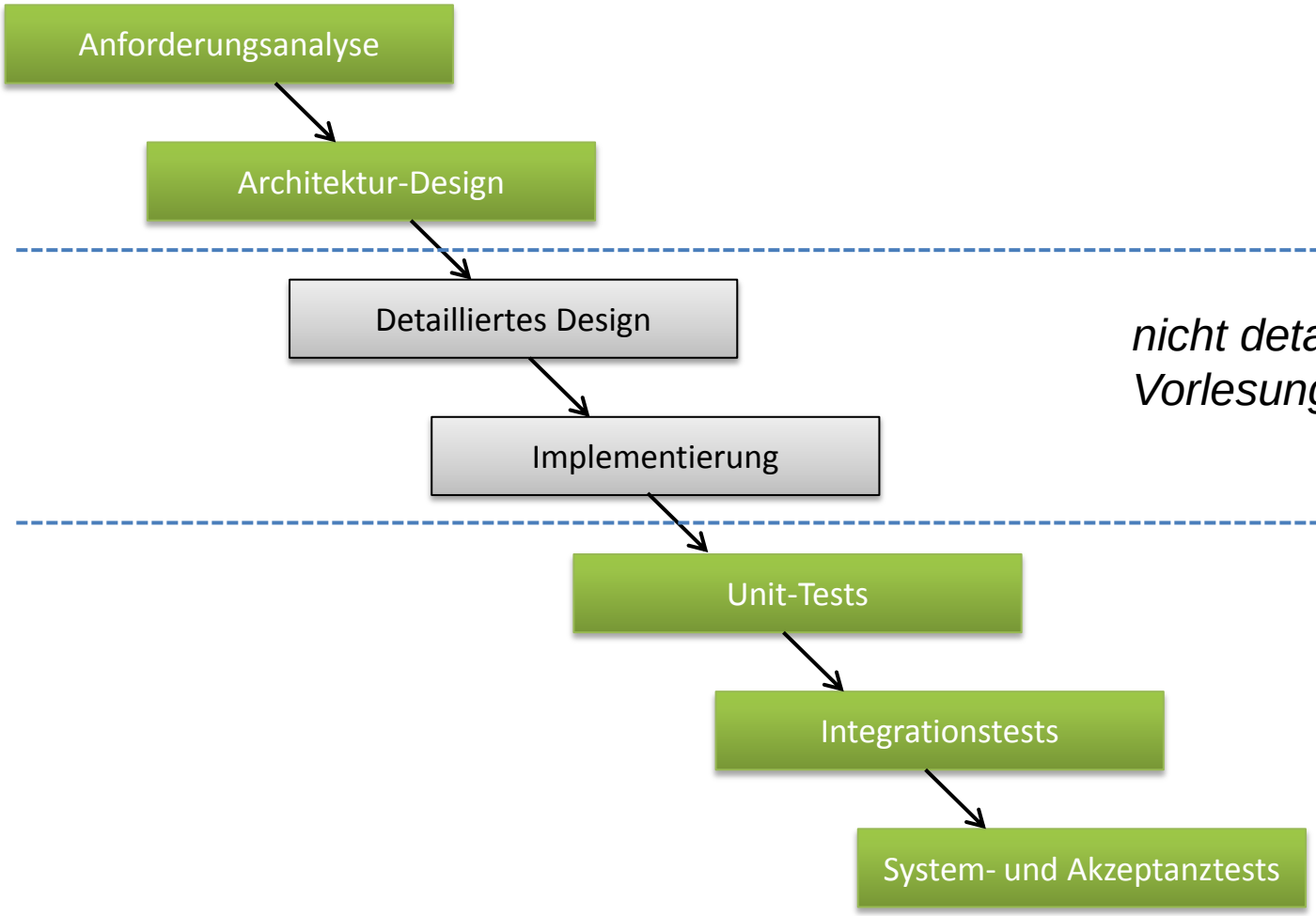
http://www.pst.ifi.lmu.de/Lehre/wise-10-11/swtechnik/materialien

Wasserfall-Modell (in Vorlesung Softwaretechnik)



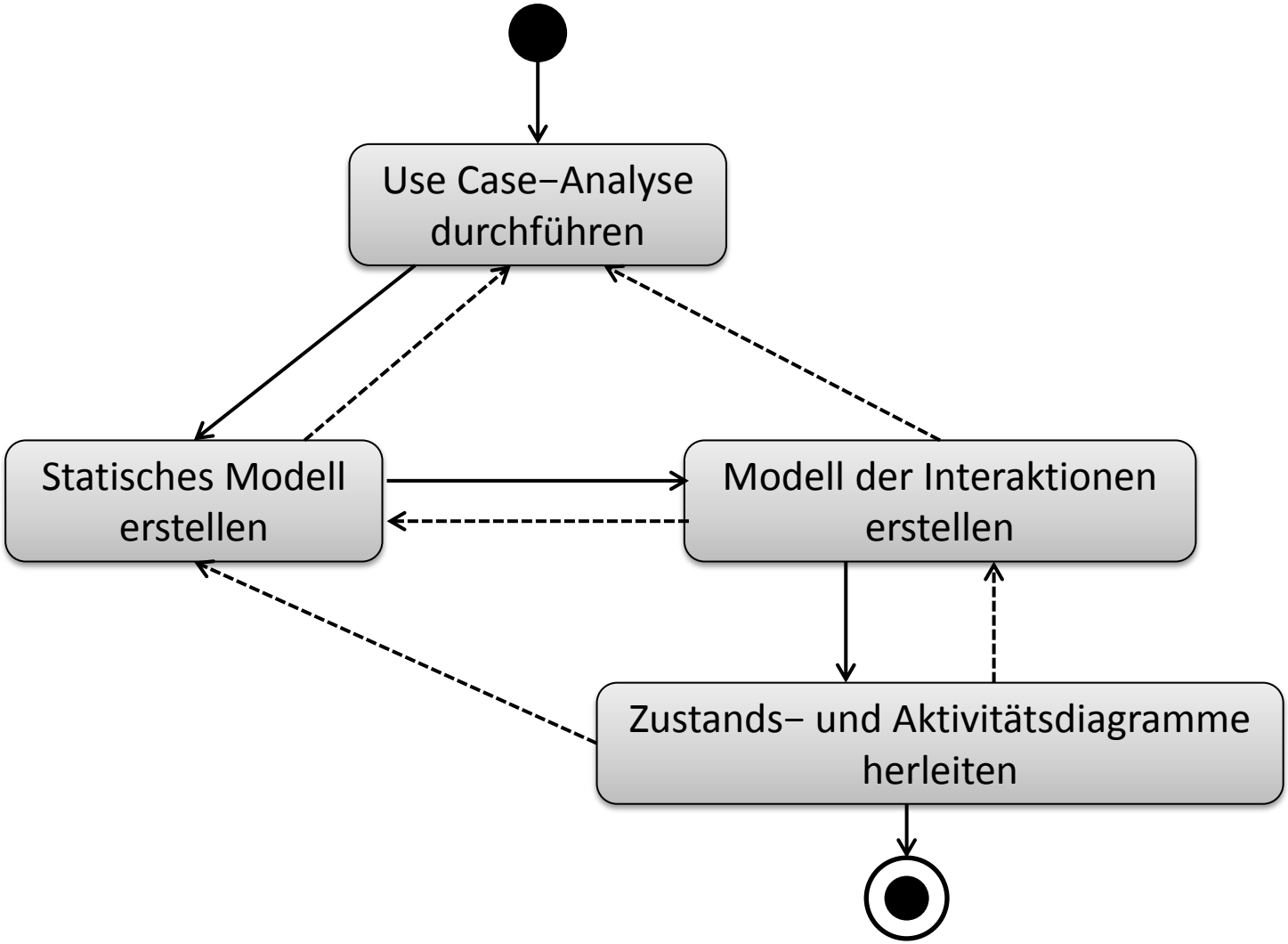
nicht detailliert in Vorlesung Softwaretechnik behandelt

Wasserfall-Modell (aus Sicht dieser Vorlesung)



nicht detailliert in dieser Vorlesung behandelt

Objektorientierte Analyse (Vorlesung Softwaretechnik)



Frage: was ist das Use-Case-Modell?

Antwort:

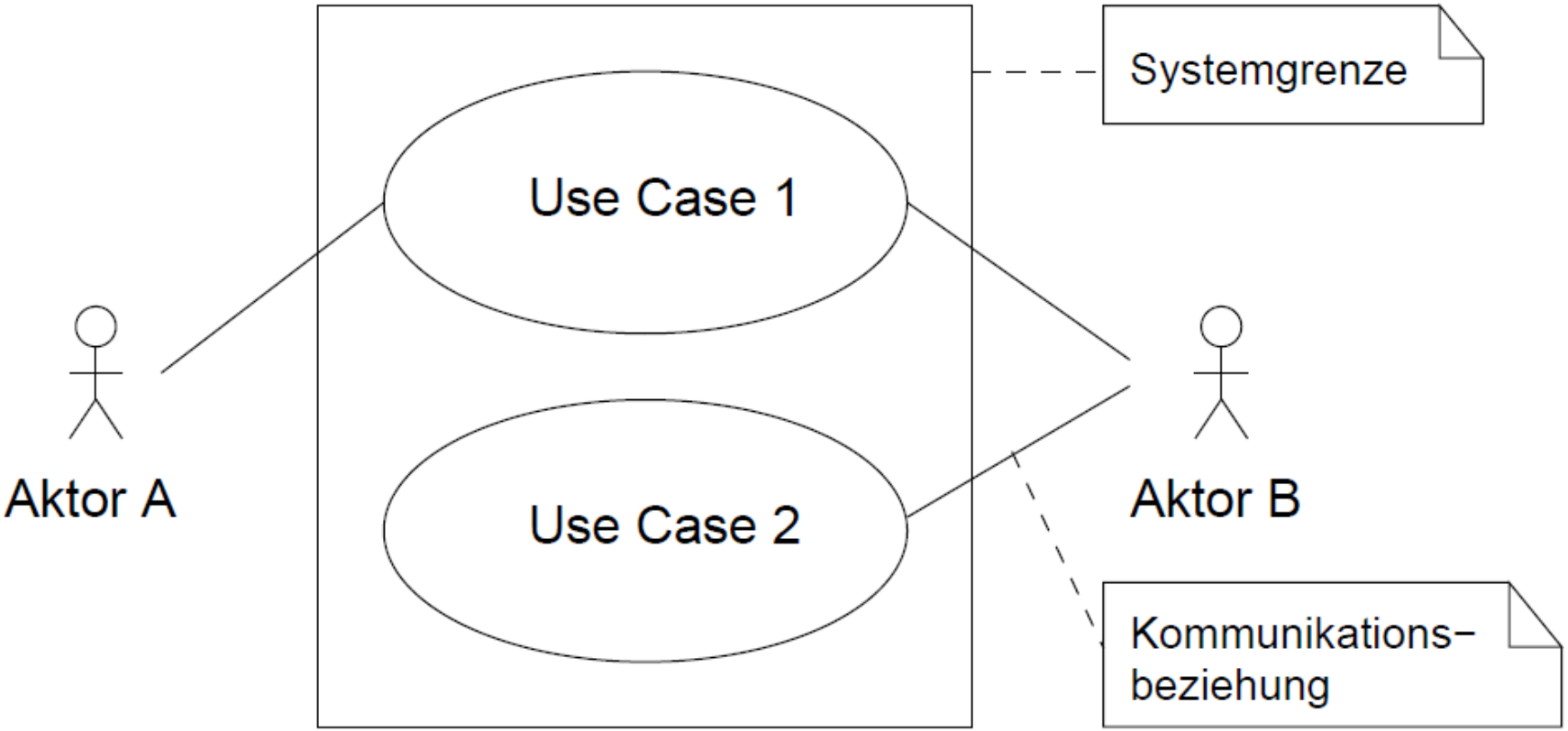
- Besteht aus Aktoren und Use Cases (Anwendungsfällen)
- Beschreibt eine externe Sicht auf das System

Frage: was ist ein Anwendungsfall?

Antwort:

- 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.

Use-Case-Diagramm:



Frage: was gehört in die Beschreibung eines Anwendungsfalles?

- Name des Anwendungsfalls
- Kurzbeschreibung
- Vorbedingung (Voraussetzung für eine erfolgreiche Ausführung des Anwendungsfalls)
- Nachbedingung (Zustand nach erfolgreicher Ausführung)
- Beschreibung Standardablauf (Primärszenario)
- Sekundärszenarien bei Fehlerfällen und Optionen

Beispiel: Use Case

Use case name: View Alarms

Summary: The monitoring operator views outstanding alarms.

Actor: Monitoring Operator

Main sequence:

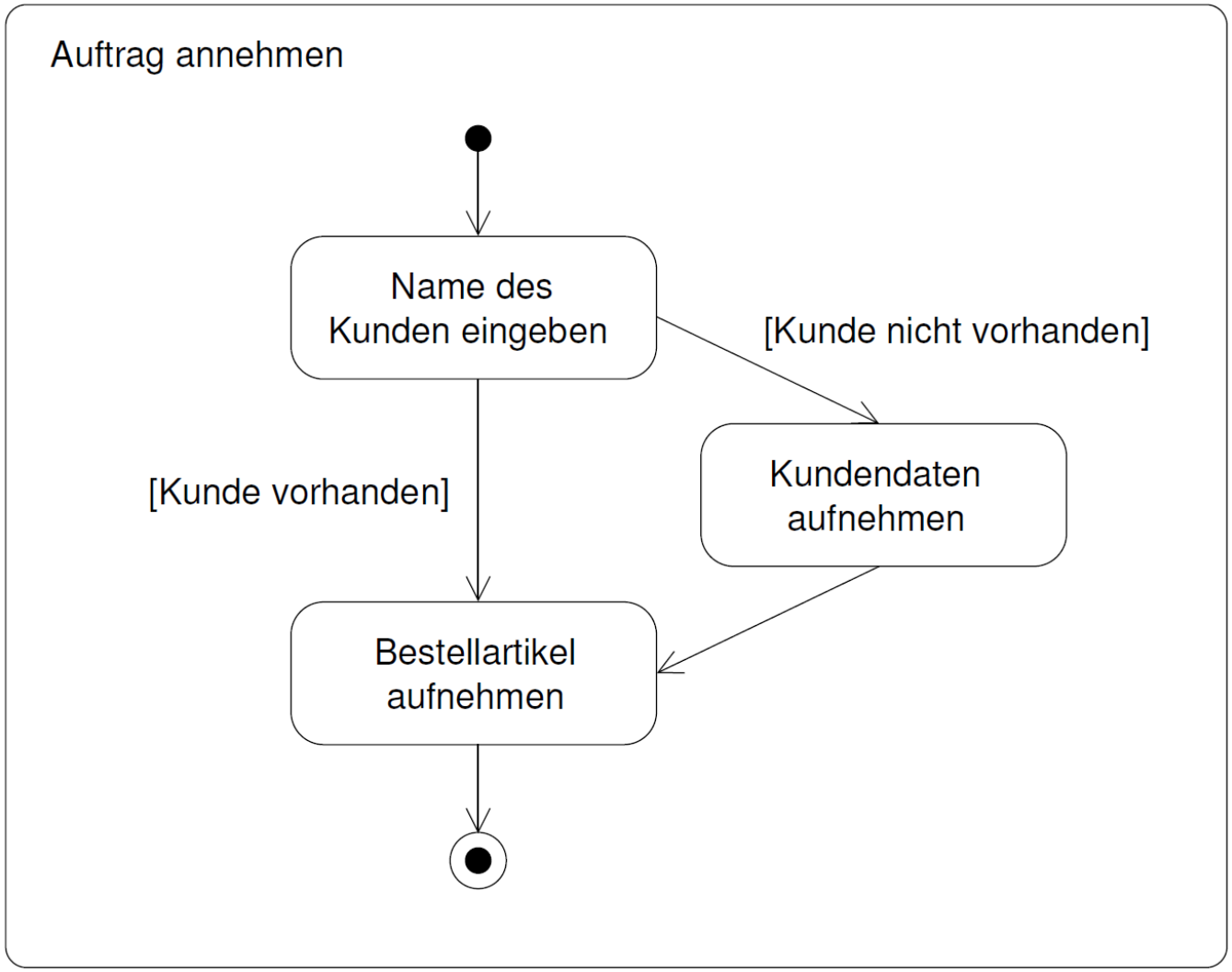
1. The monitoring operator requests to view the outstanding alarms.
2. The system displays the outstanding alarms. For each alarm, the system displays the name of the alarm, alarm description, location of alarm, and severity of alarm (high, medium, low).

Alternative sequences:

Step 2: Emergency situation. System displays emergency warning message to operator.

Use Cases

Gegebenenfalls lassen sich Use Cases gut durch Aktivitätsdiagramme darstellen



Beispiel: ATM

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.

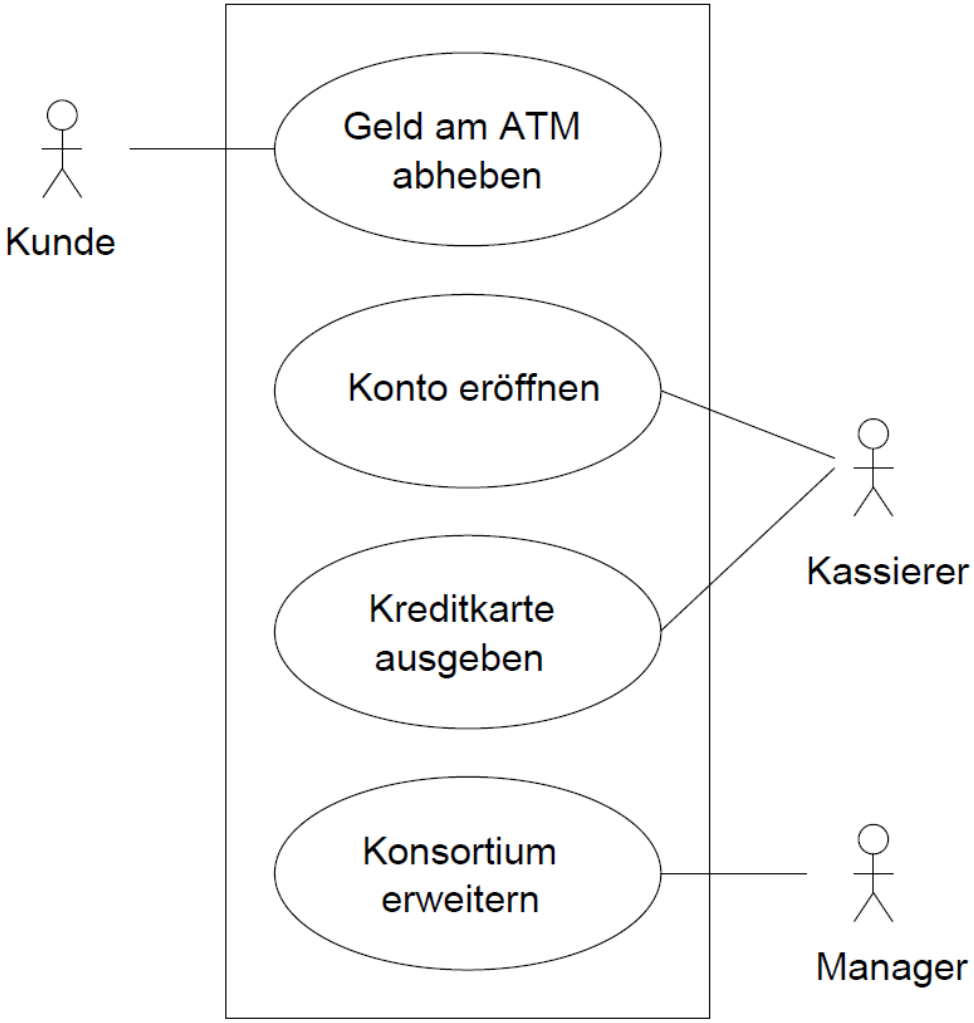
Das System erfordert geeignete Aufzeichnungsmöglichkeiten und Sicherheitsmaßnahmen. Das System muss parallele Zugriffe auf das gleiche Konto korrekt abwickeln. Die Banken stellen die SW für ihre eigenen Computer selbst bereit.

Beispiel: ATM

Aktoren: Kunde, Kassierer, Manager, ...

Use Cases:

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



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.

Beispiel: ATM

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.

Beispiel: ATM

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.

Beispiel: ATM

Frage: Weitere Alternativszenarios?

- Abbruch durch den Benutzer
- Karte nicht lesbar
- Falsche Geheimzahl
- Falsche Bankleitzahl
- Grenzen überschritten
- Karte abgelaufen
- Kein Geld im ATM
- Netzwerk unterbrochen

Statisches Modell

Input

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

Ziel: Erstellung eines Klassendiagramms (noch ohne Operationen!)

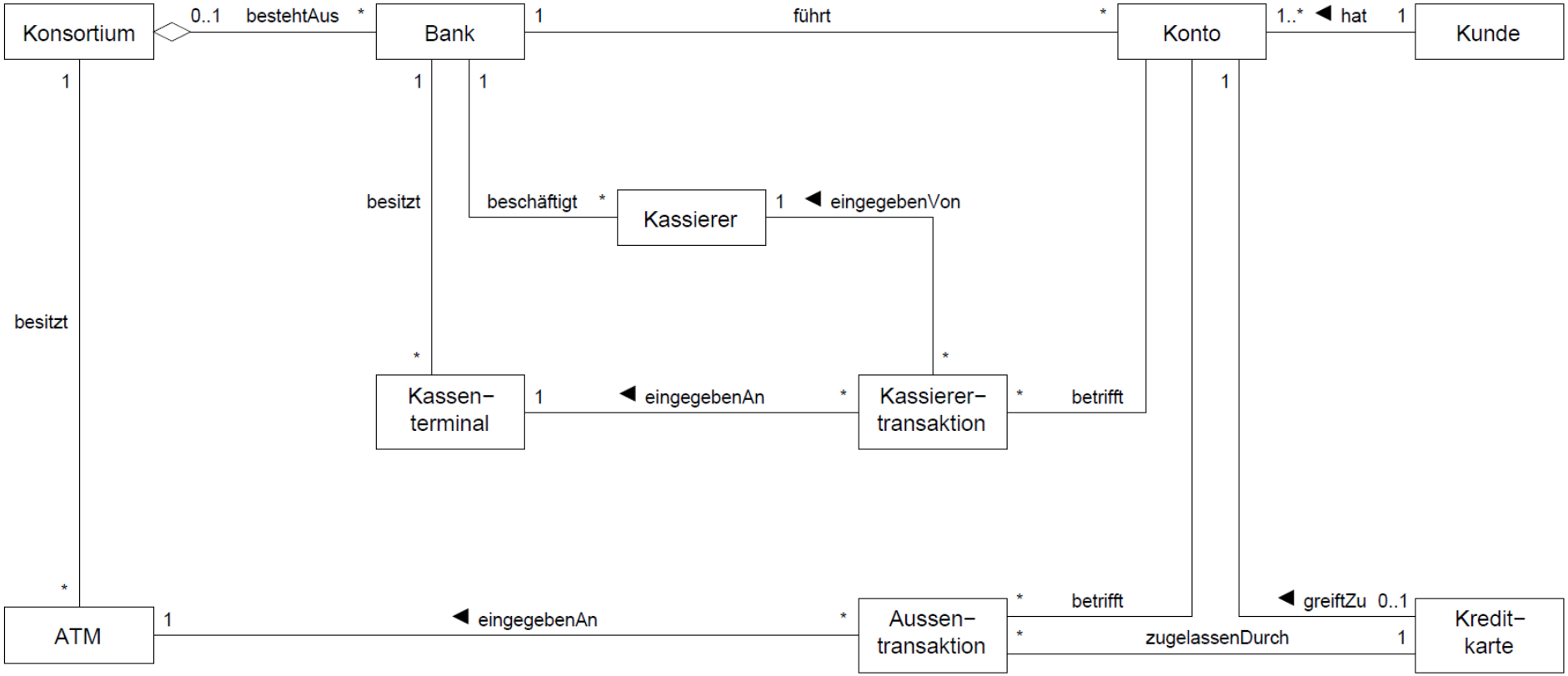
Vorgehensweise (nach OMT)

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

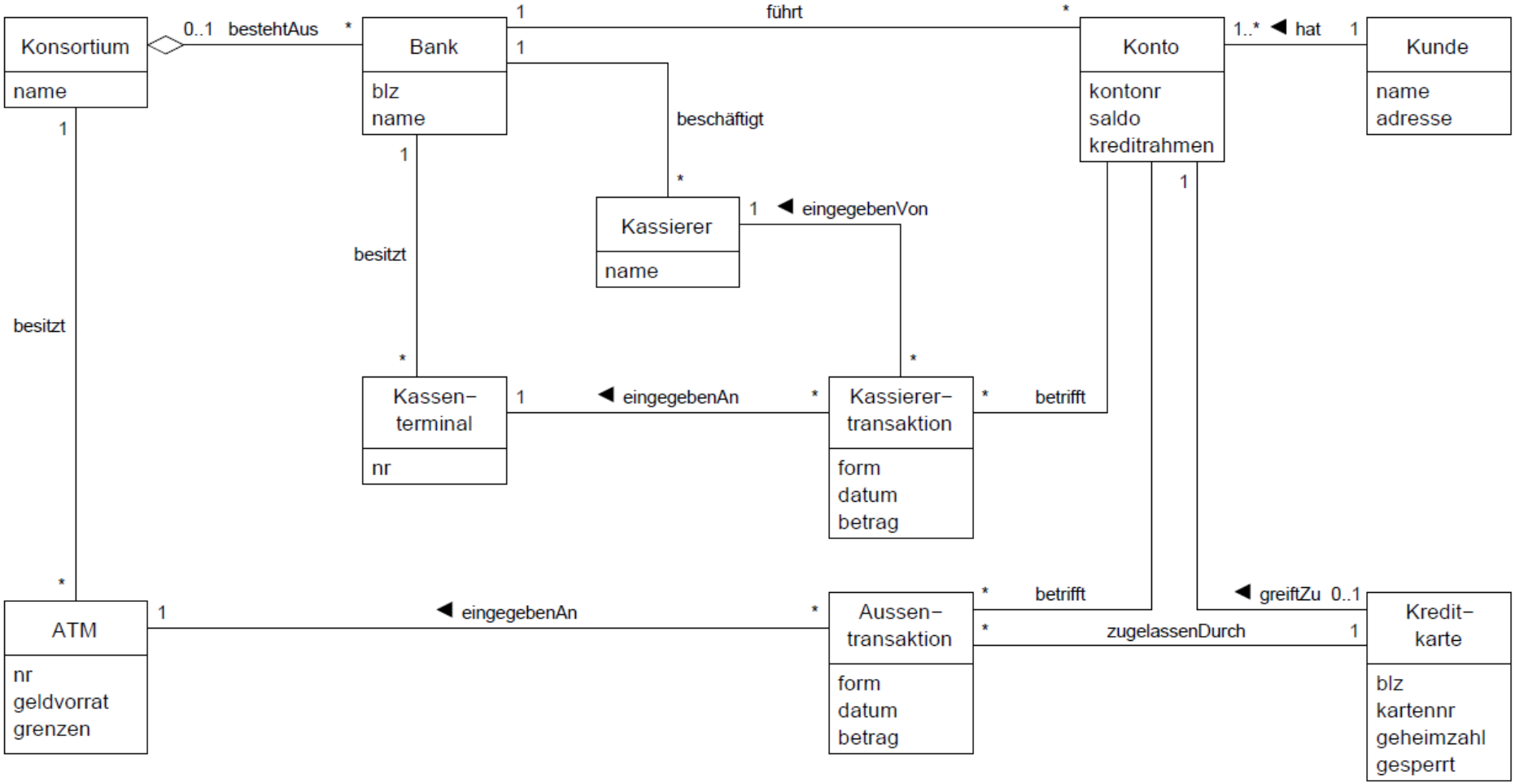
Beispiel ATM

- 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)

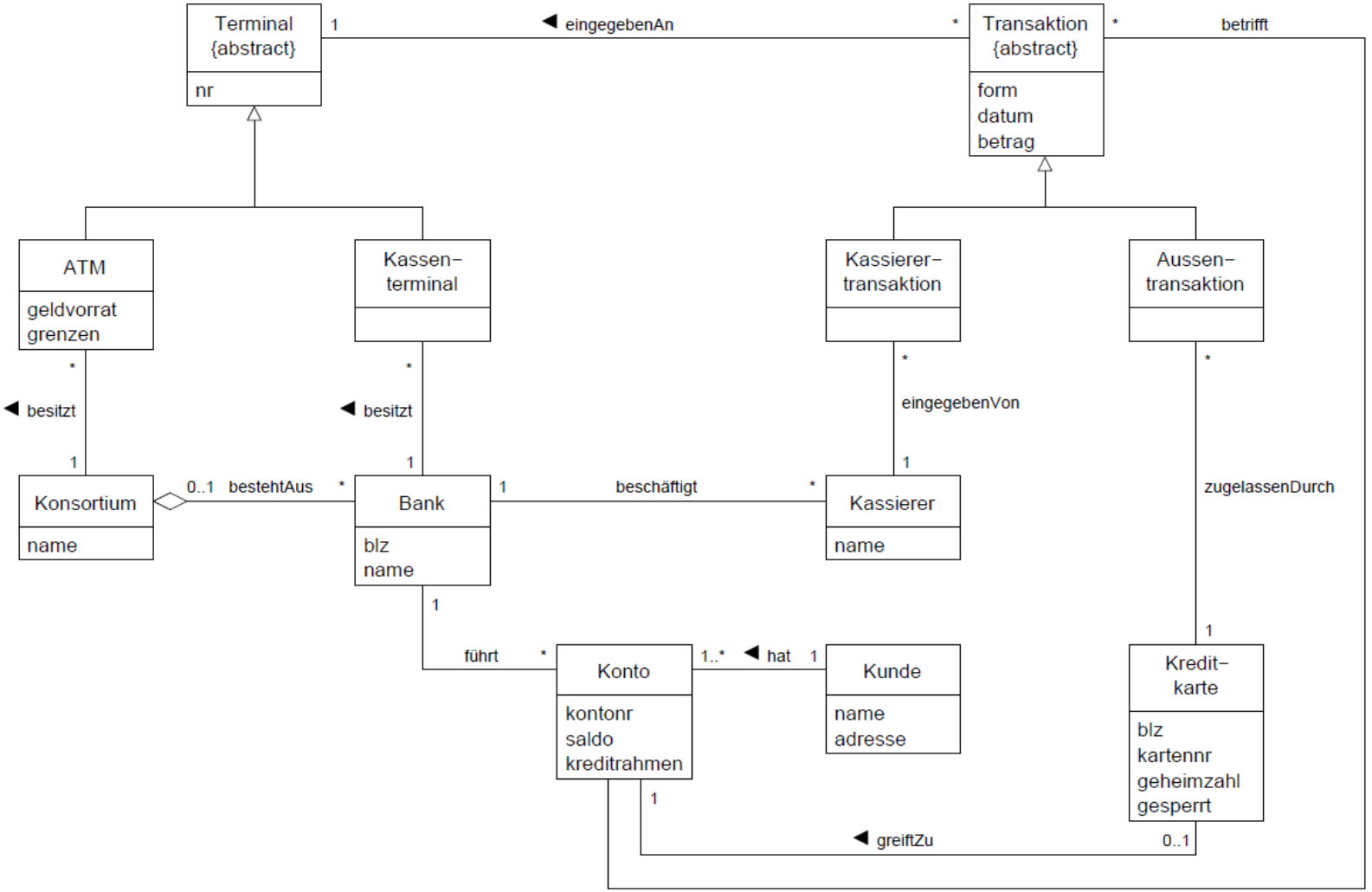
Statisches Modell Version 1



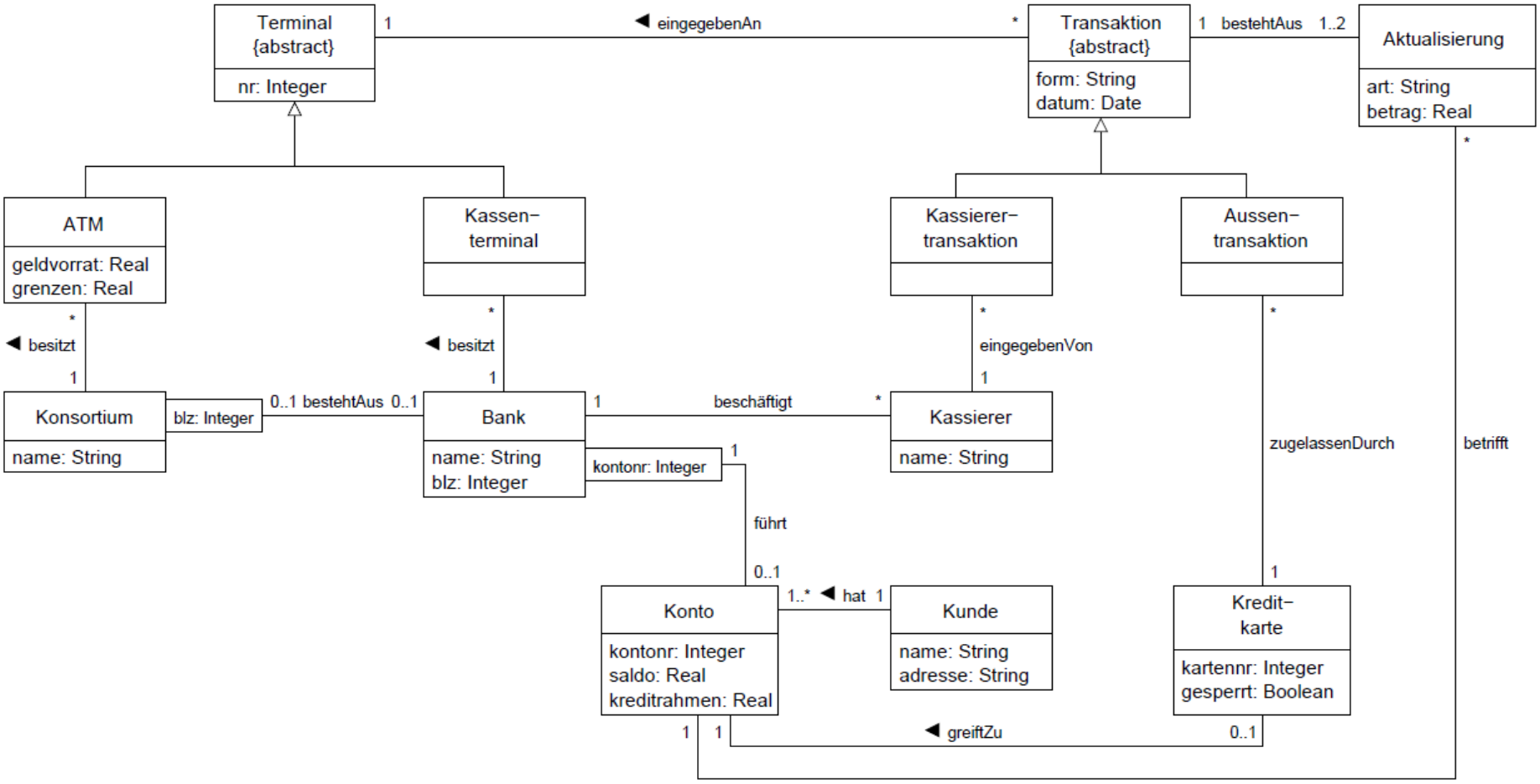
Statisches Modell mit Attributen



Statisches Modell mit Vererbung



Statisches Modell nach Überarbeitung



Entwurf von Interaktionsdiagrammen

Input

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

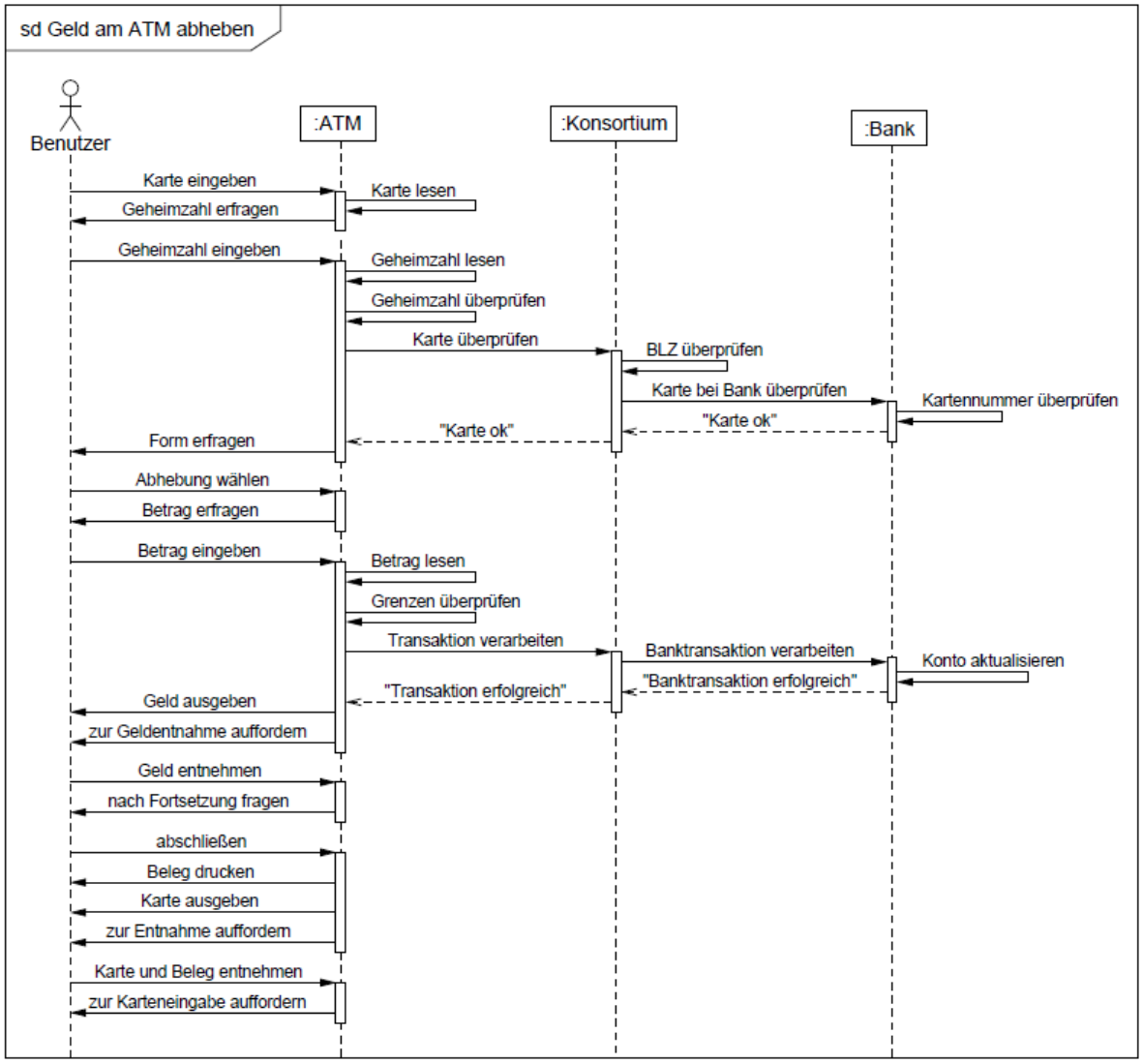
Ziel

Modellierung der Zusammenarbeit von Objekten innerhalb eines Anwendungsfalls durch Interaktionsdiagramme.

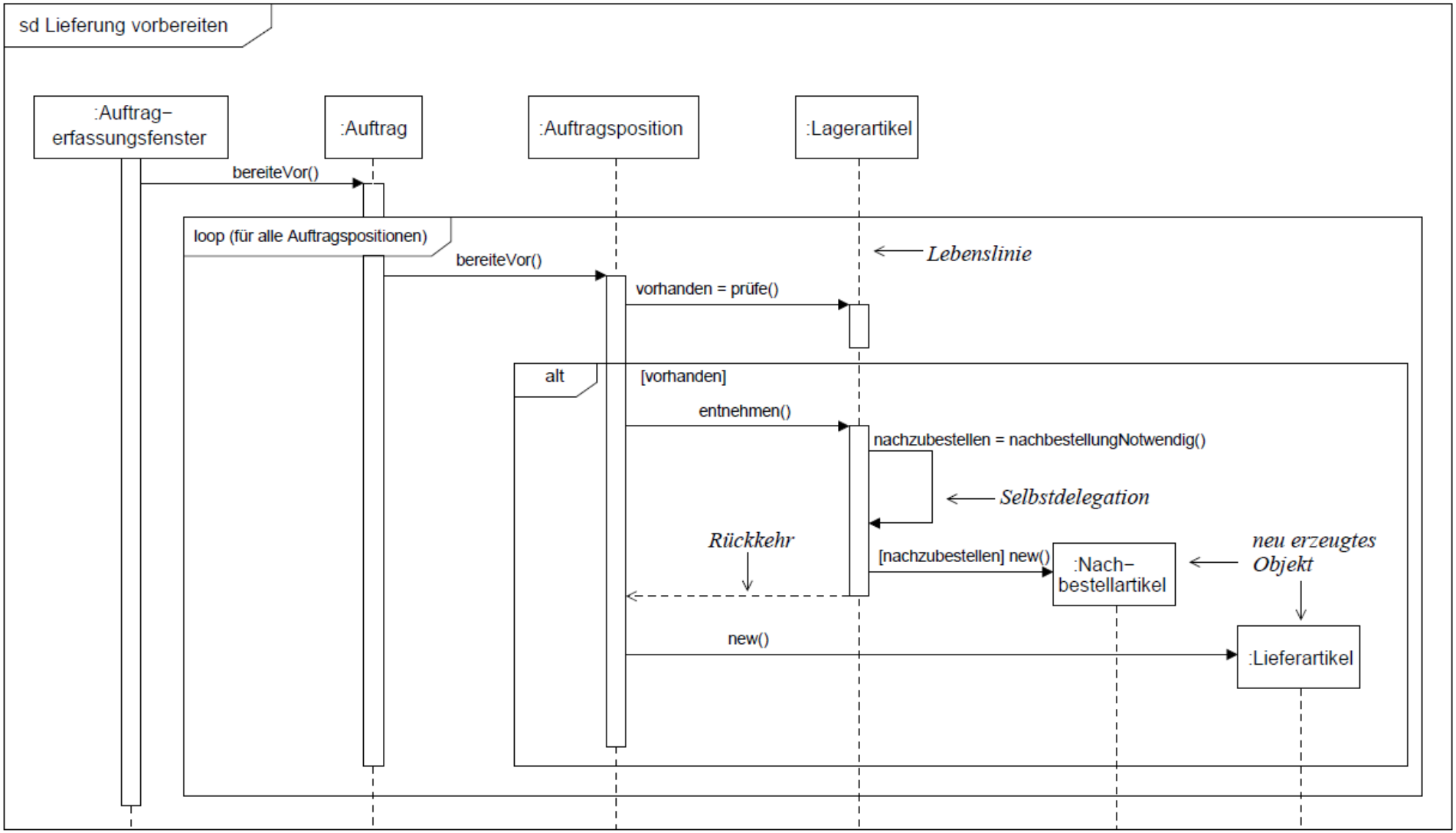
Vorgehensweise

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.

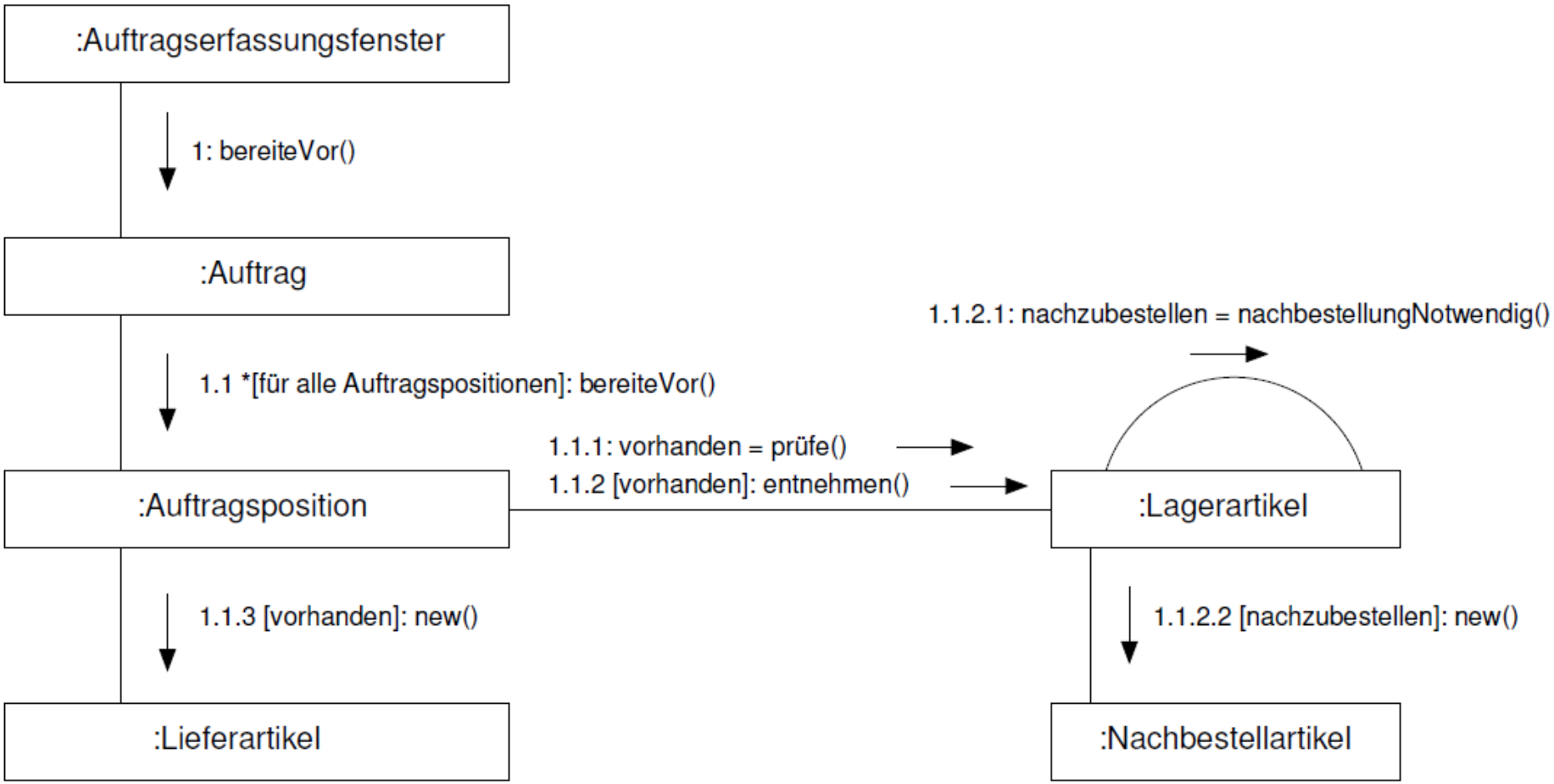
Beispiel: Sequenzdiagramm



Kontrollstrukturen in Sequenzdiagramme in UML 2.x



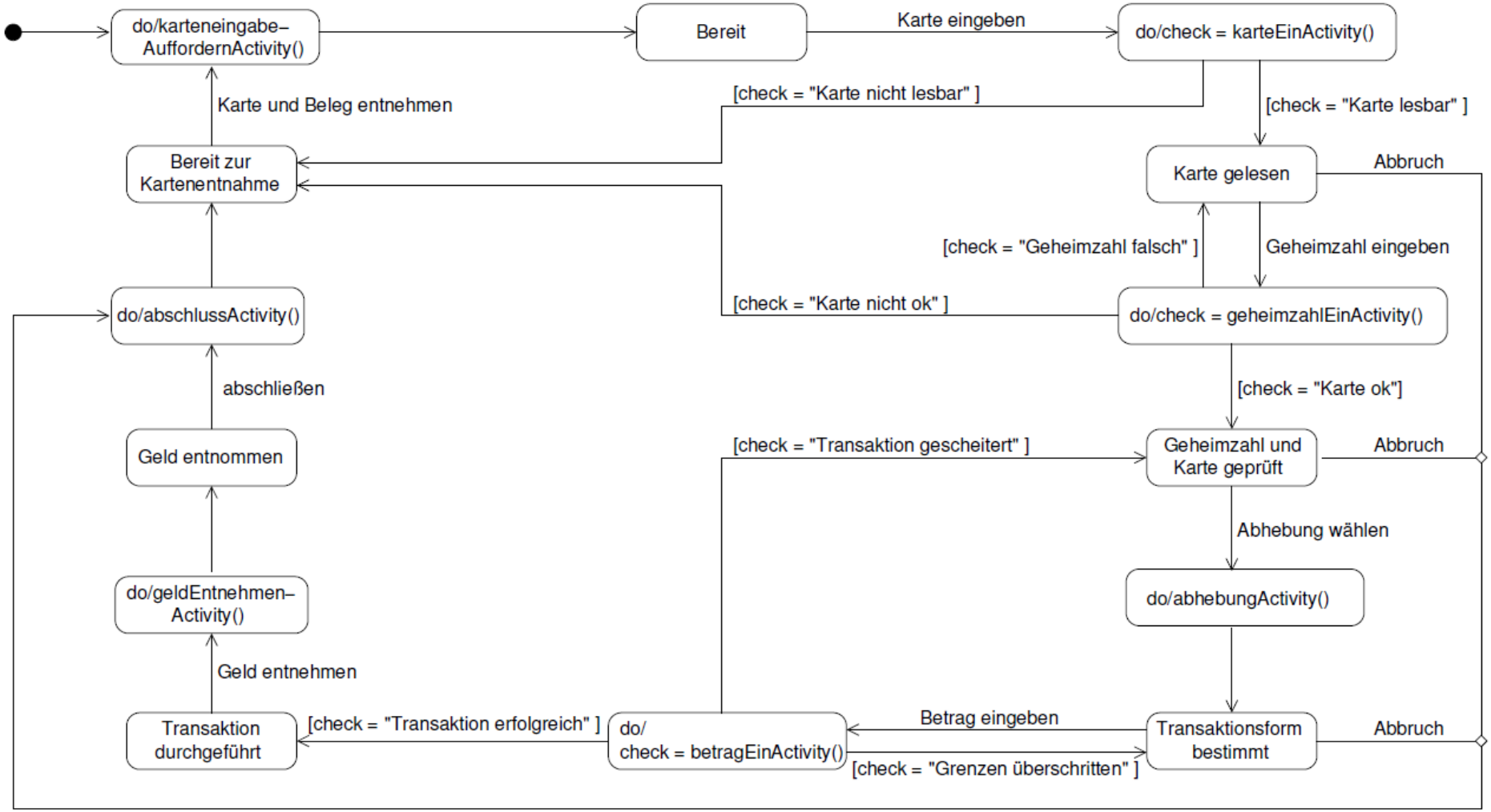
Kommunikationsdiagramme



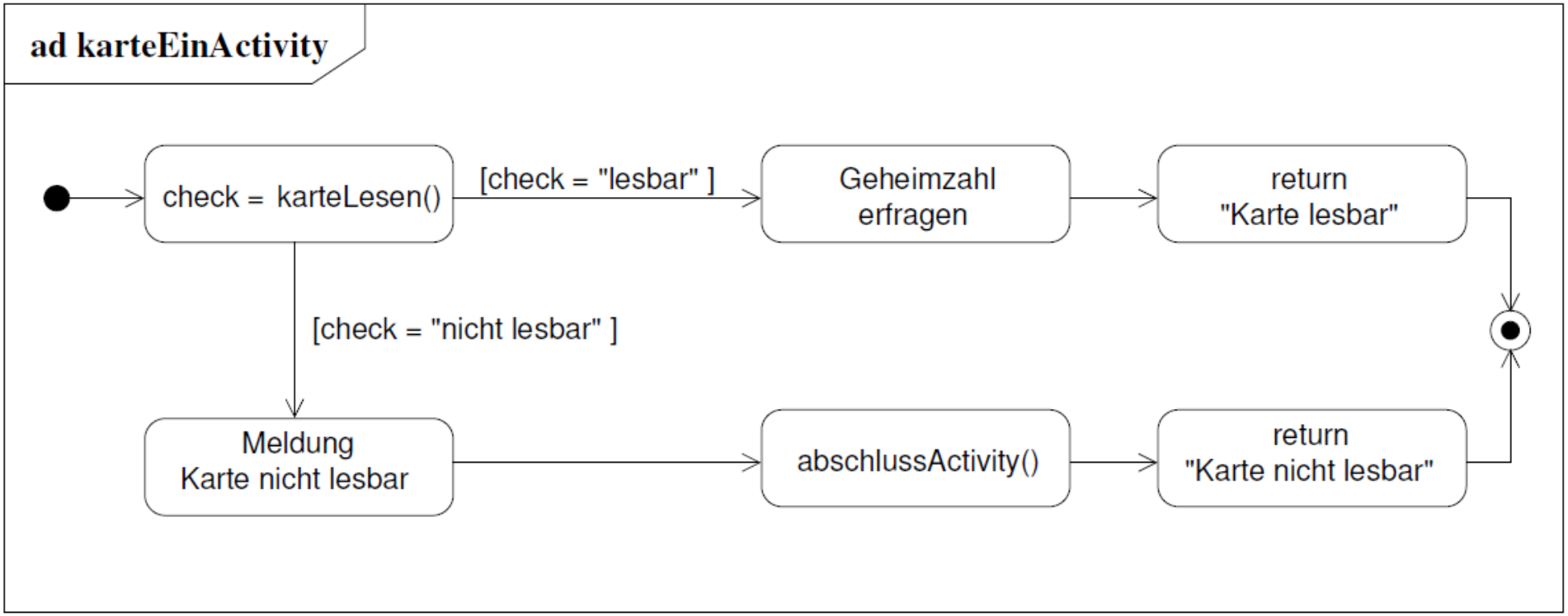
Zustandsdiagramm für ATM konstruieren

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



Aktivitätsdiagramme für Operationen (Beispiel)



Ausgangspunkt

Statisches und dynamisches Modell der objektorientierten Analyse

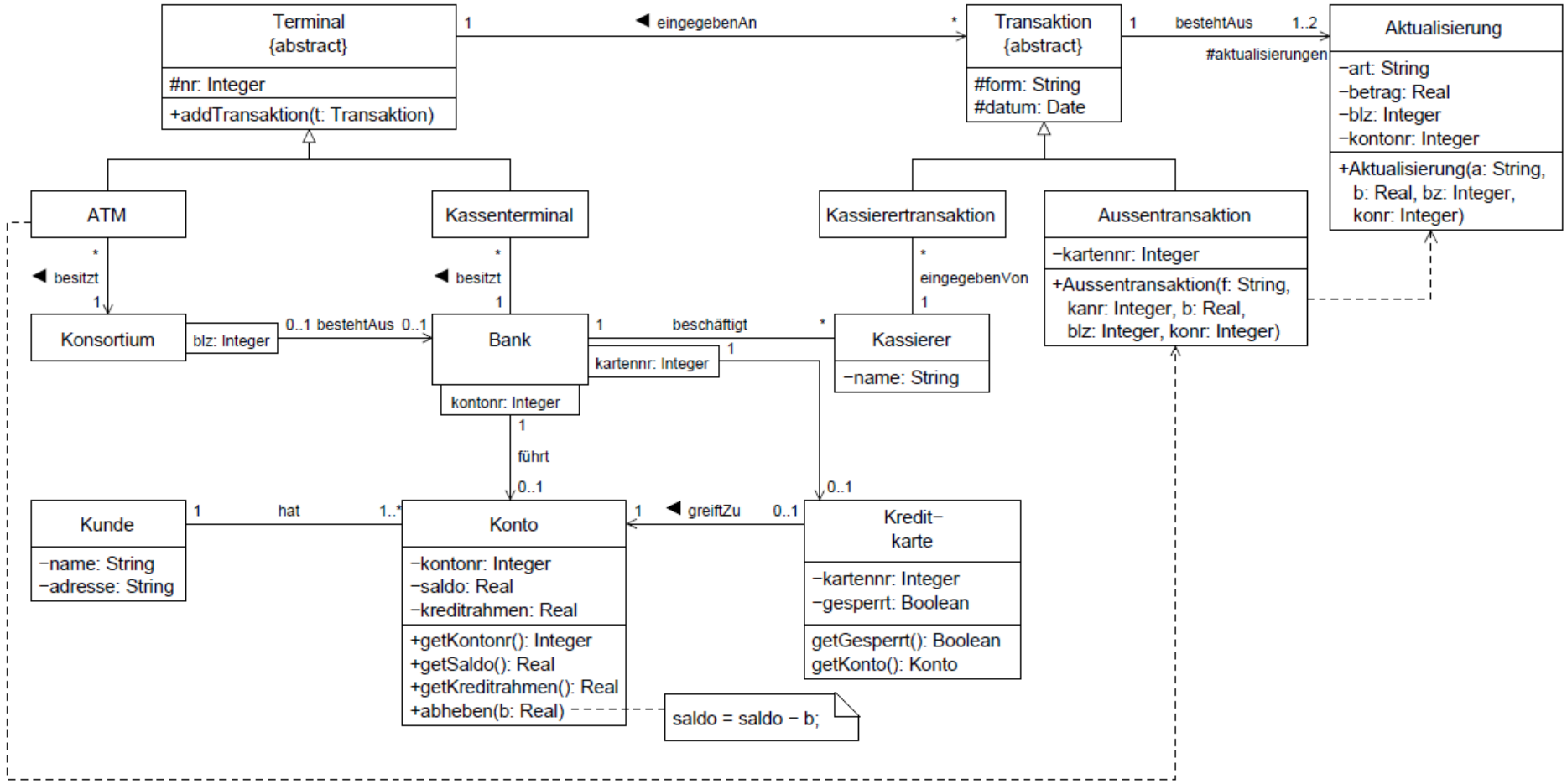
Ziel

Modell der Systemimplementierung (beschreibt **wie** die einzelnen Aufgaben gelöst werden)

Wesentliche Aufgaben

- Verfeinerung des Analysemodells durch Integration des statischen und dynamischen Modells. Führt zum Objektentwurf.
- Einbindung in die Systemumgebung durch den Entwurf von Benutzerschnittstellen, Datenbankschnittstellen, Netzwerk-Wrappern etc.
- Konstruktion der Systemarchitektur

Klassendiagramm von ATM nach dem Objektentwurf



Klassendiagramm von ATM nach dem Objektentwurf

ATM
−geldvorrat: Real −grenzen: Real −aktKartennr: Integer −aktBLZ: Integer −aktGeheimzahl: Integer −aktKontonr: Integer
+ATM() +karteEin +abbruch +geheimzahlEin +abhebungWaehlen +betragEin +geldEntnehmen +abschliessen +karteBelegEntnehmen −karteneingabeAuffordernActivity() −karteEinActivity(): String −geheimzahlEinActivity(): String −abhebungActivity() −betragEinActivity(): String −geldEntnehmenActivity() −abschlussActivity() −karteLesen(): String −geheimzahlUeberpruefen(tgz: Integer): String −grenzenUeberpruefen(b: Real): String

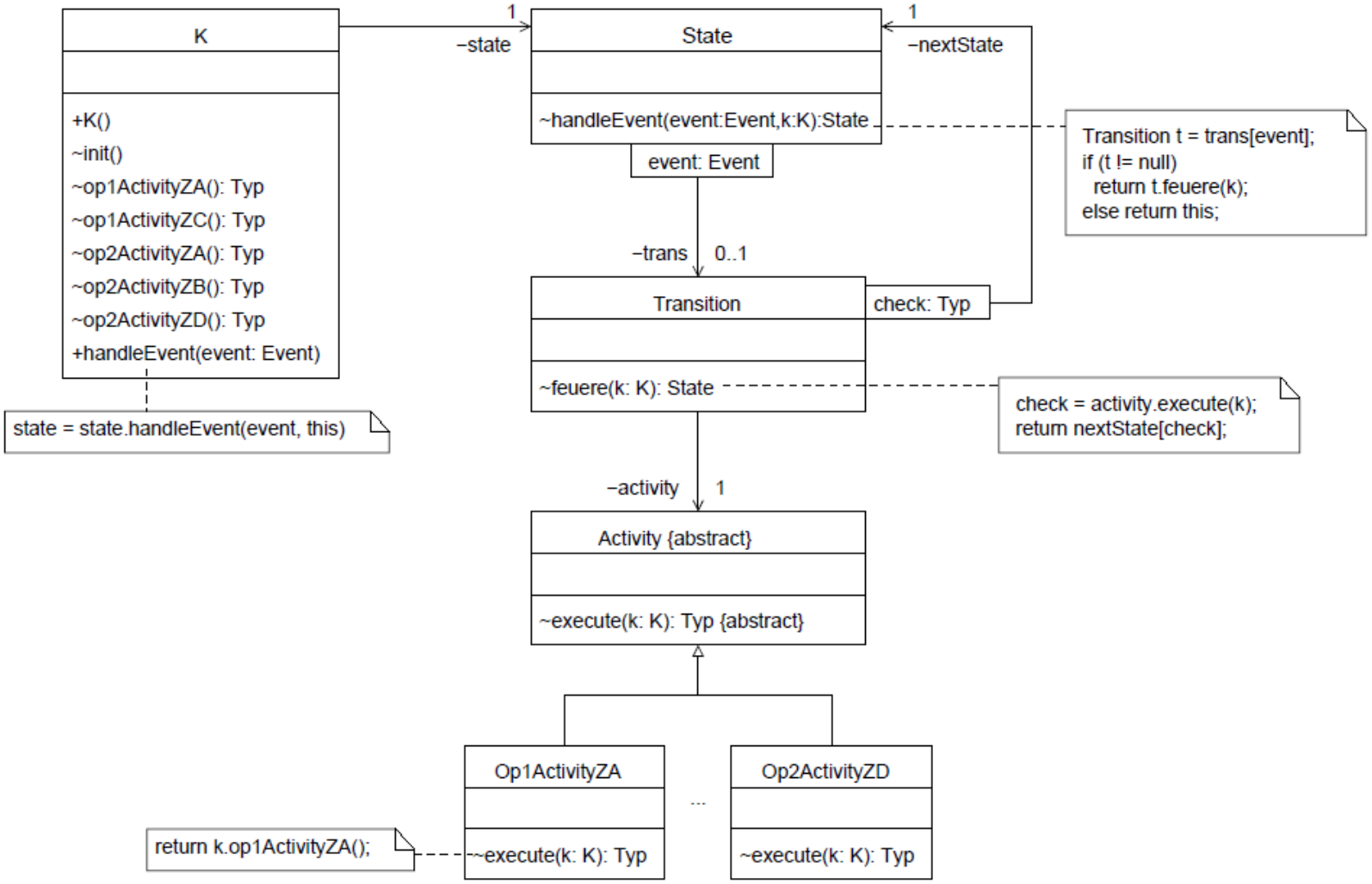
Konsortium
−name: String
+karteUeberpruefen(kartennr: Integer, blz: Integer, out kontonr: Integer): String +transaktionVerarbeiten(blz: Integer, kontonr: Integer, b: Real): String −blzUeberpruefen(blz: Integer): String

Bank
−blz: Integer −name: String
+bankKarteUeberpruefen(kartennr: Integer, out kontonr: Integer): String +bankTransaktionVerarbeiten(kontonr: Integer, b: Real): String −kartennrUeberpruefen(kartennr: Integer, out kontonr: Integer): String −kontoAktualisieren(kontonr: Integer, b: Real): String

Algorithmen aus Aktivitätsdiagrammen

```
karteEinActivity(): String {
  String check = karteLesen();
   If (check.equals("lesbar")) {
    output("Geheimzahl?");
    return "Karte lesbar";
  }
  else if (check.equals("nicht lesbar")) {
    output("Karte nicht lesbar!");
    abschlussActivity();
    return "Karte nicht lesbar";
  }
else return "Error";
}
```

Realisierung von Zustandsmaschinen



Systemarchitektur

Die Systemarchitektur beschreibt die Gesamtstruktur des SW-Systems durch Angabe von Subsystemen und von Beziehungen zwischen den Subsystemen (ggf. unter Verwendung von Schnittstellen).

Bemerkungen

Eine grobe Systemarchitektur wird häufig schon zu Beginn der Systementwicklung angegeben.

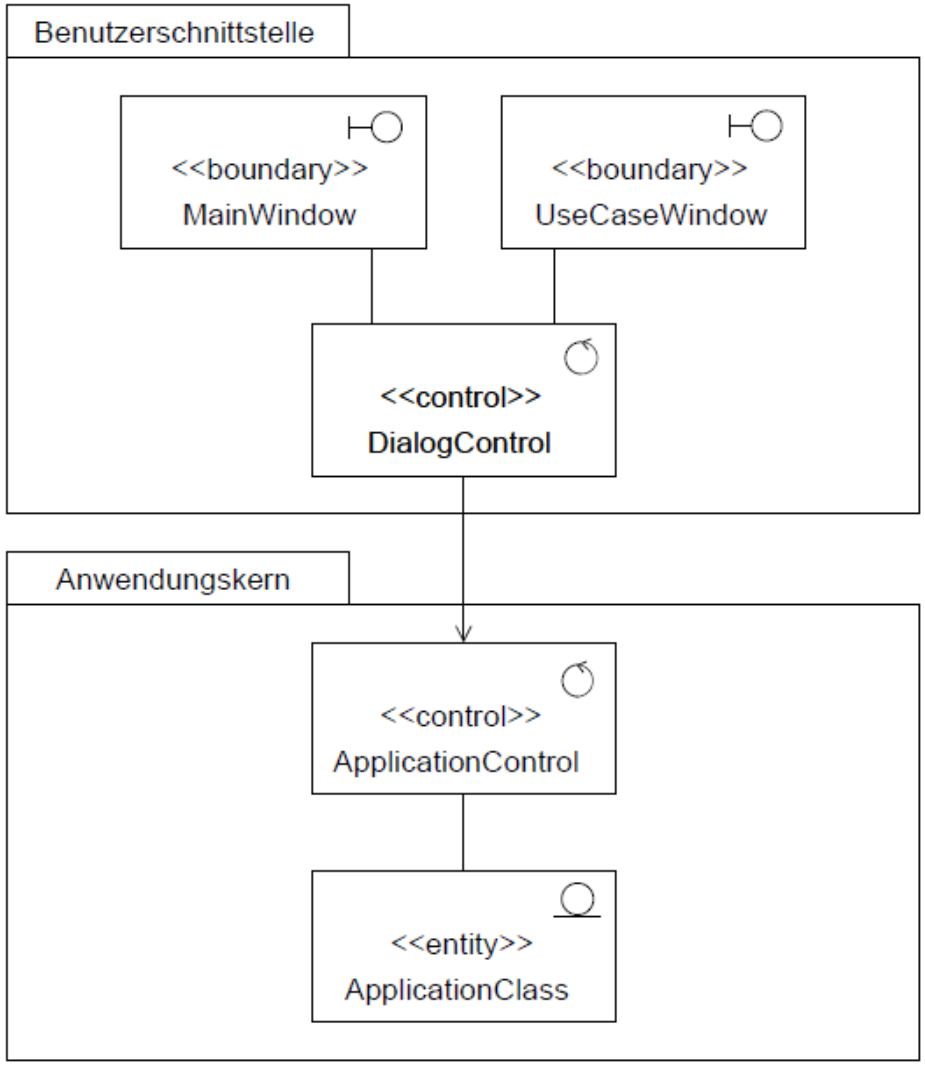
Subsysteme werden dargestellt durch Pakete oder durch Komponenten (mit Schnittstellen).

Grundregeln

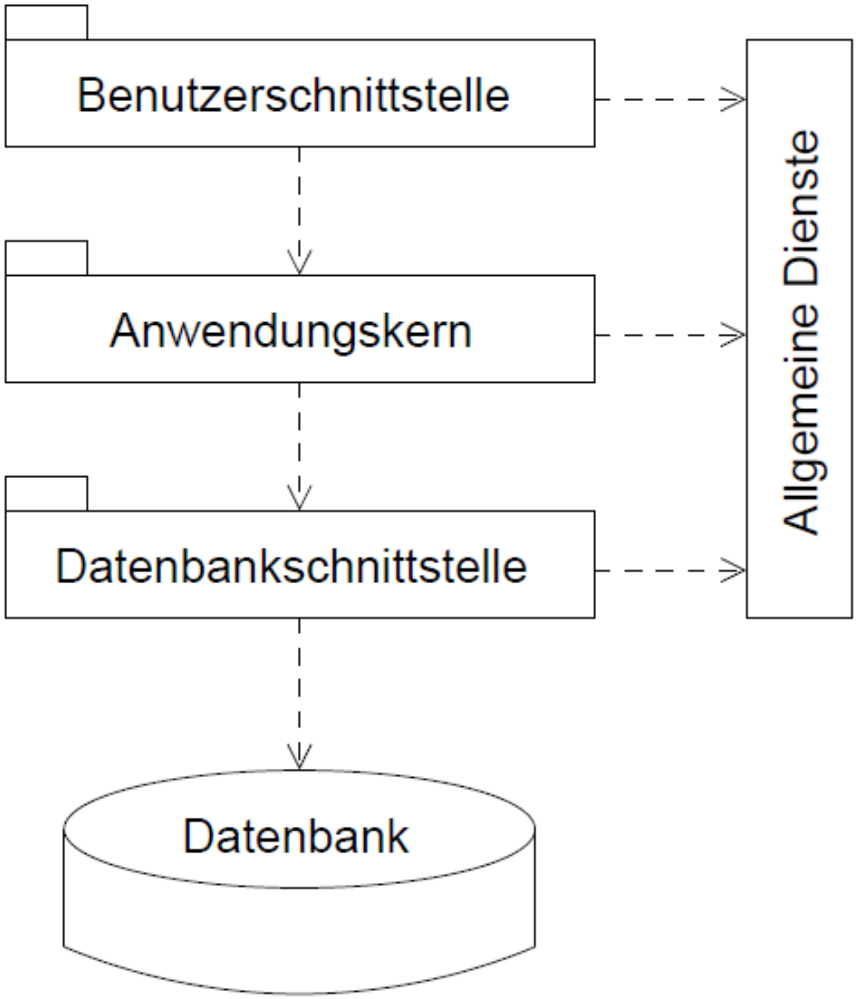
- *Hohe Kohärenz (high cohesion)*
  Zusammenfassung (logisch) zusammengehörender Teile eines Systems in einem Subsystem.
- *Geringe Kopplung (low coupling)*
  Wenige Abhängigkeiten zwischen den einzelnen Subsystemen.

Vorteil: Leichte Änderbarkeit und Austauschbarkeit von einzelnen Teilen.

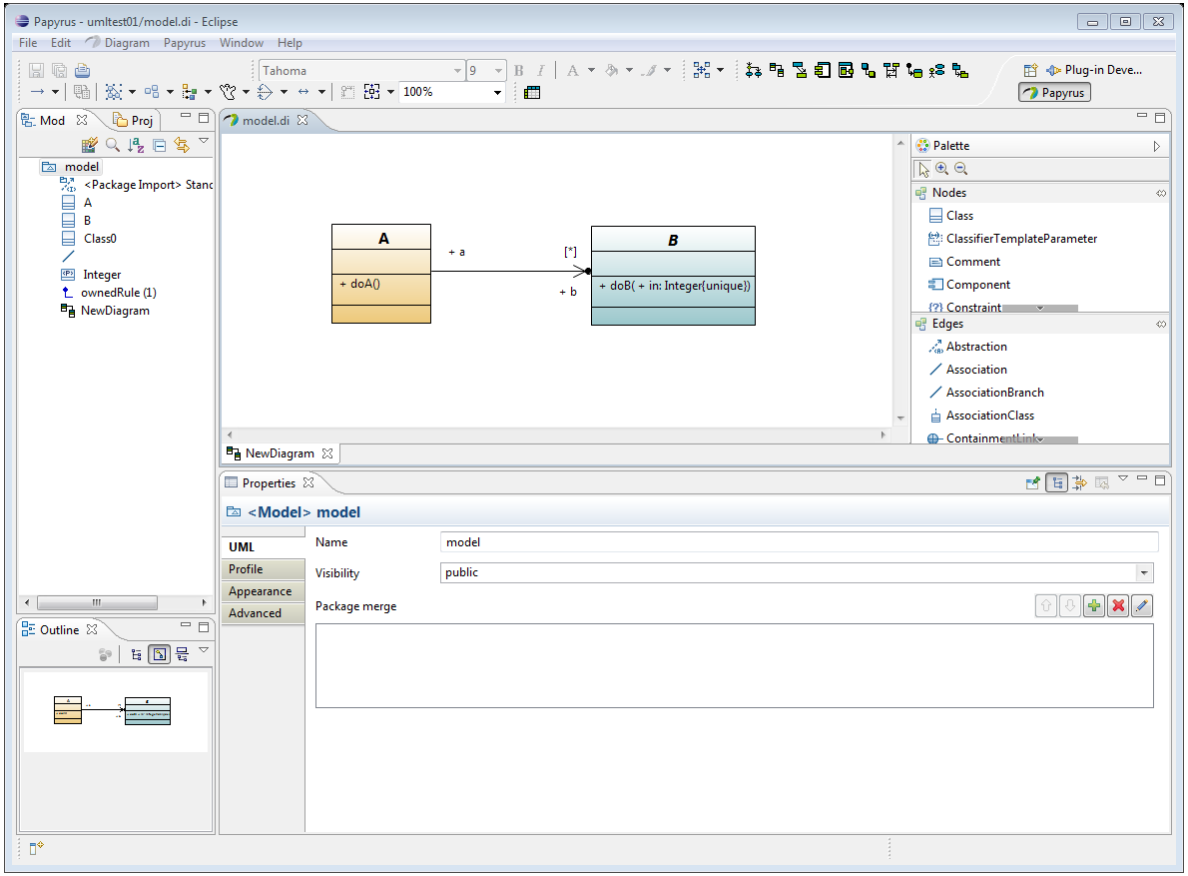
Systemarchitektur



Systemarchitektur



UML-Editor Papyrus



- www.eclipse.org/papyrus
- http://www.eclipse.org/modeling/mdt/papyrus/usersTutorials/resources/TutorialOnPapyrusUSE_d20101001.pdf