Responsibility-Driven Design

Rebecca Wirfs-Brock, Brien Wilkerson, Lauren Wiener
Designing Object-Oriented Software
Prentice-Hall

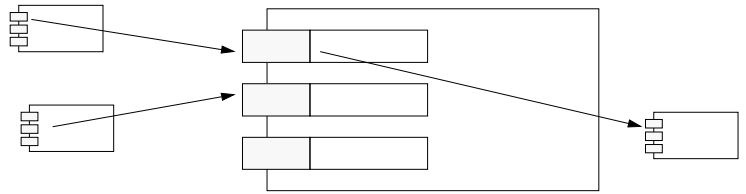
1. Begriffe und Konzepte
2. Der Entwurfsprozess
3. Hilfsmittel
4. Inhalt des Entwurfsdokuments
5. Checklisten
6. Beispiel

   ABS - der Automatische Bankschalter

Entwurfsdimensionen

Design objektorientierter Software hat 2 Dimensionen:

- Klassenhierarchie (Vererbung)
- Kooperation zwischen Objekten (Verwendung)



Grundkonzepte

1. Objektorientierte Konzepte
2. Client-Server-Prinzip
3. Subsysteme

Das Client-Server-Prinzip

Kontrakte (contracts)

Verpflichtungen (Verantwortlichkeiten, responsibilities)

Kollaborationen (collaborations)

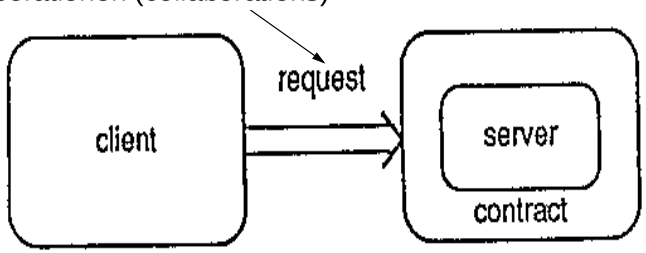


Figure 2-10: *The Client-Server Contract*

Subsysteme

Ein Subsystem ist ein Menge von Klassen, die durch Kollaborationen eine Menge von Verpflichtungen erfüllen

Klar definierte Schnittstelle nach außen

Beispiel: Printing Subsystem

Print Server
Spooler
Laser Printer
Line Printer
Plotter

Der Entwurfsprozeß

- Ausgangspunkt:

  textuelle Anforderungsspezifikation

- Ziel:

  System von Objekten

  Beschreibung des Verhaltens der Objekte

  Kommunikationsstruktur der Objekte

- Phasen:

  Erkundungsphase (Initial Exploration)

  Analysephase (Detailed Analysis)

Erkundungsphase (Initial Exploration)

1. Finde Klassen
2. Bestimme Verpflichtungen
3. Lege Kollaborationen fest

Analysephase (Detailed Analysis)

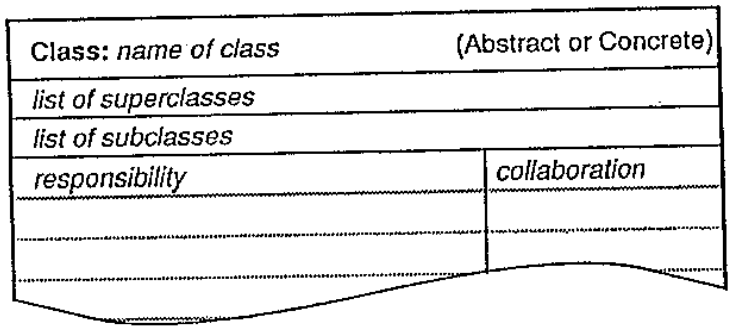
1. Bilde gute Klassenhierarchien und lege Kontrakte fest
2. Bilde Subsysteme und vereinfache Kollaborationen
3. Lege Protokolle fest

Hilfsmittel

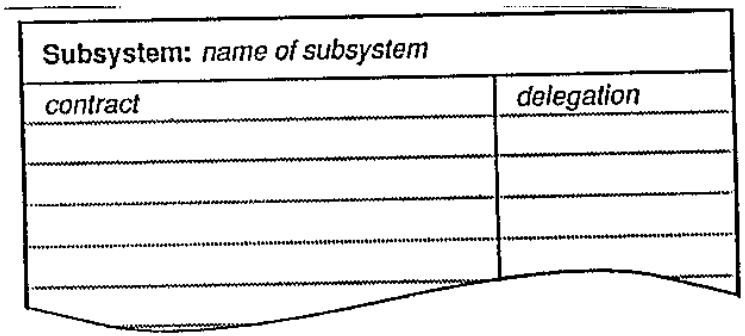
1. Klassen- und Subsystemkarten

   CRC-Karten (= Class-Responsibilities-Collaborators)
2. Hierarchiegraph
3. Venndiagramm
4. Kollaborationsgraph

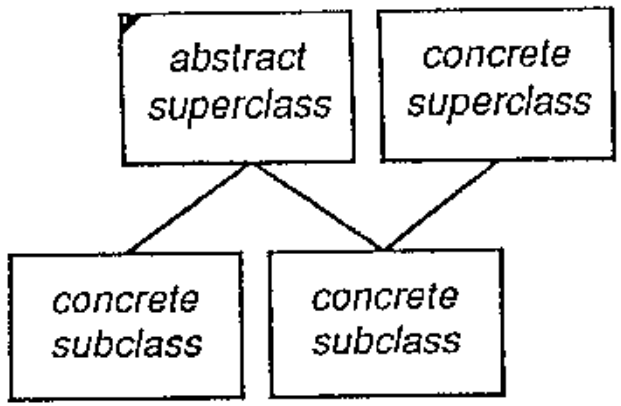
Klassenkarte



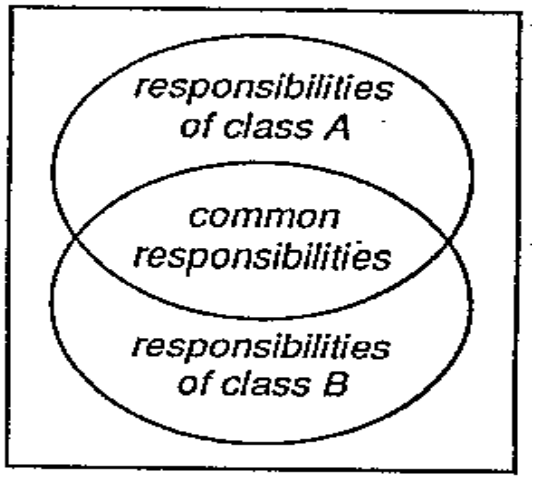
Subsystemkarte



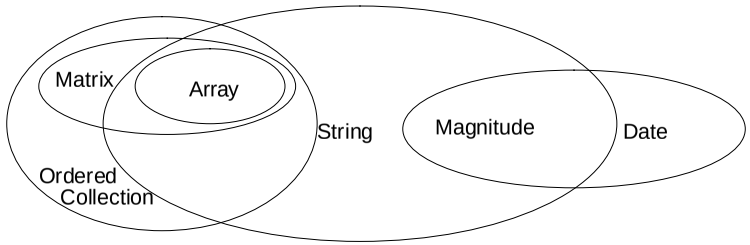
Hierarchiegraph



Venndiagramm



Venndiagramm -Beispiel



Kollaborationsgraph

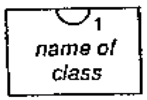


Figure A-6: *A Contract*

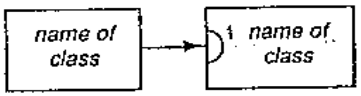


Figure A-7: *A Collaboration*

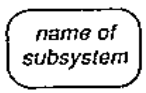


Figure A-8: *A Subsystem*

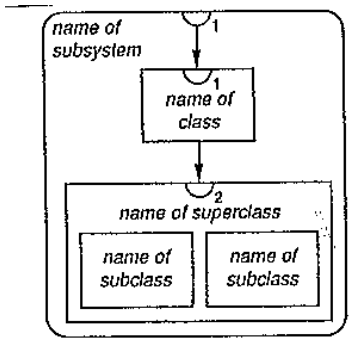


Figure A-9: *An Example Collaborations Gra*

Inhalt des Entwurfsdokuments

1. Graph für jede Klassenhierarchie
2. Kollaborationsgraph für Gesamtsystem und jedes Subsystem
3. Klassenbeschreibung für jede Klasse
4. Subsystembeschreibung für jedes Subsystem
5. Kontraktbeschreibung für jeden Kontrakt

Klassenbeschreibung

Class: *class name* (*Concrete* or *Abstract*)
Superclasses: *class names*
Subclasses: *class names*
Hierarchy Graphs: page *page number*
Collaborations Graphs: page *page number*
Description: *description of the class*
Contracts
#. *contract name*
responsibility
signature
uses *list of collaborations*
description of the signature
Private Responsibilities
responsibility
signature
uses *list of collaborations*
description of the signature
— *page #* —

Subsystembeschreibung

Subsystem: *subsystem name*
Classes: *list of classes and subsystems*
Collaborations Graphs: page *page number*
Description: *description of the subsystem*
Contracts
#. *contract name*
Server: *class or subsystem name*
— *page #* —

Figure A-11: *A Subsystem Specification*

Kontraktbeschreibung

Contract #: *contract name*
Server: *class name*
Clients: *class or subsystem names*
Description: *description of the contract*
— *page #* —

Erkundungsphase (1)

1. Lese Anforderungsspezifikation
2. Für alle weiteren Schritte: Untersuche Alternativen, halte Ergebnisse auf Karten fest
   - Ergebnisse sind meistens nicht auf Anhieb vollständig, auch nicht unbedingt korrekt
   - Schritte sind daher nicht notwendigerweise sequentiell zu durchlaufen
3. Bilde Liste von Nominalphrasen
4. Suche versteckte Nominalphrasen
5. Identifiziere Klassen

Erkundungsphase (2)

6. Suche Kandidaten für Superklassen
7. Identifiziere fehlende Klassen
8. Halte Zweck jeder Klasse schriftlich fest
9. Finde Verpflichtungen (aus Verbphrasen)
10. Ordne Verpflichtungen Klassen zu
11. Führe *Ablaufszenarien* durch, um die Vollständigkeit der Verpflichtungen zu prüfen

Erkundungsphase (3)

12. Finde Kollaborationen
13. Stelle fest, ob sich aus Beziehungen zwischen Klassen weitere Kollaborationen ergeben
14. Eliminiere unnötige Klassen
15. Zeichne Hierarchiegraphen
16. Identifiziere abstrakte und konkrete Klassen
17. Zeichne Venndiagramme
18. Restrukturiere Klassenhierarchien

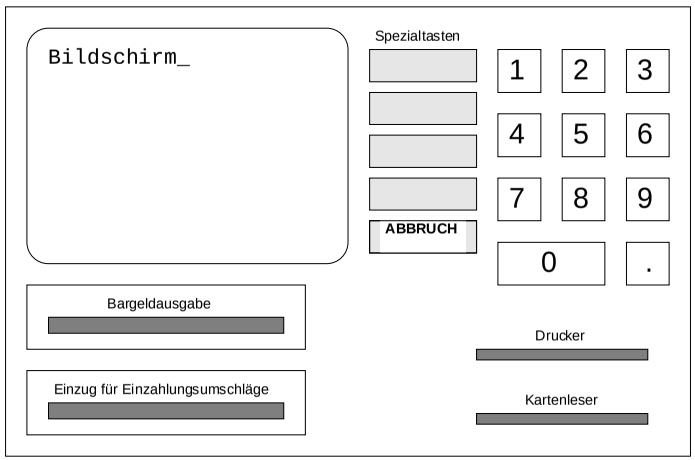
Erkundungsphase (4)

19. Lege Kontrakte von Klassen fest
    - Numerieren der Kontrakte
    - Festlegen privater Verpflichtungen
    - Explizite Angabe der Benutzung von Kontrakten auf Klassenkarten
20. Zeichne Kollaborationsgraph
21. Identifiziere mögliche Subsysteme
22. Vereinfache Kollaborationen

Erkundungsphase (5)

23. Definiere Protokoll jeder Klasse (Signaturen)
24. Erstelle Klassenbeschreibungen
25. Erstelle Subsystembeschreibungen
26. Erstelle Vertragsbeschreibungen

ABS - der Automatische Bankschalter



Anforderungsspezifikation (1)

Ein automatischer Bankschalter (ABS) ist eine Maschine, mit der Bankkunden eine Anzahl finanzieller Transaktionen durchführen können. Die Maschine besteht aus einem Bildschirm, einem Bankkartenleser, numerischen und Funktionstasten, einem Geldausgabeschlitz, einem Einzug für Umschläge mit Einzahlungen und einem Belegdrucker.

Wenn die Maschine bereit ist, wird eine Grußmeldung angezeigt. Die Tasten und der Einzug bleiben inaktiv, bis eine Karte eingeschoben wird. Wird eine Karte eingezogen, versucht der Kartenleser sie zu lesen. Ist die Karte nicht lesbar, wird der Benutzer darüber informiert und die Karte wird wieder ausgegeben.

Wenn die Karte lesbar ist, wird der Benutzer aufgefordert, eine persönliche Identifikations Nummer (PIN) einzugeben. Der Benutzer erhält die Information, wieviele Ziffern er eingegeben hat, nicht aber die einzelnen eingegebenen Ziffern. Ist die PIN korrekt eingegeben, erscheint das Hauptmenü (s.u.). Andernfalls erhält der Benutzer zwei weitere Chancen, die PIN korrekt einzugeben. Nach dem dritten erfolglosen Versuch zieht die Maschine die Karte ein. Der Benutzer kann die Karte nur noch von autorisiertem Bankpersonal zurückerhalten.

Anforderungsspezifikation (2)

Das Hauptmenü enthält eine Liste der ausführbaren Transaktionen. Diese Transaktionen sind:

- Einzahlung auf ein Konto,
- Abhebung von einem Konto,
- Überweisung von einem Konto auf ein anderes und
- Abfrage eines Kontostandes

Der Benutzer kann eine Transaktion auswählen und die relevanten Informationen spezifizieren. Nach Abschluß einer Transaktion kehrt das System zum Hauptmenu zurück.

Anforderungsspezifikation (3)

Zu jedem Zeitpunkt nach Erreichen des Hauptmenüs und vor dem Abschluß einer Transaktion (auch vor der Auswahl einer Transaktion) kann der Benutzer die “ABBRUCH”-Taste drücken. Die spezifizierte Transaktion (falls eine ausgewählt wurde) wird abgebrochen, die Bankkarte des Benutzers wird wieder ausgegeben, ein Beleg über alle durchgeführten Transaktionen wird ausgedruckt und die Maschine ist wieder bereit.

Wird eine Einzahlungstransaktion ausgewählt, wird der Benutzer aufgefordert, das Konto, auf das eingezahlt werden soll, und den Betrag anzugeben. Anschließend wird er gebeten, den Umschlag mit der Einzahlung in den Einzug zu geben.

Wird die Abhebungstransaktion gewählt, soll der Kunde Kontonummer und Betrag der Abhebung angeben. Wenn das Konto einen ausreichenden Stand hat, wird das Bargeld dem Benutzer am Ausgabeschlitz ausgegeben.

Anforderungsspezifikation (4)

Wird die Überweisung spezifiziert, wird der Benutzer gebeten, das Quell- und das Zielkonto sowie den zu überweisenden Betrag anzugeben. Befindet sich genügend Geld auf dem Ausgangskonto, wird der Transfer durchgeführt.

Wird die Kontoabfrage ausgewählt, wird der Benutzer aufgefordert, das Konto anzugeben, dessen Stand er erfragen möchte. Der Kontostand wird nicht am Bildschirm sondern am Drucker ausgegeben.

Erkundungsphase (Anfang)

1. Lese Anforderungsspezifikation
2. Für alle weiteren Schritte: Untersuche Alternativen, halte Ergebnisse auf Karten fest

- Ergebnisse sind meistens nicht auf Anhieb vollständig, auch nicht unbedingt korrekt
- Schritte sind daher nicht notwendigerweise sequentiell zu durchlaufen

Erkundungsphase (Klassen)

3. Bilde Liste von Nominalphrasen
4. Suche versteckte Nominalphrasen
5. Identifiziere Klassen

Kriterien

- Physische Objekte
- Konzeptuell sinnvolle Abstraktionen (Windows, Files...)
- Sinnvolle Auswahl bei Synonymen (Benutzer, Person)
- Bedeutung von Adjektiven? Wichtige Adjektive können auf Klassenunterscheidung hinweise
- Passivsätze (implizites Subjekt)
- Klassenkategorien (verschiedene Klassen mit ähnlicher Bedeutung)
- Schnittstellen zur Außenwelt
- Attribut*werte*, nicht Attribute selbst (Kontodaten)

Erste Liste von Nominalphrasen (1)

ABS	Normale Tastatur
Maschine	Bargeldausgabe
Bankkunde	Einzahlungseinzug
finanzielle Transaktion	Abbruchtaste
Bildschirm	Grußmeldung
Bankkartenleser	Taste
Numerische Taste	Bankkarte
Spezialtaste	Karte
Geldausgabeschlitz	Benutzer
Einzahlungseinzug	Persönliche Identifikationsnummer
Belegdrucker	PIN
Element	Feedback
Spezialtastatur	Ziffer

wird gestrichen

Erste Liste von Nominalphrasen (2)

Numerische Tastatur	Zeit
Hauptmenü	Quittung
Chance	Einzahlungstransaktion
erfolgloser Versuch	Summe
dritter Versuch	Einzahlung
autorisiertes Bankpersonal	Einzahlungsumschlag
Liste von Transaktionen	Abhebungstransaktion
Konto	Abhebung
Drucker	Überweisung von Einlagen
Einlagen	Überweisung
Kontostand	genügend Einlagen
Information	Kontostandabfrage
System	Transaktion

wird gestrichen

Systemabgrenzung

ABS	Grußmeldung	Konto
Bankkunde	Taste	Kontostand
finanzielle Transaktion	Bankkarte	Information
Bildschirm	Benutzer	Zeit
Bankkartenleser	PIN	Quittung
Numerische Taste	Feedback	Einzahlungstransaktion
Spezialtaste	Numerische Tastatur	Summe
Einzahlungseinzug	Hauptmenü	Abhebungstransaktion
Belegdrucker	Chance	Überweisung von Einlagen
Element	erfolgloser Versuch	Kontostandsabfrage
Spezialtastatur	dritter Versuch	Transaktion
Bargeldausgabe	autorisiertes Bankpersonal	
Abbruchtaste	Liste von Transaktionen	

[ ] wird gestrichen

Eliminierung von Vorgängen

ABS	Taste	Summe
finanzielle Transaktion	PIN	Abhebungstransaktion
Bildschirm	Feedback	Überweisung von Einlagen
Bankkartenleser	Numerische Tastatur	Kontostandsabfrage
Numerische Taste	Hauptmenü	Transaktion
Spezialtaste	Chance	
Einzahlungseinzug	erfolgloser Versuch	
Belegdrucker	dritter Versuch	
Element	Liste von Transaktionen	
Spezialtastatur	Konto	
Bargeldausgabe	Kontostand	
Abbruchtaste	Information	
Grußmeldung	Einzahlungstransaktion	

[ ] wird gestrichen

Überlegungen zu Kandidatenklassen

ABS	PIN (Geheimzahl)	
Bildschirm	Kontostandsabfrage	
Bankkartenleser	Grußmeldung →	*Meldung*
Numerische Taste	Hauptmenü →	*Menü*
Spezialtaste	Liste von Transaktionen	
Einzahlungseinzug	Element →	***Device***
Belegdrucker	Kontostand	
Konto	Information	***Input Device***
Spezialtastatur	Einzahlungstransaktion	***Output Device***
Bargeldausgabe	Summe	
Abbruchtaste	Abhebungstransaktion	
Numerische Tastatur	Überweisung von Einlagen	
Taste	**Transaktion**	

[ ] wird gestrichen

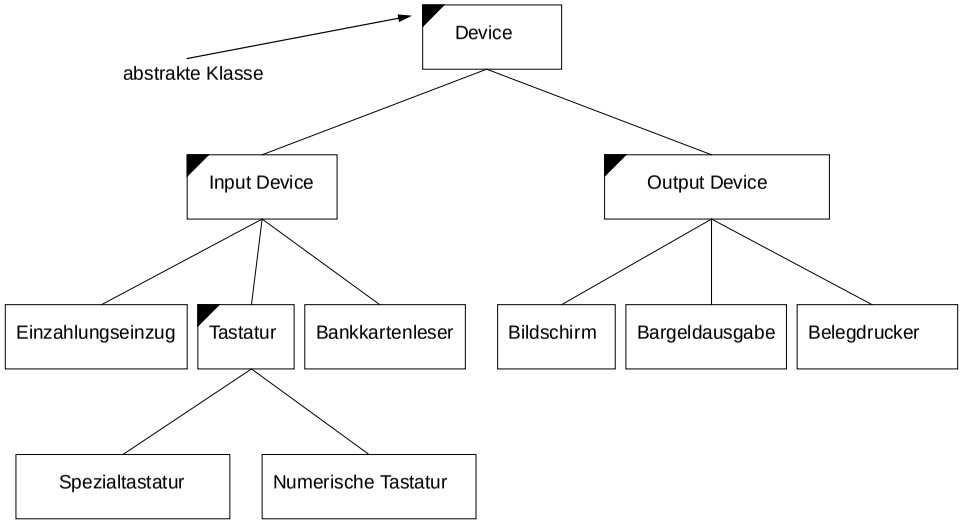
Erkundungsphase (Klassen)

6. Suche Kandidaten für Superklassen

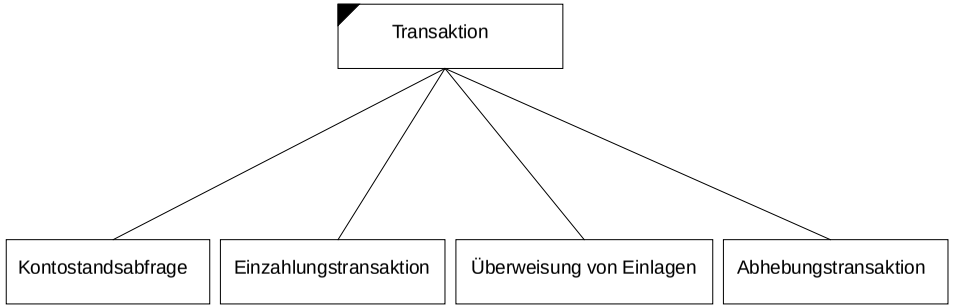
Beispiele:

- Geräte haben offensichtlich bestimmte Ähnlichkeiten
- außerdem gibt es Geräte, die Eingaben entgegennehmen, und Geräte, die Ausgaben durchführen
- Abhebungs-, Einzahlungstransaktion und Überweisung sind offensichtlich unterschiedliche Arten von Transaktionen

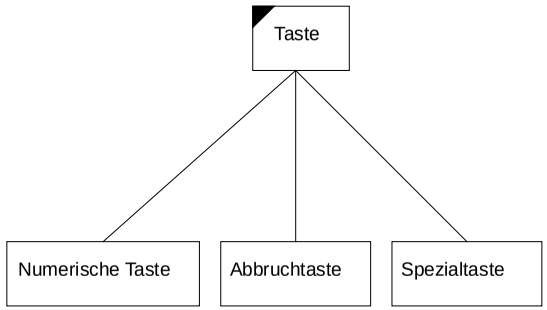
Device-Hierarchie



Transaktionshierarchie



Tastenhierarchie



Erkundungsphase (Klassen)

7. Identifiziere fehlende Klassen
8. Halte Zweck jeder Klasse schriftlich fest

Anm.: Schritt 7 so früh ziemlich problematisch - Methode gibt wenig Hilfestellung

Liste der Klassenkandidaten (1)

ABS: führt finanzielle Dienstleistungen für Bankkunden aus

Kontostandsabfrage: informiert den Kunden über seinen Kontostand

Bankkartenleser: liest und überprüft die Bankkarte eines Kunden

Abbruchtaste: stellt das Beenden eines Dienstes durch den Kunden fest

Bargeldausgabe: gibt Bargeld an den Kunden aus

Einzahlungseinzug: nimmt Umschläge mit Einzahlungen des Kunden entgegen

Einzahlungstransaktion: führt eine Einzahlung des Kunden durch und korrigiert seinen Kontostand entsprechend

Device: repräsentiert Hardware, mit Hilfe derer ABS und Bankkunde kommunizieren

Bildschirm: Der Bildschirm stellt Text und Graphik dar

Überweisung von Einlagen

Liste der Klassenkandidaten (2)

Input Device: Device, durch das der Kunde mit dem ABS kommuniziert

Taste: Knopf, den der Bankkunde drücken kann

Tastatur: eine Gruppe von Tasten, die alle den selben Zweck erfüllen

Menü: zeigt Information und ermöglicht dem Kunden eine Auswahl

Numerische Taste: eine Taste in einer Numerischen Tastatur

Numerische Tastatur: registriert die Eingabe numerischer Daten

Output Device: ein Device, durch das der ABS mit dem Kunden kommuniziert

PIN: die Autorisierung eines Bankkunden

Belegdrucker: druckt alle Transaktionen eines Kunden aus

Spezialtaste: eine Taste in einer Spezialtastatur

Spezialtastatur: registriert eine Menüauswahl

Liste der Klassenkandidaten (3)

Transaktion: führt einen finanziellen Dienst für einen Kunden aus und korrigiert sein Kon*to (bzw. seine Konten)*

Konto: repräsentiert das Bankkonto eines Kunden in der Datenbank der Bank

Abhebungstransaktion: führt eine Abhebung für den Kunden aus und korrigiert seinen Kontostand

Erkundungsphase (Verpflichtungen)

9. Finde Verpflichtungen (aus Verbphrasen)
10. Ordne Verpflichtungen Klassen zu

Kriterien

- “Intelligenz” gleichmäßig verteilen
  - “intelligenz”: “Wissen” und Informationen einer Klasse, welche anderen Objekte beeinflußt sie?
  - Beispiel: GeometrischesObjekt hat Wissen über Drehen, Verschieben, Zeichnen
  - Indikator für Komplexität einer Klasse: Anzahl der Verpflichtungen
  - Vorteile bei gleichmäßiger Verteilung: Wenn Einzelobjekte über weniger Dinge Bescheid wissen, bedeutet das Ändern eines Dings weniger Arbeit beim Ändern von Objekten

Kriterien (2)

- Verpflichtungen allgemein formulieren
  - erleichtert das Finden von Gemeinsamkeiten (und letztendlich das Finden von Oberklassen)
  - Bsp.: Rechtecke sollen nicht “Rechtecke zeichnen” (oder Kreise: “Kreise zeichnen”), beide haben die Verpflichtung, sich selbst zu zeichnen
- Verhalten zs. mit der entsprechenden Information unterbringen
  - Wenn ein Objekt bestimmte Informationen verwaltet, sollten Verpflichtungen, die auf diesen Operationen arbeiten, Verpflichtungen dieses Objekts sein (Datenkapselung)

Kriterien (2)

- Informationen nicht mehrfach abspeichern
  - Beispiel: Zeichnung, geometrische Objekte
    Entweder die Zeichnung kennt die Positionen, oder die geometrischen Objekte selbst
  - Wege dorthin:
    - Zusammenfassen der betroffenen Objekte
    - Spezielles Objekt definieren
    - ein Objekt zur Speicherung auswählen

Kriterien (3)

- manchmal Verpflichtungen teilen
  - Beispiel:
    - Zeichenprogramm weiß, wann neu gezeichnet wird
    - Zeichnung kennt die darzustellenden Elemente
    - Elemente zeichnen sich selbst
- Suche nach “Beziehungen”:
  - is-kind-of
  - part-of
  - Analogien (können zu analogen Verpflichtungen führen)
- neue Klassen? (z.B. Manipulieren von geometrischen Objekten in Gruppen)

Zuordnung von Verpflichtungen(1)

Finanzielle Transaktionen durchführen → **ABS** (nur "Transaktionen generieren + initialisieren")

Grußmeldung anzeigen → **ABS** (globale Kontrolle, nicht Trans.-/Kartenspez.)

Eine Bankkarte lesen → **Bankkartenleser** (*validierte* Karte lesen)

Benutzer über unlesbare Karte informieren → **Bankkartenleser**

Karte ausgeben → **Bankkartenleser**

Den Benutzer zur Eingabe einer PIN auffordern → **Bankkartenleser + Formular** (s.o.: *validierte* Karte)
Formular: 3. Eingabeart (Prompt+numerische Eing.)

Anzahl der eingegebenen Ziffern anzeigen → **Geschütztes Formular**

Das Hauptmenü anzeigen → **ABS + Menü**

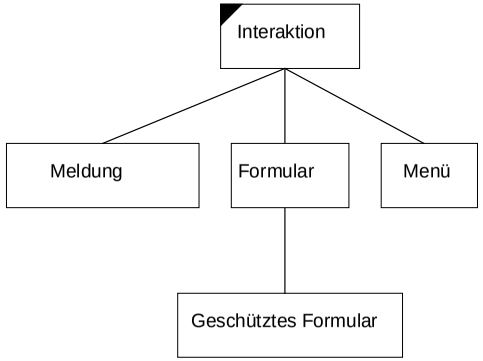
Eine Bankkarte einbehalten → **Bankkartenleser**

Die aktuelle Transaktion abbrechen → **Transaktion** (Abbruchtaste? ABS?)

Einen Beleg der Transaktionen drucken → **Belegdrucker**

Das Konto für eine Einzahlung erfragen → **Einzahlungstransaktion + Formular**

Interaktionshierarchie



Zuordnung von Verpflichtungen (2)

Den Betrag einer Einzahlung erfragen → **Einzahlungstransaktion + Formular**

Einen Einzahlungsumschlag annehmen → **Einzahlungseinzug**

Das Konto einer Abhebung erfragen → **Abhebung + Menü**

Den Betrag einer Abhebung erfragen → **Abhebungstransaktion + Formular**

Bargeld an den Benutzer ausgeben → **Bargeldausgabe**

Den Betrag einer Überweisung erfragen → **Überweisung von Einlagen + Formular**

Geld überweisen → **Überweisung von Einlagen**

Das abzufragende Konto erfragen → **Kontostandabfrage + Menü**

Verpflichtungen auf der Klassenkarte

Bankkartenleser	
Eine Bankkarte lesen	
Benutzer über unlesbare Karte informieren	
Bankkarte ausgeben	
Den Benutzer zur Eingabe einer PIN auffordern	
Eine Bankkarte einbehalten	

Umformulieren von Verpflichtungen

Formular

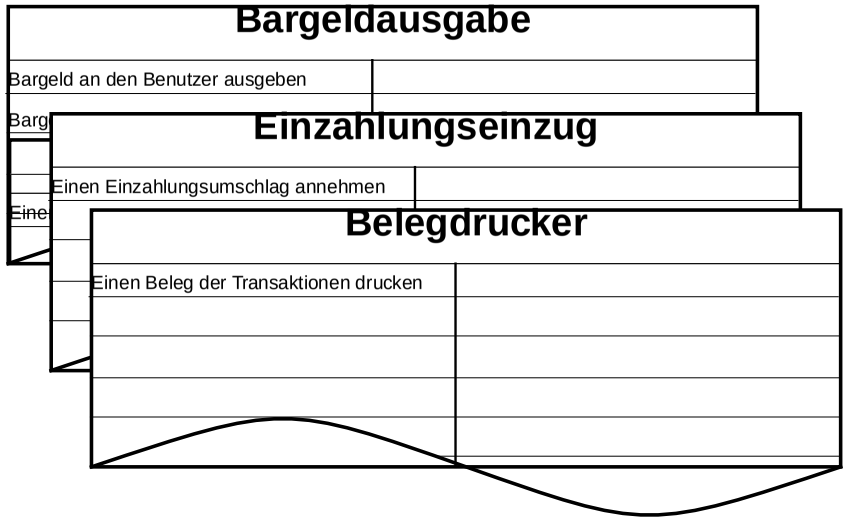
Dem Benutzer die Beschreibung gewünschter Informationen präsentieren
Wissen, ob der Benutzer geantwortet hat
Während der Eingabe Feedback geben
Die Antwort des Benutzers kennen

Menü

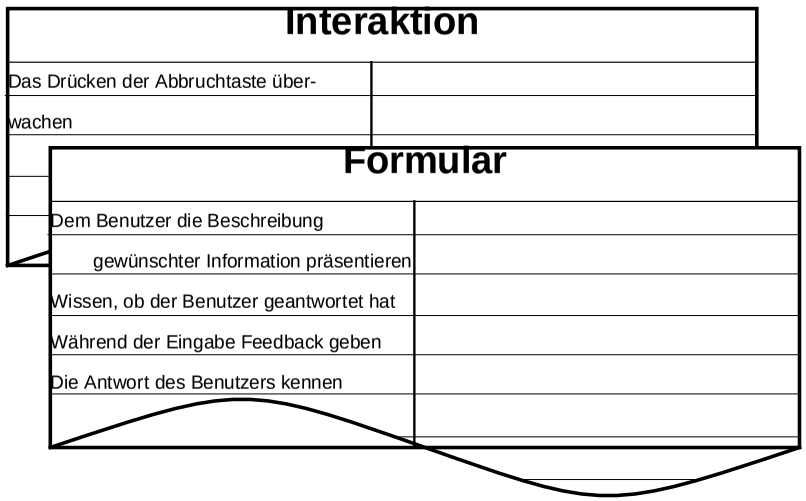
Dem Benutzer eine Auswahlliste anbieten
Wissen, ob der Benutzer geantwortet hat
Die Antwort des Benutzers kennen

- Suche nach Verpflichtungen für Klassen, die noch keine Verpflichtungen haben (Konto, Taste, Tastatur, ...)

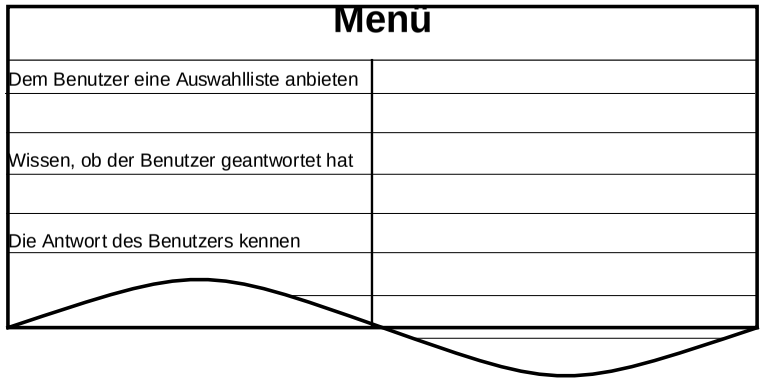
Verpflichtungen auf der Klassenkarte



Verpflichtungen auf der Klassenkarte



Verpflichtungen auf der Klassenkarte



Erkundungsphase (Verpflichtungen)

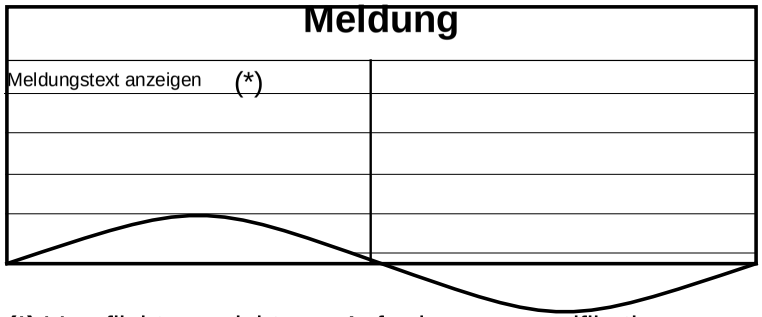
11. Führe *Ablaufszenarien* durch, um die Vollständigkeit der Verpflichtungen zu prüfen

Beispiel eines Ablaufszenarios

- Der ABS ist bereit
  - **ABS** gibt Grußmeldung aus (**Meldung**)
  - **ABS** fragt **Kartenleser**, ob gültige Karte eingegeben wurde
  - **Kartenleser** prüft, ob Karte vorhanden
  - **Kartenleser** prüft, ob Karte lesbar
  - **Kartenleser** fordert zur Eingabe einer PIN auf (**Geschütztes Formular**)
  - **Kartenleser** überprüft **PIN**

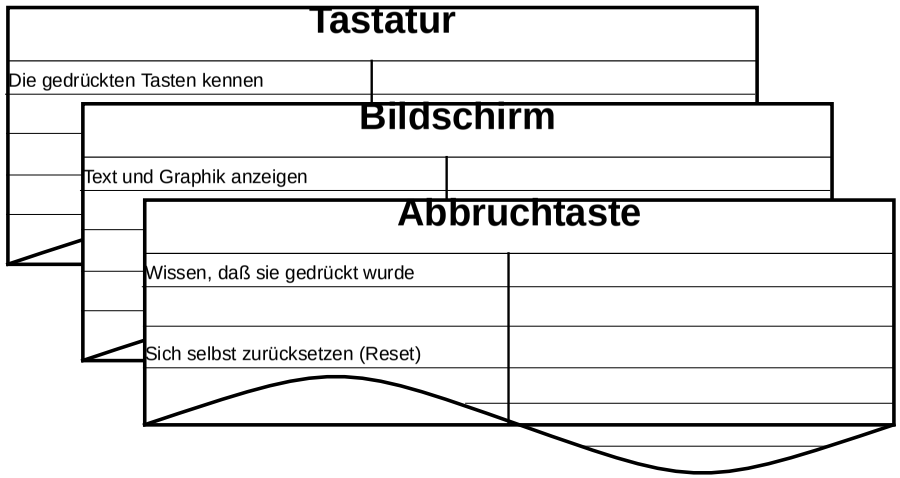
neue Verpflichtung: Meldungstext anzeigen (**Meldung)**

Verpflichtungen auf der Klassenkarte



(*) Verpflichtung nicht aus Anforderungsspezifikation

Verpflichtungen auf der Klassenkarte



Klassen ohne Verpflichtungen

Geschütztes Formular

bleibt erhalten, weil ererbtes Verhalten redefiniert wird

PIN

wird gestrichen, weil nicht mehr benötigt

Erkundungsphase (Kollaborationen)

12. Finde Kollaborationen
13. Stelle fest, ob sich aus Beziehungen zwischen Klassen weitere Kollaborationen ergeben
14. Eliminiere unnötige Klassen

Kollaborationen

Kollaborationen stellen Zusammenarbeit von Objekten mit dem Ziel der Erfüllung von Verpflichtungen dar

- Eine Kollaboration dient dem Erfüllen einer bestimmten Verpflichtung
- Mehrere Kollaborationen können zum Erfüllen dieser Verpflichtungen notwendig sein
- Kollaborationen sind immer gerichtet: vom Client zum Server
- Jede Klasse sollte in irgendeiner Richtung an einer Kollaboration beteiligt sein (Ausn.: Systemgrenzen)
- möglich: Kollaborationen mit sich selbst

Finden von Kollaborationen

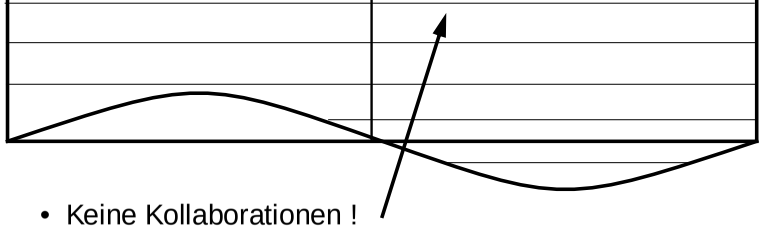
- Frage für jede Verpflichtung einer Klasse: Kann die Klasse diese Verpflichtung allein erfüllen?
- Welche Information benötigt diese Klasse? Welche Klassen verfügen über diese Information?
- In welcher Beziehung steht diese Klasse zu anderen Klassen?
  - Teilebeziehungen
  - Informationsfluß
  - Abhängigkeiten

Kollaborationen

Bankkartenleser	
Superklasse(n): Input Device	
Eine Bankkarte lesen	
Benutzer über unlesbare Karte	Meldung
informieren	
Bankkarte ausgeben	
Den Benutzer zur Eingabe einer	Geschütztes Formular
PIN auffordern	
Eine Bankkarte einbehalten	

Kollaborationen

Bargeldausgabe	
Superklasse(n): Output Device	
Bargeld an den Benutzer ausgeben	



- Keine Kollaborationen !

Kollaborationen

Einzahlungseinzug	
Superklasse(n): Input Device	
Einen Einzahlungsumschlag annehmen	

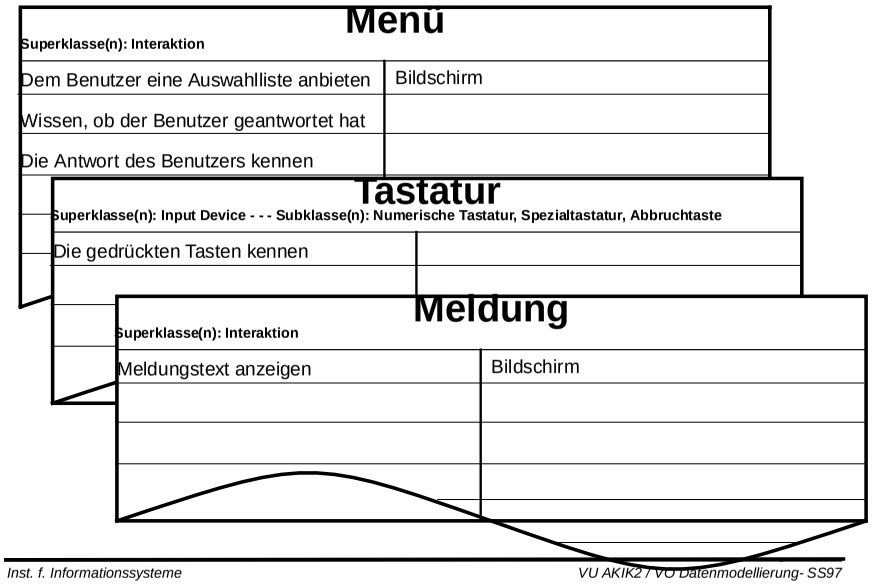
Belegdrucker	
Superklasse(n): Output Device	
Einen Beleg der Transaktionen drucken	
Einen Beleg ausgeben	

Interaktion	
Subklasse(n): Formular, Menü, Meldung	
Das Drücken der Abbruchtaste über-	Tastatur
wachen	

Kollaborationen

Formular	
Superklasse(n): Interaktion - - - Subklasse(n): Geschütztes Formular	
Dem Benutzer die Beschreibung	Bildschirm
gewünschter Information präsentieren	
Wissen, ob der Benutzer geantwortet hat	
Während der Eingabe Feedback geben	Bildschirm
Die Antwort des Benutzers kennen	Numerische Tastatur

Kollaborationen



Analysephase (Hierarchien)

15. Zeichne Hierarchiegraphen
16. Identifiziere abstrakte und konkrete Klassen
17. Zeichne Venndiagramme
18. Restrukturiere Klassenhierarchien

Kriterien für die Hierarchie

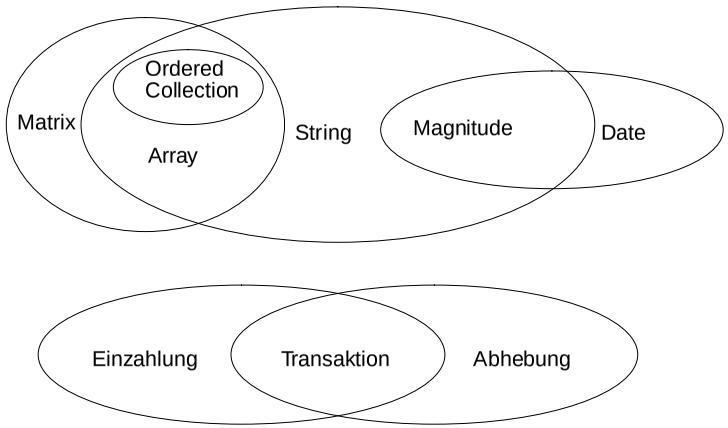
- Die Hierarchie soll eine “Kind-of”-Hierarchie darstellen (IS-A, Spezialisierung)
- Gemeinsame Verpflichtungen so hoch oben wie möglich
  - Wenn mehrere Klassen dieselbe Verpflichtung haben, dann eine abstrakte Superklasse (erhöhte Wiederverwendbarkeit, leichtere Modifizierbarkeit)
  - Abstrakte Superklassen nur dann, wenn mehrfache Unterklassen tatsächlich vorhanden oder vorstellbar: Konkrete Beispiele sind notwendig, um sinnvolle Abstraktionen entwerfen zu können

Kriterien für die Hierarchie (2)

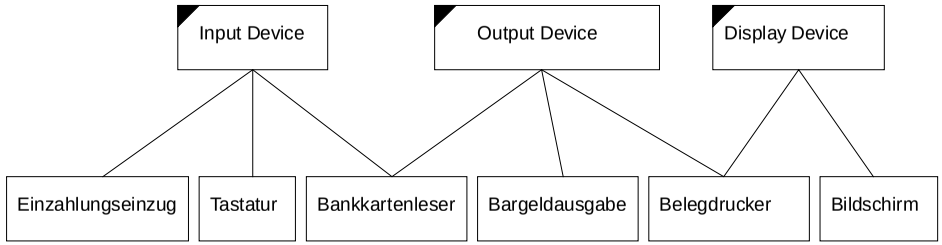
- Abstrakte Klassen sollten nicht von nicht-abstrakten erben
  - Grund: Abstrakte Klassen heißen so, genau weil sie implementierungsunabhängig sind
- Klassen, die keine neue Funktionalität definieren, werden entfernt
  - im allgemeinen: Klassen, die keine Verpflichtungen definieren
  - **Aber**: Klassen, die ererbte Verpflichtungen speziell implementieren, bleiben erhalten

Venndiagramme

- Teilmengenbeziehungen zwischen Verpflichtungen



Überarbeitete Device-Hierarchie



Analysephase (Kontrakte)

19. Lege Kontrakte von Klassen fest
    - Numerieren der Kontrakte
    - Festlegen privater Verpflichtungen
    - Explizite Angabe der Benutzung von Kontrakten auf Klassenkarten

Kontrakte

- Kontrakte (*Contracts*, Verträge) dienen zur inhaltlichen Gruppierung von Verpflichtungen
- Eine Klasse kann mehrere Kontrakte erfüllen
- Manche Verpflichtungen dienen internen Zwecken und sollten nach außen nicht sichtbar sein: private Verpflichtungen

Kontrakte (2)

- Richtlinien:
  - Zusammenfassen von Verpflichtungen, die von denselben Clients in Anspruch genommen werden
  - Die Kontrakte, die eine Klasse unterstützt, sollten einen inhaltlichen Zusammenhang aufweisen. Sonst Auslagerung in andere Klassen: Unter- oder Oberklassen, oder eine andere Server-Klasse (⇒ *Zusammensetzung*)
  - Möglichst wenig Kontrakte zwecks besserer Verständlichkeit. Ausnützung der Vererbung - ein Kontrakt sollte bei einer Klasse definiert sein und ihre Subklassen sollten ihn unterstützen.

Kontrakte - Beispiel

Konto

1. Kontostand feststellen und ändern
   Kontostand kennen
   Einzahlungen akzeptieren
   Auszahlungen akzeptieren
2. Änderungen in die Datenbank eintragen

DisplayDevice

3. Information darstellen
   Text/Graphik darstellen

Formular

4. Zahlenwert vom Benutzer erfragen
   Benutzer nach Informationen fragen
   Feststellen, ob Benutzer geantwortet hat
   Benutzerantwort kennen
   Feedback geben

Kontrakte - Beispiel (2)

Input Device

5. Benutzereingabe entgegennehmen
   physisches Objekt entgegennehmen

Menu

6. Benutzereingabe aus einer Menge von Optionen
   Auswahlmöglichkeiten präsentieren
   Wissen, ob der Benutzer geantwortet hat
   Wissen, was der Benutzer geantwortet hat

Output Deviee

7. Ausgabe zum Benutzer
   Ausgabe eines physischen Objekts

Kontrakte - Beispiel (3)

Transaktion

8. Finanzielle Transaktion ausführen

Meldung

9. Meldung darstellen und auf ein Ereignis warten
   Meldungstext ausgeben

Überarbeitete Klassenkarte

Bankkartenleser	
Superklasse(n): Input Device, Output Device	
private Verpflichtung:	
- Eine Bankkarte lesen	
- Benutzer über unlesbare Karte	Meldung (9)
informieren	
- Bankkarte ausgeben	
- Den Benutzer zur Eingabe einer	Geschütztes Formular (4)
PIN auffordern	
- Eine Bankkarte einbehalten	

Überarbeitete Klassenkarte

Bargeldausgabe	
Superklasse(n): Output Device	
private Verpflichtung	
- Bargeld an den Benutzer ausgeben	

Überarbeitete Klassenkarte

Einzahlungseinzug	
Superklasse(n): Input Device	
private Verpflichtung	
- Einen Einzahlungsumschlag annehmen	
- Kontonummer und Summe auf Umschlag	
drucken	

Belegdrucker	
Superklasse(n): Display Device, Output Device	
private Verpflichtung	
- Einen Beleg der Transaktionen drucken	
- Einen Beleg ausgeben	

Überarbeitete Klassenkarte

Formular	
Superklasse(n): Interaktion - - - Subklasse(n): Geschütztes Formular	
(4) numerischen Wert einlesen	
- Dem Benutzer die Beschreibung	Bildschirm (3)
gewünschter Information präsentieren	
- Wissen, ob der Benutzer geantwortet hat	
- Während der Eingabe Feedback geben	Bildschirm (3)
- Die Antwort des Benutzers kennen	Tastatur (5)

Überarbeitete Klassenkarte

Menü

Superklasse(n): Interaktion	
(6) Selektion aus einer Auswahl einlesen	
- Dem Benutzer eine Auswahl anbieten	Bildschirm (3)
- Wissen, ob der Benutzer geantwortet hat	
- Die Antwort des Benutzers kennen	Tastatur (5)

Meldung

Superklasse(n): Interaktion	
(9) Meldung ausgeben und auf ein	
Ereignis warten	
- Meldungstext anzeigen	Bildschirm (3)
	Spezialtastatur
	- - - Subklasse(n): Geschütztes Formular

Überarbeitete Klassenkarte

Display Device

Subklasse(n): Bildschirm, Belegdrucker	
(3) Information anzeigen	
- Text und Graphik anzeigen	

Interaktion

Subklasse(n): Formular, Menü, Meldung	
private Verpflichtung	
- Das Drücken der Abbruchtaste über-	Tastatur (5)
wachen	

Tastatur

Superklasse(n): Input Device	
private Verpflichtung	
- Die gedrückten Tasten kennen	

Überarbeitete Klassenkarte

Bildschirm

Superklasse(n): Display Device	
(*)	

(*) Verpflichtung in der Superklasse *Display Device* spezifiziert, wird hier redefiniert

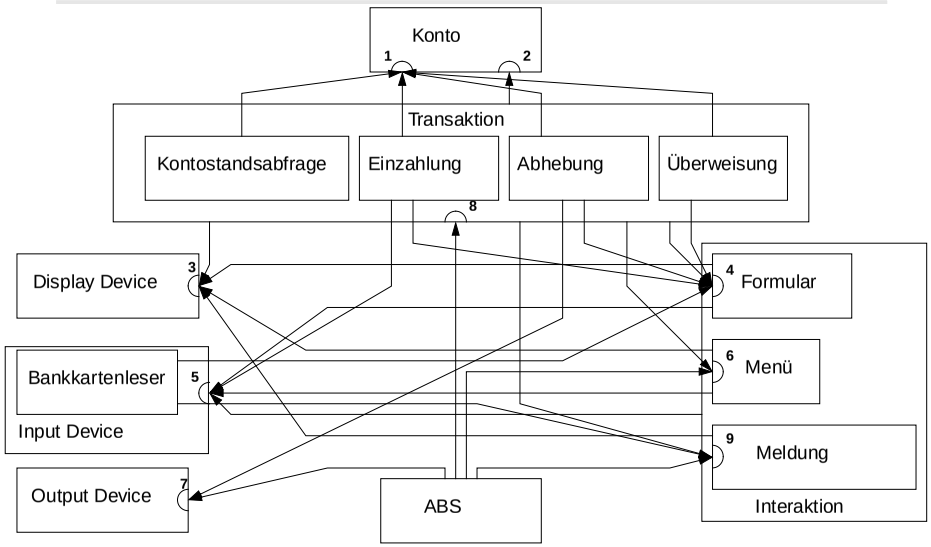
Analysephase (Subsysteme)

20. Zeichne Kollaborationsgraph
21. Identifiziere mögliche Subsysteme
22. Vereinfache Kollaborationen

Kollaborationsgraph

- Zweck von Walkthroughs: Überprüfung des Kontrollflusses
- Kollaborationsgraph: alle Kontrollflüsse auf einmal
- Zweck: Engpässe, wichtige Verbindungen
- Inhalt: Klassen, Kontrakte, Kollaborationen, hierarchische Vererbungsbeziehungen
- Vererbung: Annahme, daß alle Verpflichtungen einer Klasse auch von allen ihren Subklassen übernommen werden ⇒ Spezialisierungssicht der Vererbung

ABS-Kollaborationsgraph

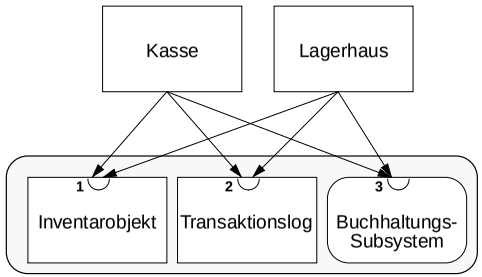


Subsysteme

- Kollaborationsgraphen können sehr komplex werden ⇒ zwei Auswege:
  - Abstraktion der Vererbung (Weglassen von Subklassen, die keine neuen Kontrakte definieren)
  - *Subsysteme*
- Ein Subsystem ist eine Kombination von Klassen und Subsystemen, die zusammenarbeiten, um eine umfassende Verpflichtung zu erfüllen ⇒ Abstraktion
- Subsysteme existieren nicht zur Laufzeit
- Verpflichtungen von Subsystemen werden an Klassen innerhalb des Subsystems delegiert

Subsystembildung (Vorgangsweise)

- Suche nach oft benutzten Kollaborationen (evt. graphische Hervorhebung im Kollaborationsgraph)
- Suche nach vielen Kollaborationen zwischen relativ wenigen Klassen. Beispiel: Supermarktkette:

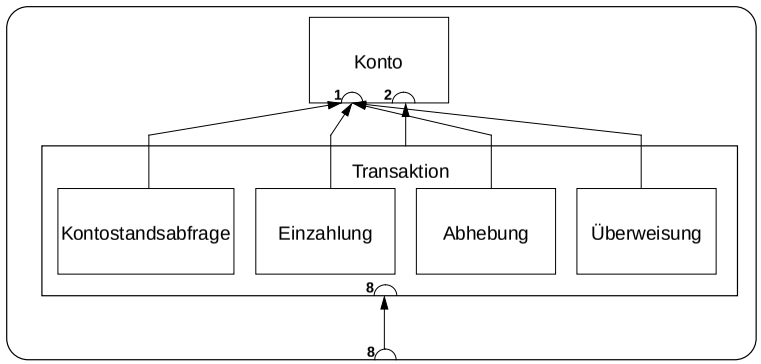


Zerlegung

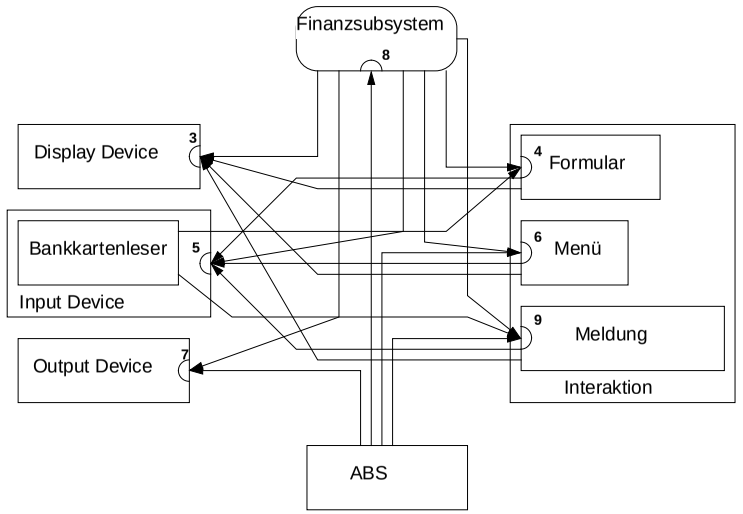
Wenn Kandidaten gefunden: Vereinfachung der Kollaborationsbeziehungen

- Wenige Kollaborationen pro Klasse
- Wenige Klassen/Subsys., an die ein Subsys. delegiert
- Wenige Kontrakte für ein Subsystem
- Nach Veränderungen: Durchführen v. Ablaufszenarios

Das Finanzsubsystem



Subsystembildung (Finanzen)



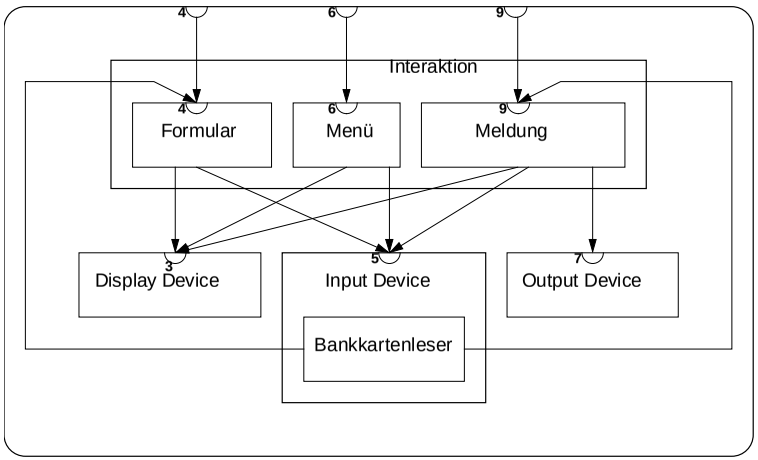
Noch ein Subsystem?

- Interface-Subsystem würde 6 Kontrakte benötigen
- warum so komplex? Wegen Transaktionen (benötigen Menüs, Geräte, Messages, Forms etc.)
- Schirm und Keypad werden nur von User Interaction-Klassen angesprochen, Kartenleser und andere Geräte aber von Transaktionen direkt
- In jedem dieser Fälle wird etwas getan, eine Message ausgegeben, und gewartet, bis der Benutzer reagiert (Karte einlegt, Geld entnimmt, Geld einlegt)

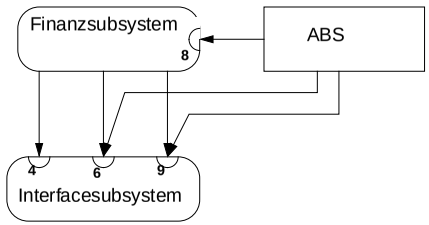
Noch ein Subsystem? (2)

- User Message: gibt bisher nur Messages aus
- zusätzliche Verpflichtung: Message ausgeben, auf richtige Reaktion warten (Reaktion hängt von Parameter ab)
- Transaktionen interagieren jetzt nicht mehr direkt mit Geräten, sondern nur noch mit User Messages
- Kartenleser: gibt Grußbotschaft über User Message aus, die auf Karteneingabe wartet

Das Interface-Subsystem



Subsystembildung (Ergebnis)



Beschreibung von Subsystemen durch Subsystemkarte

- Wie Klassenkarte, ohne Vererbungsangaben
- Kollaborationen mit Kontrakten, die durch Subsysteme erfüllt werden, auf den Klassen/Subs.-Karten notieren

Auswirkung auf Klassenkarte

Meldung Superklasse(n): Interaktion	
(9) Meldung ausgeben und auf ein	
Ereignis warten	
- Meldungstext anzeigen	Bildschirm (3)
- auf geeignete Benutzerreaktion warten	Display Device (3), Input Device (5),
	Output Device (7)

Analysephase (Protokolle)

23. Definiere Protokoll jeder Klasse (Signaturen)
24. Erstelle Klassenbeschreibungen
25. Erstelle Subsystembeschreibungen
26. Erstelle Vertragsbeschreibungen

Festlegen der Signaturen (1)

Klasse: *Meldung*

(9) Meldung ausgeben und auf Ereignis warten

gültigeKarteEingeben () returns **BenutzerAntwort**

EinzahlungEinschieben (Text) returns **Benutzer-Antwort**

entnehmeKarte () returns **BenutzerAntwort**

entnehmeQuittung () returns **BenutzerAntwort**

entnehmeBargeld (FixedPoint) returns **Benutzer-Antwort**

Festlegen der Signaturen (2)

Klasse: *BenutzerAntwort*

(10) Die Antwort des Benutzers kennen

gültig () returns **Boolean**

Wert () returns **Object**

(11) Die Antwort des Benutzers speichern

setzeGültig ()

setzeUngültig ()

setzeWert (Object)

Genaue Klassenbeschreibung (1)

konkrete Klasse*: Meldung*

Superklasse(n): *Interaktion-* - **Subklasse(n):** keine

Hierarchiegraph: Seite nn

Kollaborationsgraph: Seite nn *(Interfacesubsystem)*

Beschreibung: Diese Klasse repräsentiert eine Klasse von Benutzerinteraktion, in der der Benutzer zu einer Aktion aufgefordert wird. Die Kontrolle wird erst nach Ausführen der Aktion an das aufrufende Objekt zurückgegeben.

Verträge:

(9) Meldung ausgeben und auf ein Ereignis warten

Meldung anzeigen

gültigeKarteEingeben () returns **BenutzerAntwort**

benutzt **DisplayDevice(3) , InputDevice(5)**

Diese Methode zeigt eine Meldung, die den Benutzer zum Einschieben einer Karte auffordert, und wartet bis eine gültige Karte eingeschoben wurde

Genaue Klassenbeschreibung (2)

Klasse*: Meldung* (Fortsetzung)

EinzahlungEinschieben () returns **BenutzerAntwort**

. . .

entnehmeKarte () returns **BenutzerAntwort**

. . .

entnehmeQuittung () returns **BenutzerAntwort**

. . .

entnehmeBargeld () returns **BenutzerAntwort**

. . .

Private Verpflichtungen:

Auf geeignete Benutzerreaktion warten

Genaue Subsystembeschreibung

Subsystem*: Interfacesubsystem*

Klassen:

abstrakt: *Input Device, Output Device, Display Device*, *Interaktion*

konkret: *Einzahlungseinzug, Tastatur, Bakkartenleser, Bargeldausgabe, Belegdrucker, Bildschirm, Formular, Menü, Meldung*

Kollaborationsgraph: Seite nn

Beschreibung: Dieses Subsystem implementiert die Schnittstelle zwischen einer ABS-Maschine und dem Bankkunden.

Verträge:

(4) Numerischen Wert einlesen - Anbieter: *Formular*

(6) Selektion aus einer Auswahl einlesen - Anbieter: *Menü*

(9) Meldung ausgeben und auf ein Ereignis warten - Anbieter: *Meldung*

Genaue Vertragsbeschreibung

Vertrag 9*:* **Meldung ausgeben und auf ein Ereignis warten**

Anbieter: *Meldung*

Klienten: *ABS, Bankkartenleser*, *Einzahlungstransaktion, Transaktion, Abhebungstransaktion*

Beschreibung: Dieser Vertrag unterstützt die Aufforderung an den Bankkunden, eine bestimmte Handlung auszuführen, wie etwa das Einschieben oder Entnehmen der Bankkarte.

Offene Punkte

- Anforderungsspezifikation

  Vor- und Nachteile objektorientierter Anforderungs-spezifikationen

- Wiederverwendung

  nicht explizit berücksichtigt

- Grosse Systeme

  Wie kommt man zur Gesamtarchitektur?
  .... zu einer Aufgabenteilung?

- Checklisten

  sehr allgemein gehalten