Übungen zur Vorlesung

Softwaretechnologie

- Wintersemester 2010/2011 -

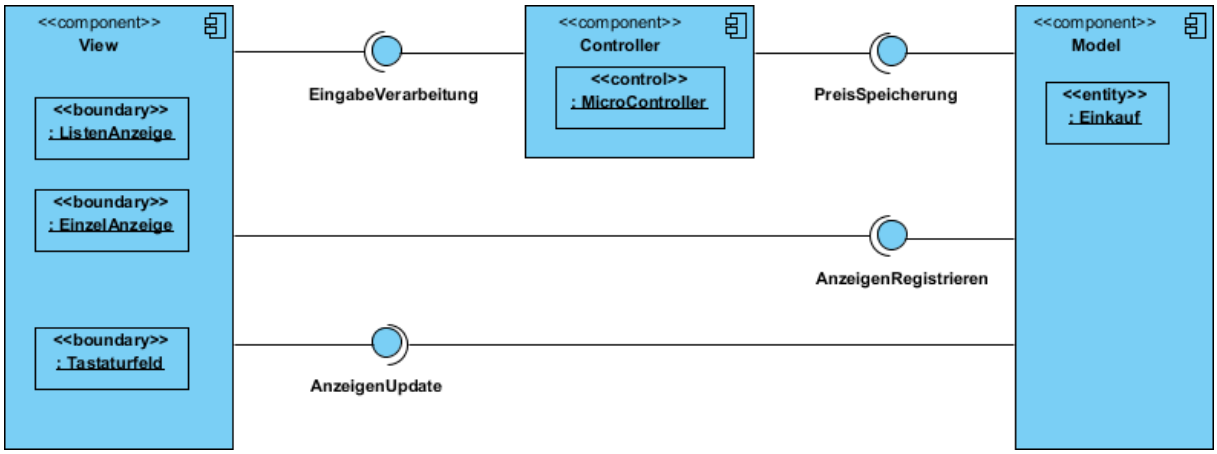
Dr. Günter Kniesel

Übungsblatt 9 - Lösungshilfe

Aufgabe 1. *Systementwurf* (7 Punkte)

Die Softwarefirma entscheidet sich, eine **Model-View-Controller-Architektur** für die Modellierung der Kasse zu verwenden, bei dem sich die Anzeigen beim Initialisieren am Modell zwecks Aktualisierungs-Benachrichtigung anmelden.

a) Überlegen Sie sich eine sinnvolle Gruppierung der Analyseklassen in Komponenten (Subsysteme) und definieren Sie Dienste. Zeichnen Sie ein passendes **Komponentendiagramm.**



b) Diskutieren Sie, wie die Wahl für die **Model-View-Controller-Architektur** folgende Entwurfsziele erfüllt oder verletzt:

- Erweiterbarkeit (z.B. neue „Views“)
  - Sehr gut, solange die Domäne nicht erweitert wird
  - Neue Views: problemlose Anmeldung
  - Neuer Controller: problemlose Einbindung
  - Erweiterung der Anwendungsdomäne
    - Erweiterung des model-Subsystems
    - Zur geeigneten Darstellung des neuen Wissens, müssten evtl. die bestehenden view-Subsysteme angepasst oder neue erzeugt werden.
    - Evtl. Anpassung Der Interaktionen zwischen Benutzer und model-Subsystem auf die neuen Daten
      - neue controller-Subsysteme hinzufügen bzw.
      - bestehende verändern.

- Reaktionszeit (Zeit zwischen einer Benutzereingabe und dem Abschluss der Aktualisierung aller Views)
  - Abhängig von System und Anforderungen
  - Für Echtzeitsysteme evtl. nicht sinnvoll
    - Zugriff auf die Domänendaten erfordert Zwischenschritte
  - Schlecht, wenn sich der Zustand des model-Subsystems oft ändert
    - viel Kommunikation zwischen den Sub-Systemen
  - Höhere Effizienz z.B. durch Zusammenfassen mehrerer Nachrichten zu einer
- Änderbarkeit (z.B. die Erweiterung des Modells um zusätzliche Attribute)
  - Sehr gut, solange sich die Domäne nicht ändert
  - Analog zum Entwurfsziel „Erweiterbarkeit“
  - Frage auch hier: Muss zur Erweiterung des Models (neue Attribute oder Operationen) sein Interface geändert werden?
- Zugriffs-Kontrolle (d.h. die Sicherstellung, dass nur berechtigte Benutzer auf bestimmte Teile des Modells zugreifen können)
  - Relativ gut erfüllt
  - Verantwortung für Aktionen auf Model liegt bei den Controllers
    - Controller sollte nur mit vertrauenswürdigen Nutzern interagieren
  - Verantwortung für Visualisierung des Models liegt bei Views
    - Falls nicht alle Änderungen visualisiert werden sollen kann man einen Controller zwischen Model und Views stellen
    - Sieht für Model wie ein View aus
    - Datenweitergabe nur an vertrauenswürdige Views

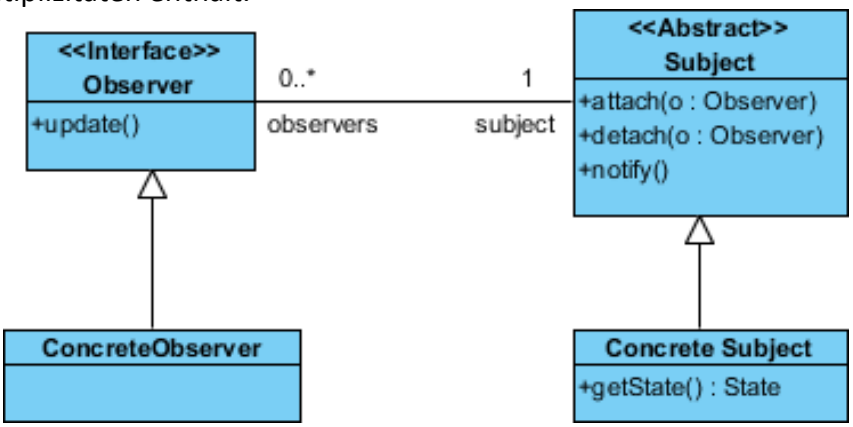
Aufgabe 2. *Entwurfsmuster* (4 Punkte)

In Aufgabe 2 sollten Sie auf eine zyklische Abhängigkeit zwischen der View- und der Controller-Komponente gestoßen sein.

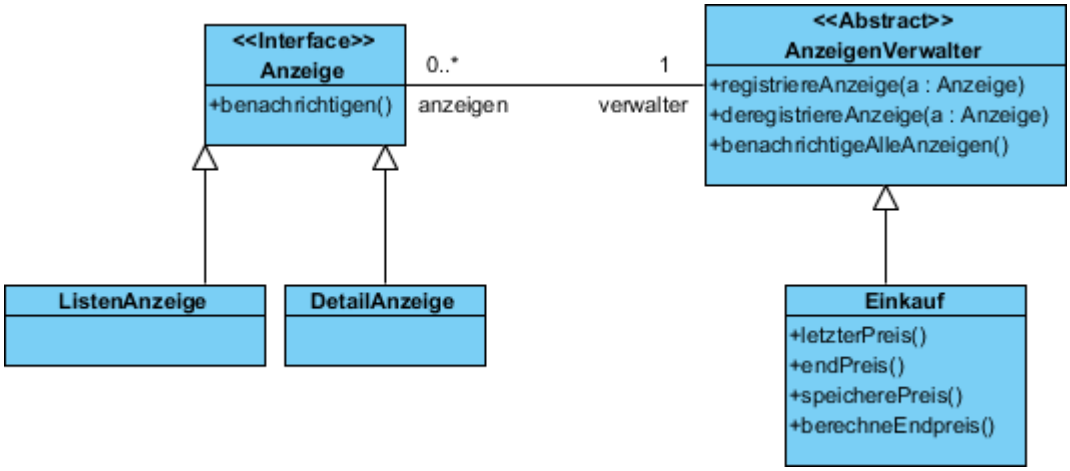
a) Mit welchem Entwurfsmuster können Sie – ohne die Kommunikation zu ändern – eine daraus folgende Abhängigkeit auf Klassenebene vermeiden?

- Observer-Pattern (Beobachter)

b) Wie ist dieses Pattern **im Allgemeinen** aufgebaut? Zeichnen Sie ein entsprechendes Klassendiagramm, das die wichtigsten Klassen und Methoden sowie Assoziationen und Multiplizitäten enthält.



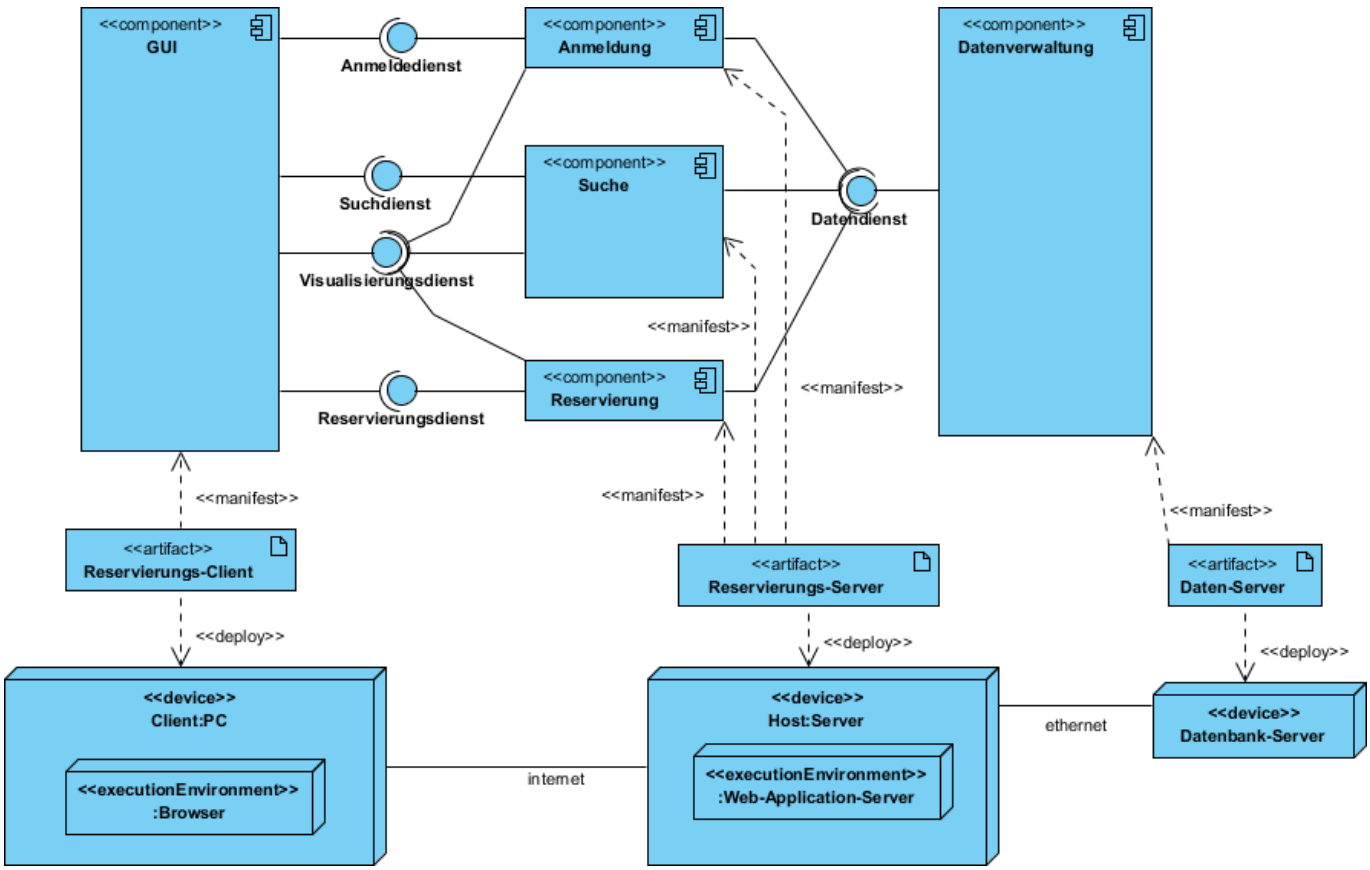
c) Wie lässt sich das Pattern **im konkreten Fall der Situation aus Aufgabe 1** einsetzen? Zeichnen Sie, analog zu Teilaufgabe b, ein Klassendiagram und halten Sie die Bezeichnungen konsistent zu Ihren Lösungen in Aufgabe 1.



Aufgabe 3. *Deployment* (8 Punkte)

Zu Beginn des Projektes stellt Ihnen die auftraggebende Universität einen Server, eine Schnittstelle zum Internet sowie eine Studierenden-, Seminar- und Reservierungs-Datenbank zur Verfügung. Studierende sollen ihr System per Browser via Internet nutzen.

a) Zeichnen Sie ein Verteilungsdiagramm (engl. „Deployment-Diagram“), das die Hardware-Software-Zuordnung wiedergibt. **Hinweis**: Die Elemente von Verteilungsdiagrammen sind im Foliensatz zum Kapitel Systementwurf erläutert.



b) Die folgenden Entwurfsziele sind in unserer Anwendung zu beachten:

   a. Nebenläufigkeit unabhängiger Workflows
   b. Zentrale Verwaltung der Seminardaten
   c. Schutz sensibler Daten der Studierenden
   d. Zeitweise hohe gleichzeitige Zugriffszahlen

   Welche der in der Vorlesung vorgestellten Architekturen machen in diesem Fall Sinn? Begründen Sie jeweils ihre Wahl.

- Model-View-Controller (MVC)
  - Model: Subsystem „Datenverwaltung“
  - View: Subsystem „GUI“
  - Controller: „Anmeldung“, „Suche“, „Reservierung“

- Repository Architektur:
  - Für viele unabhängige Nutzer, die nichts von einander wissen sollten
  - Zur Nutzung eines DBMSs
  - zentrale Lagerung des Domänenwissens vereinfacht den Umgang mit Nebenläufigkeit und Integrität.

- Client-Server Architektur
  - Ist Spezialfall der Repository Architektur
  - Server entspricht dem Repository
  - Typisch: remote procedure call Mechanismus
  - Unterschied zu „normalem Repository“
  - Kontrollflüsse auf Client und Server bis auf Synchronisation für Anfragen unabhängig

Aufgabe 4. *Persistenz* (5 Punkte)

a) Welche Möglichkeiten des Datenmanagements kennen Sie aus der Vorlesung? Wie werden diese realisiert? Aufgrund welcher Anforderungen entscheiden Sie sich für die jeweilige Möglichkeit?

- Persistieren
  - Datenbank
    - Verschiedene Detailstufen der Daten für viele Nutzer
    - Heterogenes System (verschiedene Plattformen)
    - Verschiedene Anwendungen, die auf die Daten zugreifen
    - Komplexe Infrastruktur für das Datenmanagement
      - Sicherheit
      - Zuverlässigkeit
      - Hoher Durchsatz

- (Temporäre) Dateien

- Große Daten
- „Raw“ Daten
- Nur kurzzeitige Speicherung von Daten
- Niedrige Informationsdichte

- Nicht Persistieren
  - Datenstruktur im Hauptspeicher

b) Welche Elemente von Aufgabe 3 müssen persistent sein, welche nicht? Warum?

- Immer wieder benötigte Elemente (Use-Case-Übergreifend):
  - Seminar
  - Reservierung
  - Studierende

⇨ **Persistieren**

- Nur temporär benötigt:
  - Controller und GUI-Elemente

⇨ **Nicht Persistieren**