MVC-Architektur am Beispiel von OLAT

Marvin Frommhold

12. Januar 2009

Gliederung

Begriffe I

MVC

- bezeichnet ein Architekturmuster zur Strukturierung von Software und der Entwicklung in drei Einheiten:
  - *Datenmodell* (**M**odel)
  - *Präsentation* (**V**iew)
  - *Programmsteuerung* (**C**ontroller)

Begriffe I

MVC

- bezeichnet ein Architekturmuster zur Strukturierung von Software und der Entwicklung in drei Einheiten:
  - *Datenmodell* (**M**odel)
  - *Präsentation* (**V**iew)
  - *Programmsteuerung* (**C**ontroller)
- **Ziel:** ein flexibler Programmentwurf, der u. A. eine spätere Änderung oder Erweiterung erleichtert
- Wiederverwendbarkeit der einzelnen Komponenten, leichter Austausch dieser

Begriffe II

Modell:

- enthält die Darzustellenden Daten und meist auch deren Verarbeitung (Berechnungen), die Geschäftslogik
- Änderungen werden nach dem Entwurfsmuster „Beobachter“ bekanntgegeben

Begriffe II

Modell:

- enthält die Darzustellenden Daten und meist auch deren Verarbeitung (Berechnungen), die Geschäftslogik
- Änderungen werden nach dem Entwurfsmuster „Beobachter“ bekanntgegeben

Präsentation:

- Darstellung der jeweiligen Daten aus dem Modell
- Entgegennahme von Benutzerinteraktionen
- verwendet meist das Entwurfsmuster „Kompositum“

Begriffe III

Steuerung:

- verwaltet eine oder mehrere Präsentationen
- nimmt deren Benutzerinteraktionen entgegen, wertet diese aus und reagiert entsprechend
- auf Grund der Aktionen entscheidet die Steuerung, welche Daten im Modell verändert werden müssen
- verwendet Entwurfsmuster „Strategie“

Entwurfsmuster I

Observer: (Beobachter)

- *Problem:* eine/mehrere Komponenten stellen Zustand eines Objekts aus dem Modell dar, diese kennen gesamte Schnittstelle dieses Objekts; andererseits soll das Objekt von den Komponenten unabhängig bleiben, also deren Schnittstellen nicht kennen

Entwurfsmuster I

Observer: (Beobachter)

- *Problem:* eine/mehrere Komponenten stellen Zustand eines Objekts aus dem Modell dar, diese kennen gesamte Schnittstelle dieses Objekts; andererseits soll das Objekt von den Komponenten unabhängig bleiben, also deren Schnittstellen nicht kennen
- *Lösung:* das beobachtete Objekt stellt Mechanismus bereit, über den Beobachter sich an-/abmelden können und diese vom Objekt über Änderungen informiert werden; das Objekt muss somit nur eine einheitliche Schnittstelle seiner Beobachter kennen

Entwurfsmuster II

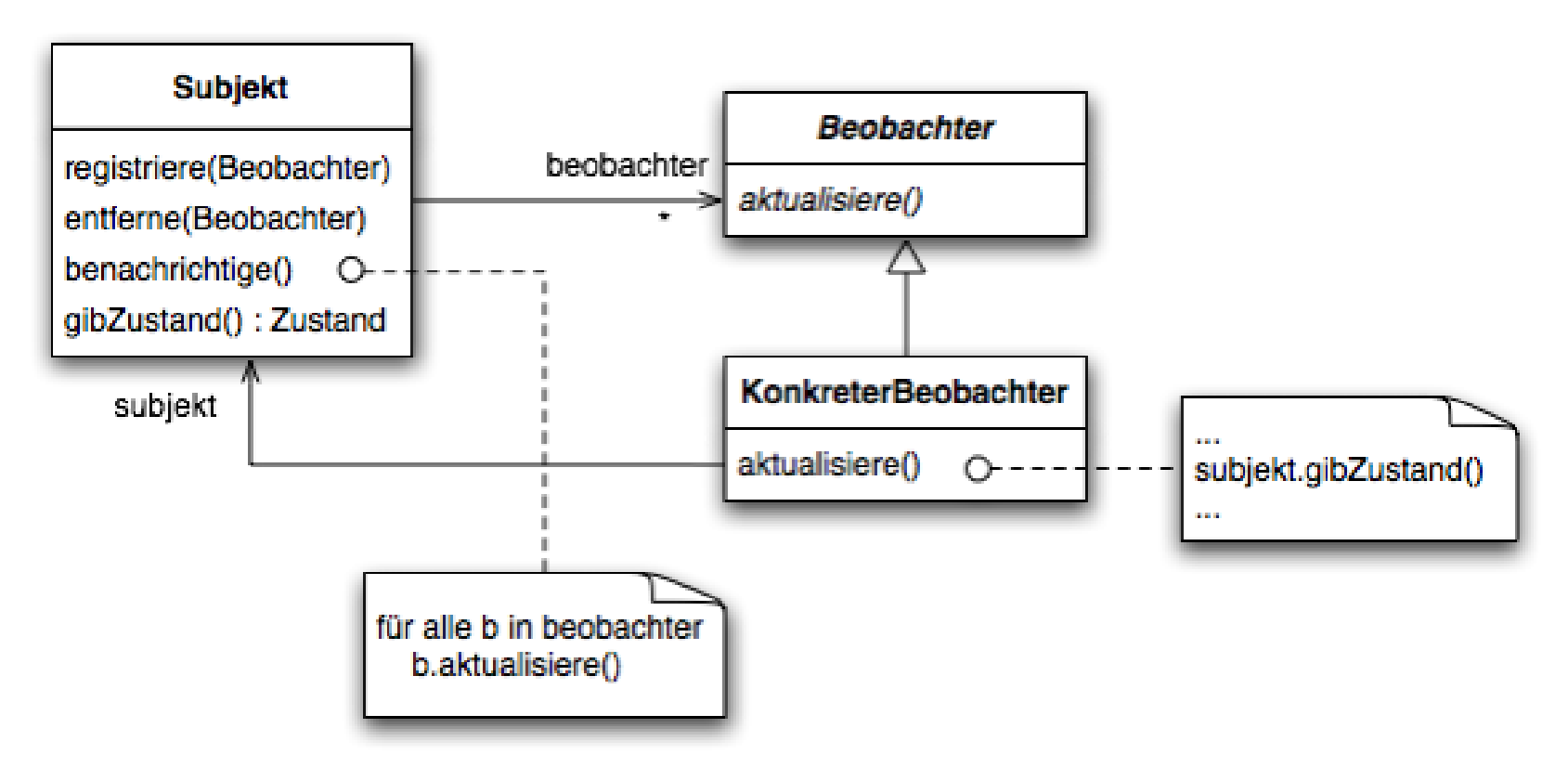


Abbildung: http://de.wikipedia.org/w/index.php?title=Datei:Beobachter-pattern.png&filetimestamp=20080305110419

Entwurfsmuster III

Kompositum:

- *Grundidee:* Repräsentation von primitiven Objekten sowie deren Behälter in einer abstrakten Klasse; somit können einzelne Objekte, als auch ihre Kompositionen einheitlich betrachtet werden

Entwurfsmuster III

Kompositum:

- *Grundidee:* Repräsentation von primitiven Objekten sowie deren Behälter in einer abstrakten Klasse; somit können einzelne Objekte, als auch ihre Kompositionen einheitlich betrachtet werden
- *Verwendung:* Implementierung von Teil-Ganzes-Hierarchien; Verbergen der Unterschiede zwischen einzelnen und zusammengesetzten Objekten

Entwurfsmuster IV

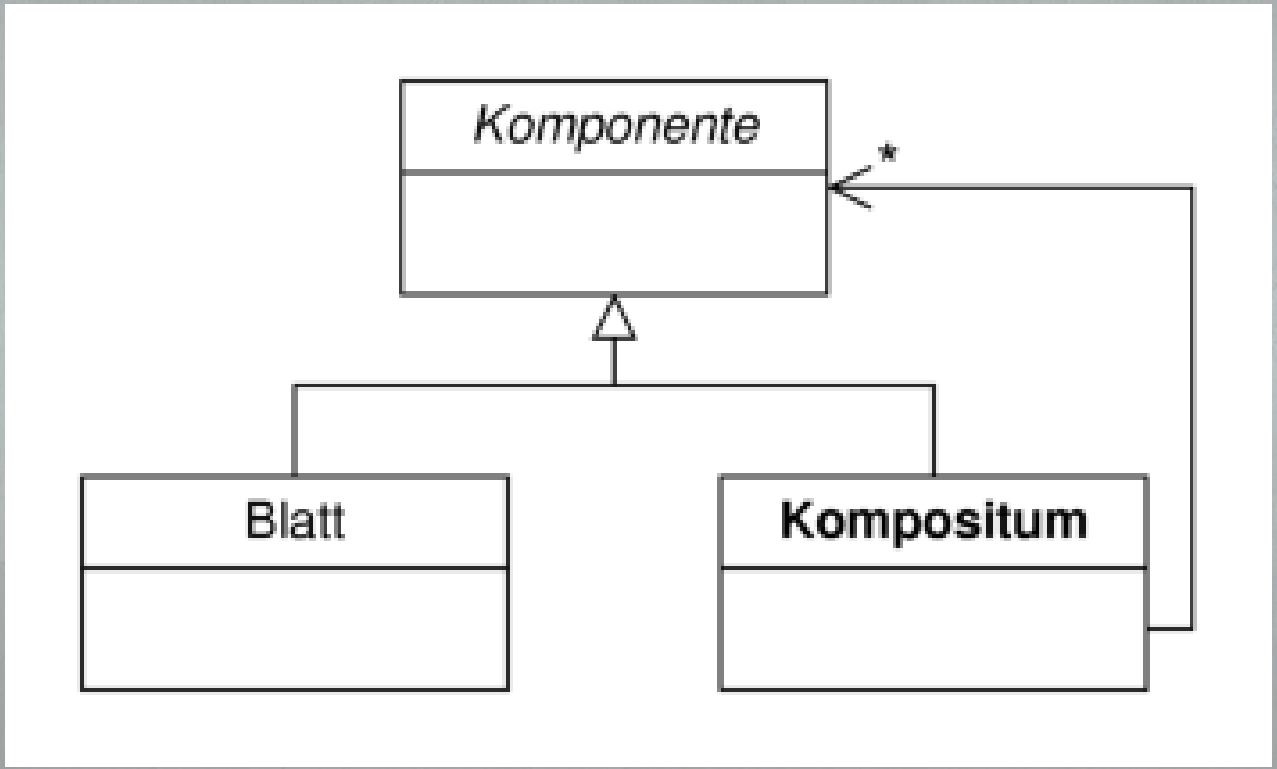


Abbildung: http://de.wikipedia.org/w/index.php?title=Datei:Kompositum_Klassen.png&filetimestamp=20060407095951

Entwurfsmuster V

Strategie:

- Umsetzung meist durch eine Klasse, die eine bestimmte Schnittstelle implementiert

Entwurfsmuster V

Strategie:

- Umsetzung meist durch eine Klasse, die eine bestimmte Schnittstelle implementiert
- *Anwendung:*
  - viele verwandte Objekte unterscheiden sich nur im Verhalten
  - unterschiedliche (austauschbare) Varianten eines Algorithmus werden benötigt

Entwurfsmuster VI

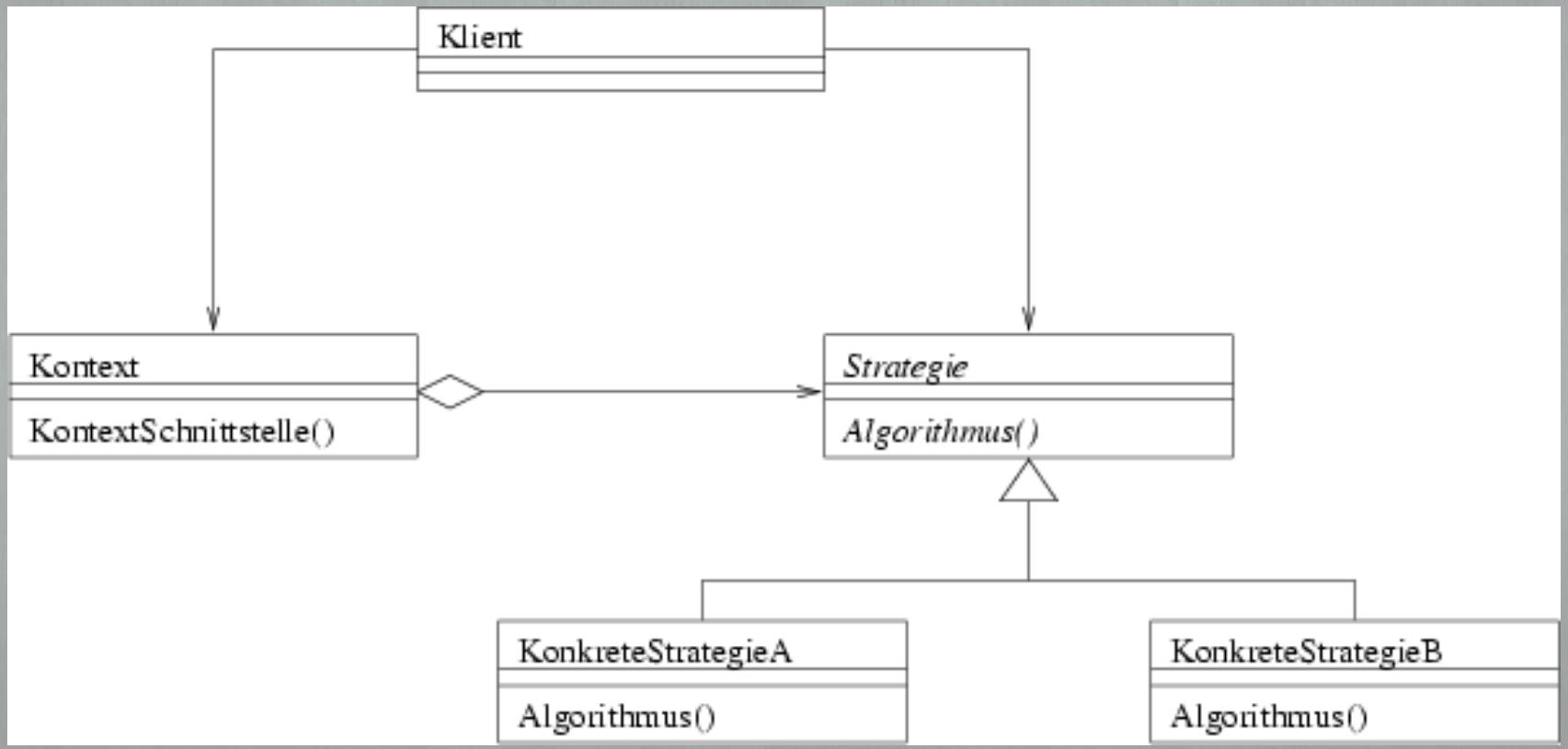


Abbildung: http://de.wikipedia.org/w/index.php?title=Datei:Strategie.png&filetimestamp=20050307081336

Umsetzung in OLAT

OLAT:

- **O**nline **L**earning **A**nd **T**raining
- web-basiertes Learning Management System (LMS)
- basiert auf Java, Open Source
- Entwicklung an der Universität Zürich

- aus dieser Entwicklung heraus entstand das Brasato-Framework

Aufbau des Brasato-Frameworks I

- im Brasato-Framework sind die beiden Schichten „Model“ und „View“ sehr eng miteinander verbunden
- werden in einer Klasse zusammengefügt, aber durch verschiedene darunterliegende Klassen und Dateien repräsentiert, dadurch wird wieder eine Trennung der beiden Schichten ermöglicht

Aufbau des Brasato-Frameworks I

- im Brasato-Framework sind die beiden Schichten „Model“ und „View“ sehr eng miteinander verbunden
- werden in einer Klasse zusammengefügt, aber durch verschiedene darunterliegende Klassen und Dateien repräsentiert, dadurch wird wieder eine Trennung der beiden Schichten ermöglicht

Modell/View

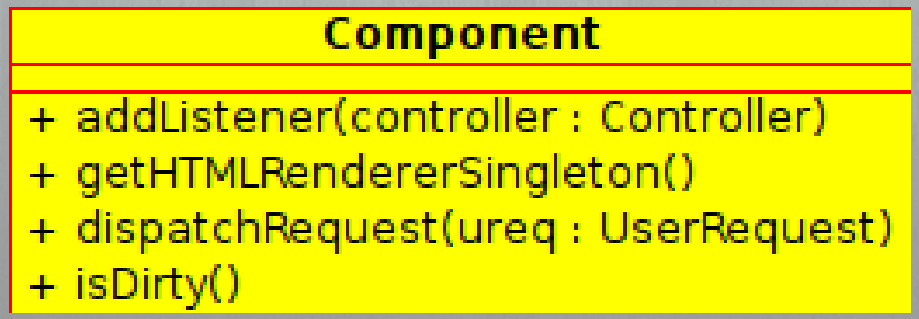
- `org.olat.core.gui.components.Component`
- Vereint die Entwurfsmuster „Observer“ und „Kompositum“
- Schnittstelle zum Anmelden von Überwachern (Controller)
- Schnittstelle zum Benachrichtigen der Überwacher

Aufbau des Brasato-Frameworks II

- abstrakte Basisklasse aller zu präsentierenden Objekte (Kompositum)
- die Darstellung der Objekte geschieht durch `org.olat.core.gui.components.ComponentRenderer`

Aufbau des Brasato-Frameworks II

- abstrakte Basisklasse aller zu präsentierenden Objekte (Kompositum)
- die Darstellung der Objekte geschieht durch `org.olat.core.gui.components.ComponentRenderer`



Aufbau des Brasato-Frameworks III

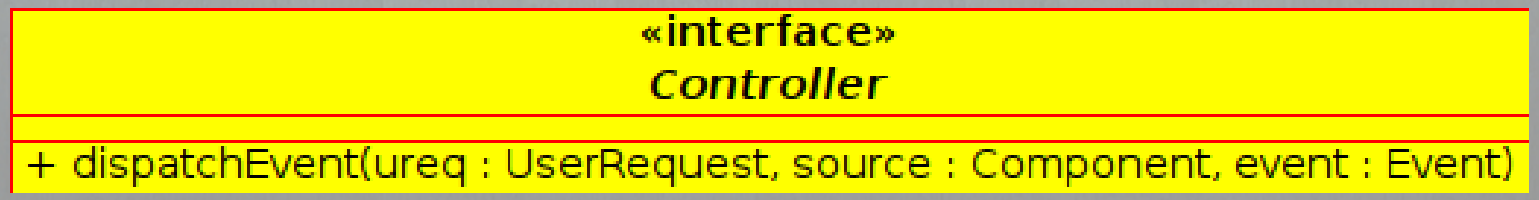
Contoller:

- `org.olat.core.gui.control.Controller`
- verwendet das Entwurfsmuster „Strategie“
- einheitliche Schnittstelle, welche Information über das überwachte Objekt und die zugehörige Aktion erhält

Aufbau des Brasato-Frameworks III

Contoller:

- `org.olat.core.gui.control.Controller`
- verwendet das Entwurfsmuster „Strategie“
- einheitliche Schnittstelle, welche Information über das überwachte Objekt und die zugehörige Aktion erhält



Quellen

- Wikipedia, Die freie Enzyklopädie: *Model View Controller*, Wikimedia Foundation Inc.

Vielen Dank für Eure Aufmerksamkeit!