Softwareentwicklung OOD Videothek

Ein mögliches Vorgehen bei OOD soll im Rahmen einer Softwareentwicklung an dem bereits bei der OOA verwendeten Beispiel einer Videothek exemplarisch vorgestellt werden.

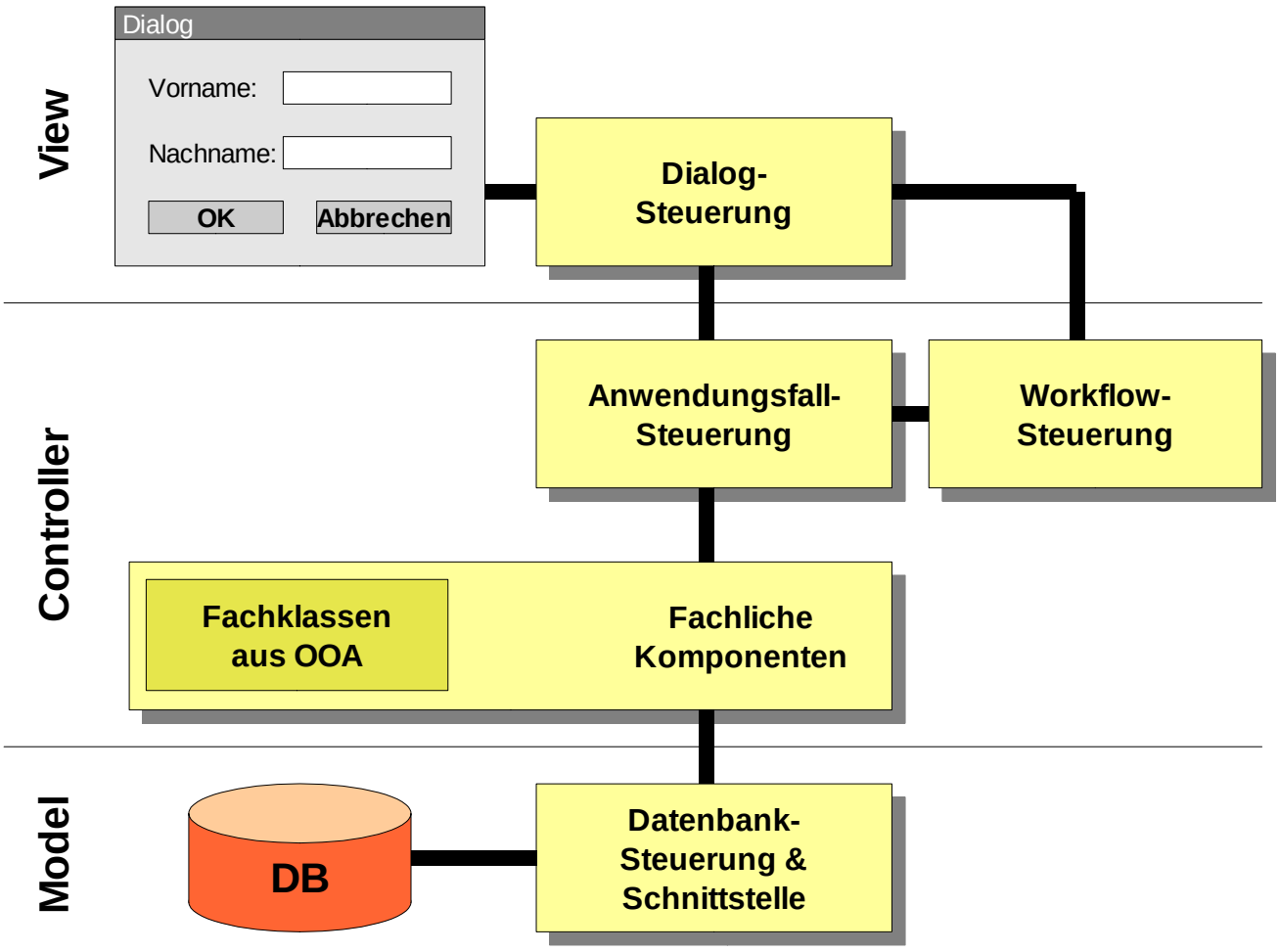
1. Festlegen der Programmarchitektur

Festlegen der Programmarchitektur
Die Struktur und der generelle Aufbau des Programms wird festgelegt z.B. MVC (Model-View-Controller)

Bei der Programmarchitektur sollten mindestens die drei Komponenten

- Model (Datenverwaltung)
- View (Benutzeroberfläche)
- Controller (Anwendung)

unterschieden werden. In der Regel ist es sogar sinnvoll noch eine genauere Aufteilung der Module vorzunehmen. Diese genauere Aufteilung könnte beispielsweise wie folgt aussehen:



Dialog

Formulare mit unterschiedlichsten Eingabekomponenten.

Dialogsteuerung

Reagiert auf Eingabeaktionen des Benutzers. Entsprechend den Event-Listenern unter Java. Oft werden Dialog und deren Dialogsteuerung zusammen behandelt, z.B. durch lokale Klassen. Dadurch wird jedoch das einfache und schnelle Austauschen und der neue Entwurf der Benutzeroberfläche erschwert.

Workflowsteuerung

Die Workflow-Steuerung initiiert, überwacht und steuert die Abarbeitung eines Geschäftsprozesses, der wiederum aus unterschiedlichen Anwendugsfällen bestehen kann und in der Regel auch bestehen wird. Eine typische Funktion, die zur Workflowsteuerung gehört, ist die main-Funktion, die z.B. ein übergeordnetes Auswahlmenü steuert und überwacht.

Anwendungsfallsteuerung

Eine Anwendungsfallsteuerung steuert die in einem Anwendungsfall bzw. dem dazugehörigen Aktivitätsmodell festgelegten Abläufe.

Fachliche Komponenten

Diese Komponenten repräsentieren den eigentlichen Anwendungsbereich. Sie beinhalten und kapseln die fachlichen Klassen (siehe OOA).

Datenbank-Steuerung und Schnittstelle

Auf eine Datenbank sollten die unterschiedlichen Komponenten bzw. Fachklassen niemals direkt zugreifen, sondern immer über die Funktionen bzw. Methoden dieser Schnittstelle. Dadurch kann die Unabhängigkeit und leichte Austauschbarkeit des RDMS gewährleistet werden.

2. Anwendungsfälle zusammenführen

Anwendungsfälle überprüfen und evtl. zusammenführen.

Evtl. Zusammenfassen von mehreren inhaltlich ähnlichen Anwendungsfällen. Oder gegebenenfalls eine gemeinsame Benutzeroberfläche für mehrere Anwendungsfälle festlegen.

Während der OOA wurden beispielsweise die Anwendungsfälle

- Kundendaten erfassen
- Kundendaten löschen
- Kundedaten ändern unterschieden.

Es ist zumindest eine Überlegung wert, ob man diese sehr ähnlichen Anwendugsfälle nicht sinnvoll zusammenfassen kann, so dass alle Anwendungsfälle mit nur einer Benutzeroberfläche und einer zentralen Anwendungsfallsteuerung auskommen.

Achtung

Die Benutzerfreundlichkeit sollte nicht dadurch leiden, dass einzelne Dialoge mit Funktionen und Schaltern überfrachtet werden. Gegebenenfalls kann es sinnvoller sein, mehrere gleichartige oder inhaltlich ähnliche Anwendungsfälle in einem eigenen Workflow mit entsprechenden Unterpunkten zusammenzufassen.

3. Entwurfsmuster (Design-Patterns)

Entwurfsmuster bestimmen
Ermitteln, ob für die gewünschte Umsetzung bzw. Problemstellung bereits Entwurfsmuster existieren und deren Anwendbarkeit überprüfen.

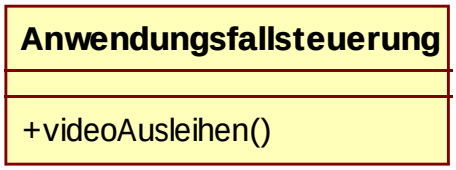
Es sollte überprüft werden, ob bei dem Zusammenspiel der einzelnen Komponenten auftretende Probleme durch bereits existierende Entwurfsmuster gelöst werden können. Um dies entscheiden zu können, ist natürlich die Kenntniss über die verschiedenen Entwurfsmuster zwingende Voraussetzung (siehe Entwurfsmuster).

4. Klassendiagramm erweitern

Klassendiagramm um Steuerkomponenten und Schnittstellen zu erweitern
Für die Umsetzung der Programmarchitektur notwendige Steuerklassen und Schnittstellen in das Klassenmodell aufnehmen.

Das Fachklassendiagramm aus der OOA muss nun um die Schnittstellen und Steuerklassen erweitert werden. Beispielsweise könnte in unserem Videothekprojekt eine Klasse Anwendungsfallsteuerung mit der Methode Videoausleihen (neben vielen anderen wie Listener, Main-Klasse, ...) hinzukommen.

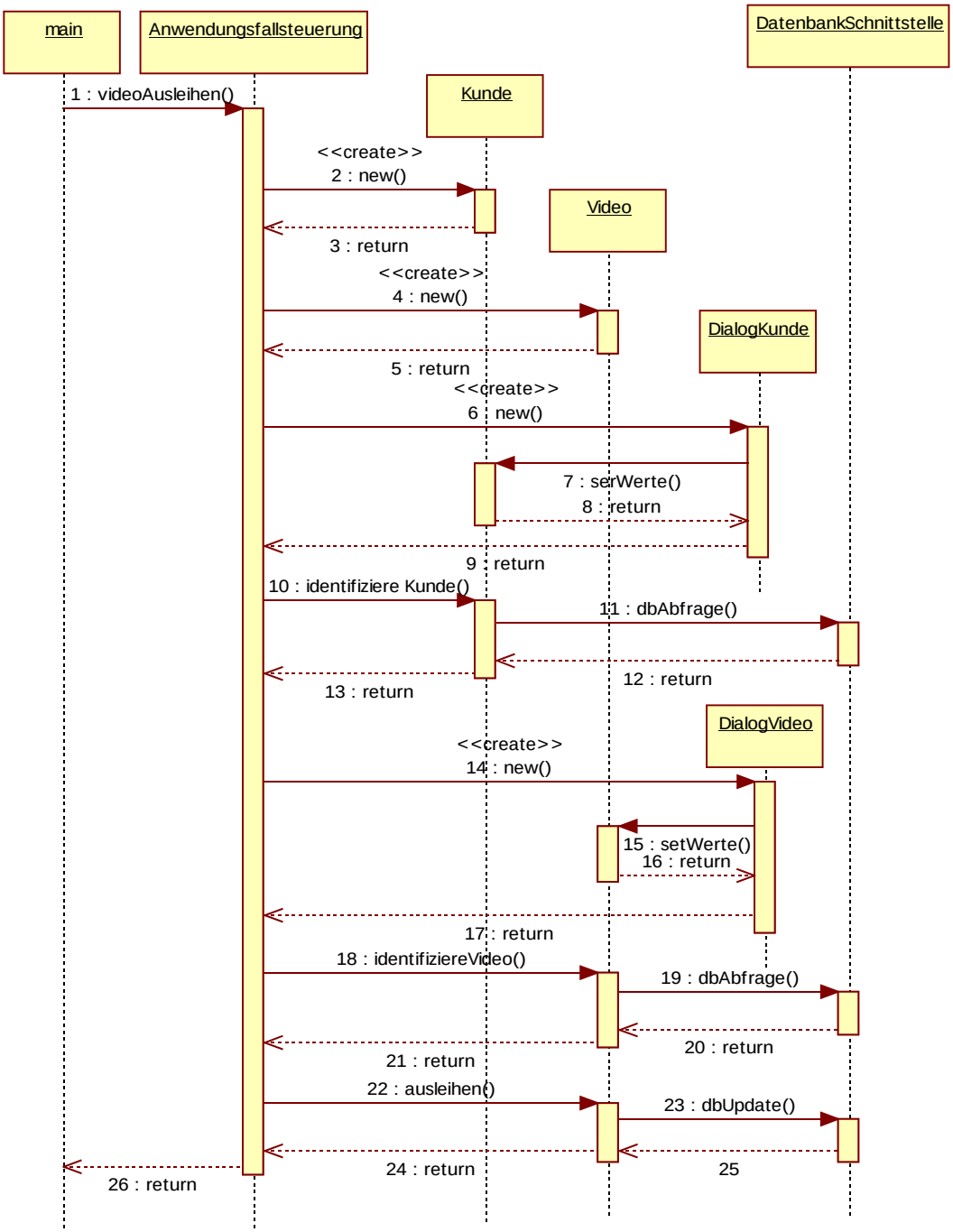
Eine von vielen zusätzlichen Klassen



5. Zusammenspiel überprüfen

Zusammenspiel zwischen Steuerklassen und Funktionsklassen überprüfen
Methoden und Assoziationen zwischen Steuerklassen und den Funktionsklassen anhand von Sequenzdiagramm überprüfen.

Das folgende Sequenzdiagramm veranschaulicht den Standardablauf zwischen dem Anwendungsfall „Video ausleihen“ und anderen Klassen. Es werden alle Nachrichten (Funktionsaufrufe) veranschaulicht und somit die Kommunikation zwischen den Klassen dargestellt.



Sprachspezifische Eigenarten berücksichtigen

Sprachspezifische Eigenarten berücksichtigen

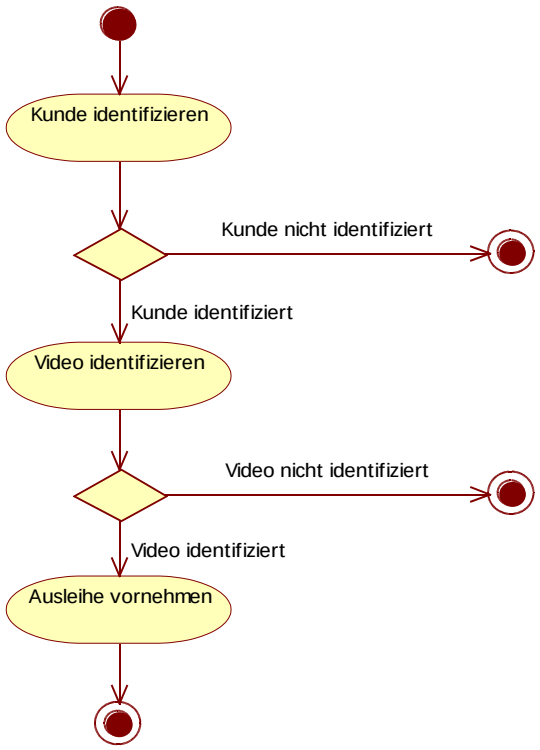
Da JAVA keine Mehrfachvererbung unterstützt, müssen beispielsweise Mehrfachvererbungen in Aggregationen transformiert werden.

Jede Programmiersprache hat ihre sezifischen Eigenheiten. Wie bei Java:

- Fehlende Mehrfachvererbung
- Event-Handling über Listener
- Exception-Handling
- ...

All diese Eigenheiten müssen bei der Umsetzung berücksichtigt werden.

Im folgenden Beispiel soll die Methode „Video ausleihen“ der Klasse „Anwendungsfallsteuerung“ beispielhaft realisiert werden. Dabei kann das zuvor erstellte Aktivitätsdiagramm (siehe OOA) sehr nützlich sein.




```
public class Anwendungsfallsteuerung
{
  public static void videoAusleihen()
  {
     Kunde kunde;                          // Kundenobjekt für Datenübergabe
     Video video;                          // Videoobjekt für Datenübergabe

     kunde = new Kunde();                  // Datenobjekte initialisieren
     video = new video();

     new DialogKunde (Kunde);              // Kundendaten eingeben
     if (Kunde.identifiziereKunde (kunde)) // Kunde wurde identifiziert
     {
        new DialogVideo (video);             // Videodaten eingeben (Rückgabe)
        if (Video.identifiziereVideo (video)) // Video wurde identifiziert
        {                                     // Video enthält Rückgabedatum
           video.ausleihen(kunde);            // Datensatz in Tabelle eintragen
        }
     }
  }
}
```