Software Design Patterns

Einführung

SDP: Organisatorisches

- Abteilung Betriebliche Informationssysteme, Prof. Fähnrich
- Seminar im Rahmen des Modules 'Betriebliche Informationssysteme'
- Seminarleiter: Frank Schumacher, Axel Ngonga, Martin Gebauer

SDP: Organisatorisches

Seminarziel:

Vermittlung von Software Design Patterns

- Prinzip von Patterns verstehen
- Überblick über gebräuchlichste Pattern
- Strukturen von Patterns auf reale Problemstellungen übertragen
- Patterns in der Softwareentwicklung in geeigneter Form einsetzen

SDP: Organisatorisches

- Umfang: 2 SWS
- Ort und Zeit: mittwochs **13:15**-14:45 **Jo 1-22**
- Seminarablauf (abhängig von Teilnehmerzahl)
  - Vortrag eines Seminarteilnehmers zu den ausgewählten Pattern (ca. 45 min)
  - Diskussion und Fragestellungen (ca. 15 min)
  - Entwurf einer einfachen Anwendung des vorgestellten Patterns in Gruppenarbeit (ca. 10 min)
  - Vorstellung des Entwurfes (ca. je 5 Minuten)

SDP: Organisatorisches

- Mitzubringende Voraussetzungen
  - Objektorientierte Entwicklung
  - C#, C++ oder Java
- Voraussetzung für Scheinerhalt
  - Teilnahme an den Präsenzveranstaltungen
  - Vortrag von ca. 45 min (Note: 1/4)
  - Hausarbeit zum gewählten Thema (Note: 2/4)
  - Beispielhafte Implementierung (Note: 1/4)

SDP: Organisatorisches

- **Master neu**: benoteter Schein, der in die Modulnote mit einfließt.
- **Diplom**: benoteter oder unbenoteter Problemseminarschein
- **Master alt, Bachelor neu + alt**: benoteter oder unbenoteter Seminarschein

SDP: Organisatorisches

- Einführung, Themenverteilung
- Allgemeine Prinzipien des Softwaredesigns

- Vorbereitungszeit

- **Studentenvorträge zu den gewählten Themen**

- Abgabe der Hausarbeit und des Beispielprogramms

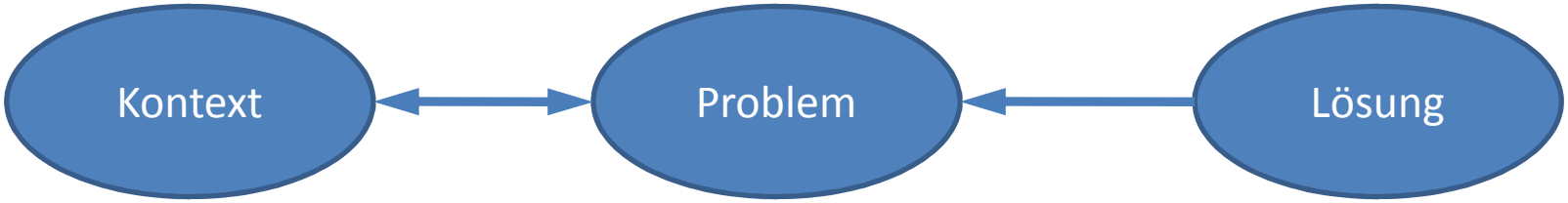
SDP: Organisatorisches

- Anmeldung zum Seminar per Mail beim Seminarleiter (fs@informatik.uni-leipzig.de)
  - Name
  - Matrikel
  - Studiengang (Diplom, Master-Neu, …)
  - Gewünschtes Thema (auch mehrere)

SDP: Organisatorisches

Fragen?

Patterns - Muster



- Kontext: Eine Situation ergibt ein Problem
- Problem: Das akute Problem im gegebenen Kontext
- Lösung: Eine bewährte Lösung für das Problem

Kontext

- Beschreibung der Situation, in der das Problem auftritt.
  - "Entwicklung einer Software mit einer GUI"
- Vage, damit ist es schwierig, den genauen Kontext eines Patterns zu bestimmen
  - mehrere Situationen möglich
  - möglicherweise nicht bedachte Situationen
- dient zur Unterstützung der Patternauswahl

Problem

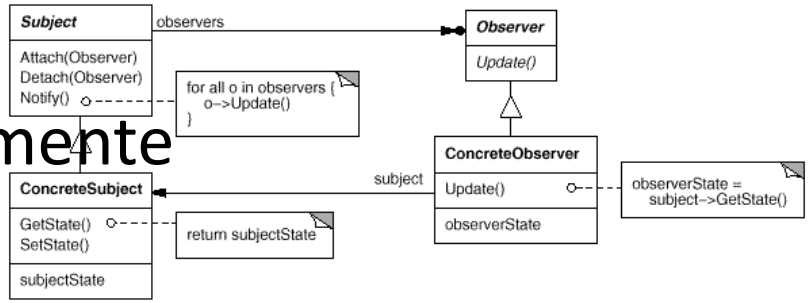
- beschreibt das wiederkehrende Problem im gegebenen Kontext
  - Was ist das genaue Design Problem, das gelöst werden muss?
- Aspekte des Problems, die für die Lösung des Problems beachtet werden müssen
  - Kräfte (***force***)

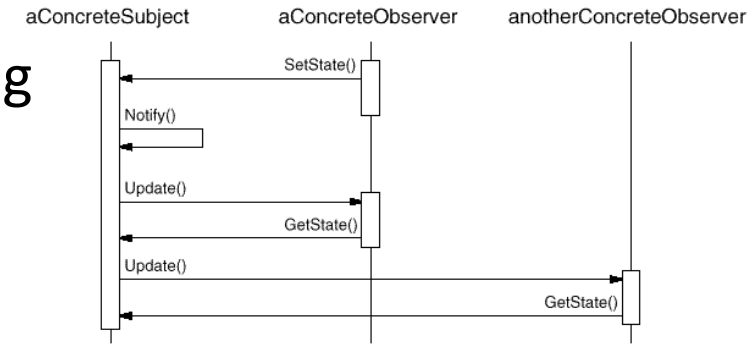
Problem

- ***forces***
  - Anforderungen (Requirements)
    - "peer-to-peer inter-process Kommunikation muss effizient sein"
  - Einschränkungen (Constraints)
    - "die inter-process Kommunikation muss ein bestimmtes Protokoll verwenden"
  - Eigenschaften (Properties)
    - "Software sollten leicht zu ändern sein"

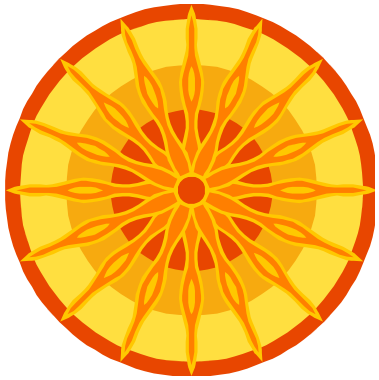
Lösung

- Zeigt ein Lösungsschema für ein spezielles Problem
- **oder**: der Ausgleich der zugeordneten Kräfte
- 2 Aspekte
  - Struktur der beteiligten Elemente
    - statische Aspekte der Lösung
  - Laufzeitverhalten
    - dynamische Aspekte der Lösung



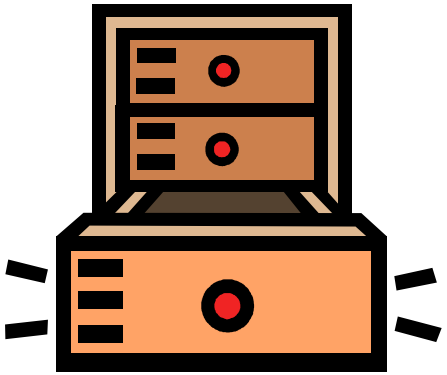


Muster für Muster



- Muster
  - Kontext
    - Designsituation, die ein Problem hervorruft
  - Problem
    - Menge an Kräften, die wiederholt im gegebenen Kontext auftreten
  - Lösung
    - Balancieren der Kräfte
      - Struktur mit Komponenten und Beziehungen zwischen diesen
      - Laufzeitverhalten

Pattern Kategorien



- Architekturmuster
  - fundamentales, strukturelles Organisationsschema eines Softwaresystems
- Designmuster
  - Schema zur Beschreibung der Komponenten oder der Beziehung zwischen diesen Komponenten
- Idioms
  - Low-Level Muster für spezifische Programmiersprache

Pakete von Patterns zur Bearbeitung

- Paket *Fabrik*
  - Factory Method
    - definiert eine Schnittstelle zur Erstellung eines Objektes ohne zu entscheiden, welche konkreten Objekttypen erstellt werden
  - Abstract Factory
    - Ermöglicht einem Client Familien von Objekten zu erstellen, ohne die konkrete Klasse anzugeben
  - Prototyp
    - Erzeuge neue Objekte durch Kopieren des Prototypen

Pakete von Patterns zur Bearbeitung

- Paket *Zusammensetzung*
  - Iterator
    - bietet eine Möglichkeit, auf die Elemente in einem aggregierten Objekt sequenziell zuzugreifen
  - Composite
    - ermöglicht die Zusammensetzung von Objekten zu einer Baumstruktur und die Gleichbehandlung individueller und zusammengesetzter Objekte
  - Flyweight
    - Nutzt Objekte kleinster Granularität zum Verwenden großer Mengen dieser Objekte

Pakete von Patterns zur Bearbeitung

- Paket *Anpassung*
  - Adapter
    - konvertiert die Schnittstelle einer Klasse in die vom Client erwartete Schnittstelle
  - Facade
    - bietet eine vereinheitlichte Schnittstelle für eine Menge von Schnittstellen eines Basissystems
  - Bridge
    - Entkoppelt Abstraktion von der Implementierung

Pakete von Patterns zur Bearbeitung

- Paket *Servicevariation*
  - Template Method
    - definiert das Gerüst / die Struktur eines Algorithmus und überlässt einige Schritte den Unterklassen
  - Strategy
    - Definiert und kapselt eine Familie von Algorithmen und ermöglicht eine Variierung unabhängig vom Client
  - State
    - ermöglicht einem Objekt, sein Verhalten zu ändern, wenn sein interner Zustand sich ändert.

Pakete von Patterns zur Bearbeitung

- Paket *Kommunikation*
  - Observer
    - definiert eine Eins-zu-viele-Abhängigkeit, so dass alle abhängigen Objekte automatisch benachrichtigt werden, wenn sich der Zustand des einen Objektes ändert
  - Chain of Responsibility
    - Entkoppelt den Auslöser einer Anfrage vom Empfänger
  - Mediator
    - Kapselt das Zusammenwirken mehrerer Objekte

Pakete von Patterns zur Bearbeitung

- Paket *Kommando*
  - Command
    - kapselt einen Auftrag (Funktionsaufruf) als ein Objekt
  - Command-Processor
    - Zusätzliche Funktionalität für Command-Objekte wie Speicherung und Wiederherstellung
  - Visitor
    - kapselt auszuführende Operationen als Objekt zur Ermöglichung der Definition neuer Operationen, ohne die Klassen der bearbeiteten Elemente zu ändern

Pakete von Patterns zur Bearbeitung

- Paket *Zugriff*
  - Decorator
    - Fügt einem Objekt dynamisch zusätzliche Verantwortlichkeiten (Funktionalität) hinzu
  - Proxy
    - kontrolliert den Zugriff auf ein Objekt mit Hilfe eines vorgelagerten Stellvertreterobjekts
  - Master-Slave
    - verteilt Arbeit an identische Subprozesse und aggregiert aus deren Ergebnissen ein Endresultat

Pakete von Patterns zur Bearbeitung

- Paket *Sonstige*
  - Singleton
    - sichert, dass es nur eine Instanz einer Klasse gibt und bietet einen globalen Zugriffspunkt für diese Instanz
  - Memento
    - Externalisiert den internen Zustand eines Objektes, ohne die Kapselung zu verletzen
  - Interpreter
    - Repräsentation der Grammatik einer gegebenen Sprache

Pakete von Patterns zur Bearbeitung

- Paket MVC
  - das Model-View-Controller Pattern ist ein zusammengesetztes Pattern
  - Kapselt Anzeige, Modell und Steuerung

Pakete von Patterns zur Bearbeitung

- Paket *Ereignissbearbeitung*
  - Proactor
    - Behandlung von Ereignissen asynchroner Operationen
  - Asynchronous Completion Token
    - Effiziente Bearbeitung der Antworten aufgerufener asynchroner Operationen
  - Acceptor-Connector
    - Entkoppelung der Verbindung und Initialisierung von peer-to-peer Anwendungen von der eigentlichen Verarbeitung

Pakete von Patterns zur Bearbeitung

- Paket *Synchronisation und Adaption*
  - Strategized Locking
    - parametrisiert Synchronisationsmechanismen
  - Thread-Safe Interface
    - Minimiert locking overhead und verhindert selbsterzeugte deadlocks
  - Reflection
    - Mechanismus zur dynamischen Änderung von Struktur und Verhalten einer Software und zur Erkundung von Objekten zur Laufzeit

Pakete von Patterns zur Bearbeitung

- Paket *Nebenläufigkeit*
  - Active Object
    - Entkoppelt Methodenausführung vom Methodenaufruf zur Vereinfachung der Zugriffssynchronisierung
  - Monitor Object
    - synchronisiert nebenläufige Methodenausführung zur Sicherstellung, dass nur eine Methode auf dem Objekt gleichzeitig ausgeführt wird
  - Leader / Followers
    - Verwaltung mehrerer Threads

Pakete von Patterns zur Bearbeitung

- Paket *Message Transformation*
  - Message Translator
    - Wie können Systeme die verschiedene Datenformate verwenden miteinander über Nachrichten kommunizieren?
  - Content Enricher
    - Wie können Systeme miteinander kommunizieren, wenn der Absender nicht alle benötigten Daten hat?
  - Normalizer
    - Wie können Nachrichten verarbeitet werden, die semantisch äquivalent sind aber in verschiedenen Formaten ankommen?

Pakete von Patterns zur Bearbeitung

- Paket *Message Routing*
  - Content based Router
    - Wie können Implementierungen behandelt werden, bei denen die Implementierung einer einzelnen Funktion über mehrere physikalische Systeme verteilt ist.
  - Aggregator
    - Wie können Resultate individueller aber zusammengehöriger Nachrichten als Ganzes verarbeitet werden.
  - Resequencer
    - Wie kann ein Strom von Nachrichten mit gestörter Reihenfolge wieder korrekt geordnet werden?

Lernen

Foto: Stephen W. Oachs

Ignorieren

Bild: Gerd Altmann, direct-dialog

Lernen zu lernen

- man erwartet, dass wir lernen, ohne uns beizubringen, wie man lernt
- Metakognition

- Verstehen > Auswendig lernen

- Wie bringt man das Gehirn dazu, den Lernstoff als etwas wichtiges anzusehen?

Software Design Pattern = Tiger?

- repeat, repeat, repeat .... repeat
- oder: Gehirnaktivität erhöhen!
  - Text IN Abbildungen
  - Sprachstil: Gespräch > Vortrag
  - Redundanz: dasselbe auf unterschiedliche Art und Weise ausdrücken
  - Überraschung, Belustigung, Interesse: Emotionen helfen sich zu erinnern
  - Herausforderungen: Mitdenken erwünscht

Besser lernen

- Langsam ist besser als schnell
  - zeitlassen zum **verstehen**
  - über das Gelesene **nachdenken**
  - imaginäre Fragen **beantworten**

Bild: toptimes

- Notizen machen
  - Aufschreiben fördert das Lernen
  - körperliche Aktivität beim Lernen kann den Lernerfolg erhöhen

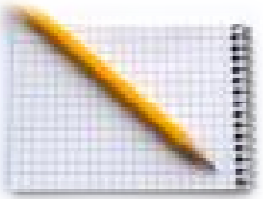


Bild: gk-photogalery

Besser lernen

Bild: thisnext

- Buch unter dem Kopfkissen
  - Übertragung ins Langzeitgedächtnis findet nach dem aktiven Lernen statt
  - Gehirn verarbeitet beim Schlafen Informationen
  - Keine anspruchsvollen Sachen nach dem Lernen vor dem Schlafengehen
- Trinken!
  - Austrocknung beeinträchtigt kognitive Leitungsfähigkeit
  - Austrocknung kommt vor dem Durst

Bild: lifedynamix

Besser lernen

- Lesen ist Silber, Reden ist Gold
  - Sprechen aktiviert einen anderen Teil des Gehirns
  - Erklärt es einer anderen Person

Bild: inventorspot

- Wegen Überfüllung geschlossen
  - Pause machen ist wichtig
  - Lesen ohne zu verstehen ist schlecht

Bild: keinfastfood

Besser lernen

- Darauf einlassen
  - darauf einlassen heißt Emotionen zu empfinden
  - besser über einen verunglückten Scherz stöhnen als nichts zu empfinden

- Selbst ist der Mann / die Frau
  - Benutze das gelernte
  - Tu irgendetwas damit. Hauptsache, du tust etwas

Bild: gnurf