Software Engineering

Workshop OOA / OOD

Prof. Dr. Bodo Kraft

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Methodik (OO-Analyse)

Umgang mit den UML-Notationen

Bis jetzt hauptsächlich Syntax,
Verbindungen nur angedeutet.
Viele (auch firmenspezifische) Modelle.

→ Heute Vervollständigung: Vorgehensweise.

Methodik (OO-Analyse)

Beschreibungen in UML oder Checklisten verbreitet.

Hier eine Methode aus [Bal96], LE 13 (Seiten 359–391).

Das Anwendungsbeispiel

Bibliothekssystem

Soll **Ausleihe** unterstützen

- **AA:** Benutzer können Bücher ausleihen.
- **AZ:** Benutzer können ausgeliehene Bücher zurückgeben.
- **AF:** Die Leihfrist ist auf 4 Wochen beschränkt, dann folgen im 2-Wochen-Takt bis zu drei gebührenpflichtige Mahnungen, danach muss der Benutzer den Ersatz bezahlen.
- **AV:** Benutzer können die Leihfrist ausgeliehenen Bücher bis zu 2-mal um je 2 Wochen verlängern.
- **AR:** Leihberechtigung ruht, wenn ein Benutzer noch überfällige Bücher oder offene Rechnungen hat.

Das Anwendungsbeispiel

Soll **Vormerkungen** unterstützen:

- **VV:** Benutzer können Bücher vormerken.
- **Va:** Ausgeliehene vorgemerkte Bücher können nicht verlängert werden.
- **Vv:** Vorhandene vorgemerkte Bücher können nur vom Vormerker ausgeliehen werden.
- **VF:** Die Vormerkung ist auf 1 Woche befristet.
- **VN:** Vormerker werden über die Rückgabe ausgeliehener vorgemerkter Bücher informiert.

Das Anwendungsbeispiel

Soll **Nutzerverwaltung** unterstützen:

- **BN:** Neue Benutzer melden sich an.
- **BB:** Benutzer können sich abmelden (ausgeliehene Bücher, offene Rechnungen!).

B: Wird von Bibliothekaren (nicht Kunden) bedient.

H: Neue Hardware kann angeschafft werden.

O: Keine Festlegung zu Betriebssystem.

OO-Analyse: Übersicht (1/2)

10 Schritte zum OOA-Modell

1. **Klassen finden**
2. **Beziehungen und Aggregationen finden**
3. Klassenkomponenten finden:
   a) Attribute finden
   b) Externe Operationen finden
4. Objekt-Lebenszyklus erstellen
5. **Vererbungsstrukturen finden**

OO-Analyse: Übersicht (2/2)

6. Interne Operationen finden
7. Operationen spezifizieren

8. Vererbungsstrukturen prüfen
9. Beziehungen und Aggregationen prüfen

10. Subsysteme finden

Je nach Projektfluss auch **Vorgriffe** und **Rückgriffe** oder **Iterationen**.

Neben dem Kern-OO-Modell noch ein

- Glossar und eine
- Liste offener Fragen

führen.

OO-Analyse: Klassen finden

Klassen finden (Schritt 1)

Probleme

Worauf aufbauen?

Welche Klassen sind wichtig?

Wann sind alle wichtigen Klassen gefunden?

Im Beispiel nahe liegend: Buch, Benutzer, vielleicht Ausleihe – und sonst?

OO-Analyse: Klassen finden - Checkliste

1. Gibt es reale Objekte als Basis (Sensor, Drucker, …)?

2. Erste Liste von Klassen und Attributen (aus Formularen, Lastenheft, …)

4. Klassen kategorisieren (Konkrete Dinge, Personen und Rollen, Aktionsbeleg (Überweisung), Orte, Organisationen, Schnittstellen (Benutzer), Beziehungen (Mietvertrag), Informationsverbunde (Seminartyp)
   Sind die Klassen aussagefähig benennbar?

5. Stimmt das Abstraktionsniveau? (liegt DeviceDriver neben Kundenkartei?)

6. Gibt es 1:1-Beziehungen?

7. Nicht-Klassen entfernen (inhaltslos, Implementation, Attribut)

Allgemeines: keine Vererbung, eher kleine Klassen

OO-Analyse: Beziehungen und Aggregationen finden

Beziehungen und Aggregationen (Schritt 2)

Checkliste **Beziehungen**

1. Liegen relevante Beziehungen vor?
2. Sind die Klassen gleichrangig?
3. Müssen die Instanzen miteinander kommunizieren?
4. Welche Rollen spielen die Klassen?
5. Hat die Beziehung einen Namen?
6. Modelliert die Beziehung einen Schnappschuss oder Historie?
7. Welche Restriktionen muss die Beziehung erfüllen?
8. Existieren zwischen Klassen mehrere Beziehungen?
9. Gehören Attribute zu einer Beziehung?

OO-Analyse: Beziehungen und Aggregationen finden

Beziehungen und Aggregationen (Schritt 2)

Checkliste **Aggregationen**

1. Existieren Rangordnung und enger Zusammenhang?
2. Sind die Objekte physisch oder logisch?
3. Ist die Aggregation transitiv und asymmetrisch?
4. Gehört die Aggregation zu einer der üblichen Anwendungen?
   - Ganzes und Teile
   - Kollektion und Eintrag
   - Behälter und Inhalt
   - Konfiguration von Teilen
   - Ganzes und Unterteilungen
   - Ganzes und untrennbare Teile
   - Verbindung zwischen Teilen
5. Nicht doch Attribut oder Beziehung?

OO-Analyse: Beziehungen und Aggregationen finden

Beziehungen und Aggregationen (Schritt 2)

Checkliste **Kardinalitäten**

1. Muss-Beziehung?
2. Kann-Beziehung?
3. Ist die Obergrenze bekannt?
4. Gelten besondere Bedingungen?
   - Gerade Anzahl
   - Mindestens zwei
   - Entweder 2–3 oder mehr als 15

Systematisch sämtliche Beziehungen durchgehen.

OO-Analyse: Beziehungen und Aggregationen finden

Beziehungen und Aggregationen (Schritt 2)

- Checkliste **Beziehungen**
- Checkliste **Aggregationen**
- Checkliste **Kardinalitäten**

Systematisch sämtliche Beziehungen durchgehen.

OO-Analyse: Klassenkomponenten finden

Checkliste **Attribute**

1. Ist das Attribut problemrelevant?
2. Stimmen Perspektive und Abstraktion?
3. Modelliert das Attribut einen Schnappschuss oder Historie?
4. Gehört das Attribut zu einer Klasse oder Beziehung?
5. Teilen sich alle Instanzen einer Klasse das Attribut?
6. Hat das Attribut einen geeigneten Namen?
7. Ist das Attribut hier keines?
   - Nur Schlüssel
   - Nur Referenz (oder andere Entwurfsdetails)
   - Nur Zustand
   - ableitbar
8. Im Zweifel neue Klassen bilden.

Pro Attribut Name, Wertebereich, Beschreibung, Schlüssel?, Option?, Menge?, Defaultwert und Restriktionen festhalten.

OO-Analyse: Klassenkomponenten finden

Klassenkomponenten - Methoden (Schritt 3)

Checkliste Operationen

1. Was sind die Aufgaben?
2. Auf welche Ereignisse wird reagiert?
3. Wird die Operation mit einer Instanz ausgeführt?
4. Zu welcher Klasse gehört die Operation?
5. Hat die Operation einen geeigneten Namen?
6. Welche Daten benötigt die Operation, welche liefert sie?
7. Welche Qualitätskriterien gelten?
8. Ist die Operation hier keine? (Details, Implizit)

Hier zuerst nur die extern sichtbaren Operationen suchen.

OO-Analyse: Vererbungsstrukturen erstellen

Vererbungsstrukturen (Schritt 5)

Checkliste Vererbungen

1. Einfachvererbung? Allgemeinere Oberklassen oder spezialisierte Unterklassen entwickeln?
2. Mehrfachvererbung? Namenskonflikte? Verständlichkeit?
3. Abstrakte Klassen?
4. Fügt sich die Klasse gut ein? Ist die Vererbungsstruktur so gut?
5. Ist eine Klasse Ober- oder Unterklasse?
6. Kann ein Objekt seine Unterklasse wechseln?
7. Ist die Vererbung hier keine? (nur Typunterscheidung, nur eine Zahl, undefinierte Attribute)

OO-Analyse: Vererbung, Beziehungen und Aggregation prüfen

Prüfung der Architektur (Schritt 8 und 9)

- Noch mal Gesamtmodell betrachten
- Änderungen können Probleme eingeführt haben
- Annahmen können ungültig geworden sein
- Nach üblichen Mustern suchen, sie können bei Verständnis und Übertragung in Entwurf helfen.
- Checklisten Vererbung, Muster, Beziehungen, Aggregationen, Kardinalitäten

Subsysteme finden (Schritt 10)

Checkliste Subsysteme

1. Starke Bindung im Subsystem?
2. Schwache Koppelung zwischen Subsystemen?
3. bottom-up und top-down suchen.
4. Stimmt die Größe