Wissenschaftliche Softwareentwicklung

15. Entwurfsmuster (*Design Patterns*)

Till Biskup
Physikalisch-Technische Bundesanstalt
04.12.2023

Zentrale Aspekte

- Der Entwurf objektorientierter Software ist schwierig,
  und der wiederverwendbarer Software noch schwieriger.
- Muster sind bewährte (abstrakte) Lösungen
  für wiederkehrende Fragestellungen.
- Entwurfsmuster beschreiben interagierende Objekte und Klassen
  als maßgeschneiderte Lösung eines generellen Problems.
- Klar benannte Entwurfsmuster ermöglichen die Entwicklung
  von Software auf einer höheren Abstraktionsebene.
- Entwurfsmuster sind kein Selbstzweck.
  Sie können zu sparsam und zu freigiebig eingesetzt werden.

Übersicht

Zur Bedeutung von Mustern

Die „klassischen“ Entwurfsmuster in der Softwareentwicklung

Warum sollte man Entwurfsmuster kennen und einsetzen?

Der richtige Umgang mit Entwurfsmustern

Ausgangspunkt: konkrete Fragestellungen

Spannungsverhältnis zwischen Einmaligkeit und Allgemeingültigkeit

- Software dient der Bearbeitung konkreter Fragestellungen.
  - Programmierung ist selten ein Selbstzweck.
  - Fragestellungen definieren Qualitätsansprüche.
  - Wissenschaftlichkeit ist ein sehr hoher Qualitätsanspruch.
- Anforderungen ändern sich ständig.
  - Das liegt an der Komplexität nichttrivialer Fragestellungen.
  - Flexibilität ist ein zentraler Aspekt größerer Programme.
- Fragestellungen ähneln sich häufig in ihrer Struktur.
  - Die konkrete Fragestellung ist häufig einmalig.
  - Aber: Oft lassen sich allgemeine Muster erkennen.

☛ Spannungsverhältnis zwischen konkreter, einmaliger Aufgabe und Ähnlichkeit zu bestehenden Fragestellungen

☛ Wissenschaft trainiert und erfordert die Mustererkennung, Forschende sollten hier eigentlich im Vorteil sein.

Ziel: Wiederverwendbare und flexible Software

Objektorientierte Programmierung ist nur ein erster Schritt.

> *Designing object-oriented software is hard, and designing reusable object-oriented software is even harder. [...] Your design should be specific to the problem at hand but also general enough to address future problems and requirements.*
>
> – Gamma *et al.*

- warum objektorientiert programmieren?
  - ermöglicht eine Abstraktionsebene, die sehr wertvoll für die Handhabung komplexer Systeme ist
- Herausforderung objektorientierter Programmierung
  - erfordert sehr viel mehr Wissen und Erfahrung als strukturierte Programmierung *kleiner* Projekte
  - liegt an der intrinsischen Komplexität der Projekte/Fragestellungen

Gamma, Helm, Johnson, Vlissides: Design Patterns. Addison-Wesley, Boston 1995, S. 1

Lösung: Verwendung bewährter Strategien

Muster sind bewährte Lösungen zu wiederkehrenden Problemen.

Pattern (Muster)

Every pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.

- Muster sind weit verbreitet.
  - Die Inspiration für Entwurfsmuster in der Softwareentwicklung kommt aus der Architektur (Christopher Alexander *et al.*).
- Muster sind ein genereller Lösungsansatz.
  - Die konkrete Umsetzung sieht jedesmal anders aus.

Christopher Alexander *et al.*: A Pattern Language. Oxford University Press, New York 1977, S. x

Übersicht

Definition: Entwurfsmuster

Problemlösungen durch Interaktionen zwischen Objekten und Klassen

Entwurfsmuster (*design pattern*)

Beschreibungen miteinander kommunizierender Objekte und Klassen, die maßgeschneidert sind, um ein generelles Entwurfsproblem in einem bestimmten Kontext zu lösen

- Entwurfsmuster sind zweierlei:
  - Beschreibung einer (wiederkehrenden) Problemstellung
  - Strategie für die Lösung des Problems
- Abstraktionsebene von Entwurfsmustern
  - oberhalb von Klassen und Objekten
  - unterhalb von der Gesamtarchitektur eines Systems

Gamma, Helm, Johnson, Vlissides: Design Patterns. Addison-Wesley, Boston 1995, S. 3

Abgrenzung zu Idiomen und Architektur

Abstrakter als Idiome, konkreter als die (globale) Architektur

- Abstraktionsebene
  - höher als Idiome, niedriger als Architektur
  - oberhalb von Klassen und Objekten
- Bindung an eine Sprache
  - Idiome sind sprachspezifisch.
  - Entwurfsmuster und Architektur sind sprachunabhängig.
- Auswirkungen
  - Architektur betrifft das ganze System.
  - Entwurfsmuster lösen konkrete Problemstellungen.
- Bezug zur Realität außerhalb der Software
  - Architektur orientiert sich an der realen Problemstellung.
  - Entwurfsmuster lösen Probleme der Umsetzung.

Essentielle Elemente

Die Beschreibung von Entwurfsmustern folgt einem festen Schema.

- Name
  - kurz und prägnant: erweitert das Entwurfsvokabular
  - ermöglicht Entwerfen auf einer höheren Abstraktionsebene
- Problemstellung
  - Beschreibung, wann ein Entwurfsmuster angewendet wird
  - Problemstellung und Kontext
- Lösung
  - Bestandteile eines Entwurfs
  - Interaktion der Bestandteile für die Lösung
- Konsequenzen
  - Verständnis der Kosten und Vorteile
  - Einfluss auf Flexibilität, Erweiterbarkeit, Portierbarkeit

Zugrundeliegende Prinzipien

Entwurfsmuster folgen grundlegenden Prinzipien der OOP.

Zielstellung

- lose Kopplung zwischen einzelnen Elementen
- einfache Wiederverwendbarkeit der Einzelteile

Strategien

- Entwicklung gegen Schnittstellen statt Implementierungen
  - Kapselung („*black box*“) sorgt für Flexibilität.
  - Schnittstellen sollten die Implementierung verbergen.
  - Schnittstellen sollten immer so minimal wie möglich sein.
- Zusammensetzung gegenüber Vererbung bevorzugen
  - Vererbung neigt dazu, Kapselung aufzuweichen.
  - Hierarchien werden schnell unübersichtlich und unflexibel.

Gamma, Helm, Johnson, Vlissides: Design Patterns. Addison-Wesley, Boston 1995, S. 18 und 20

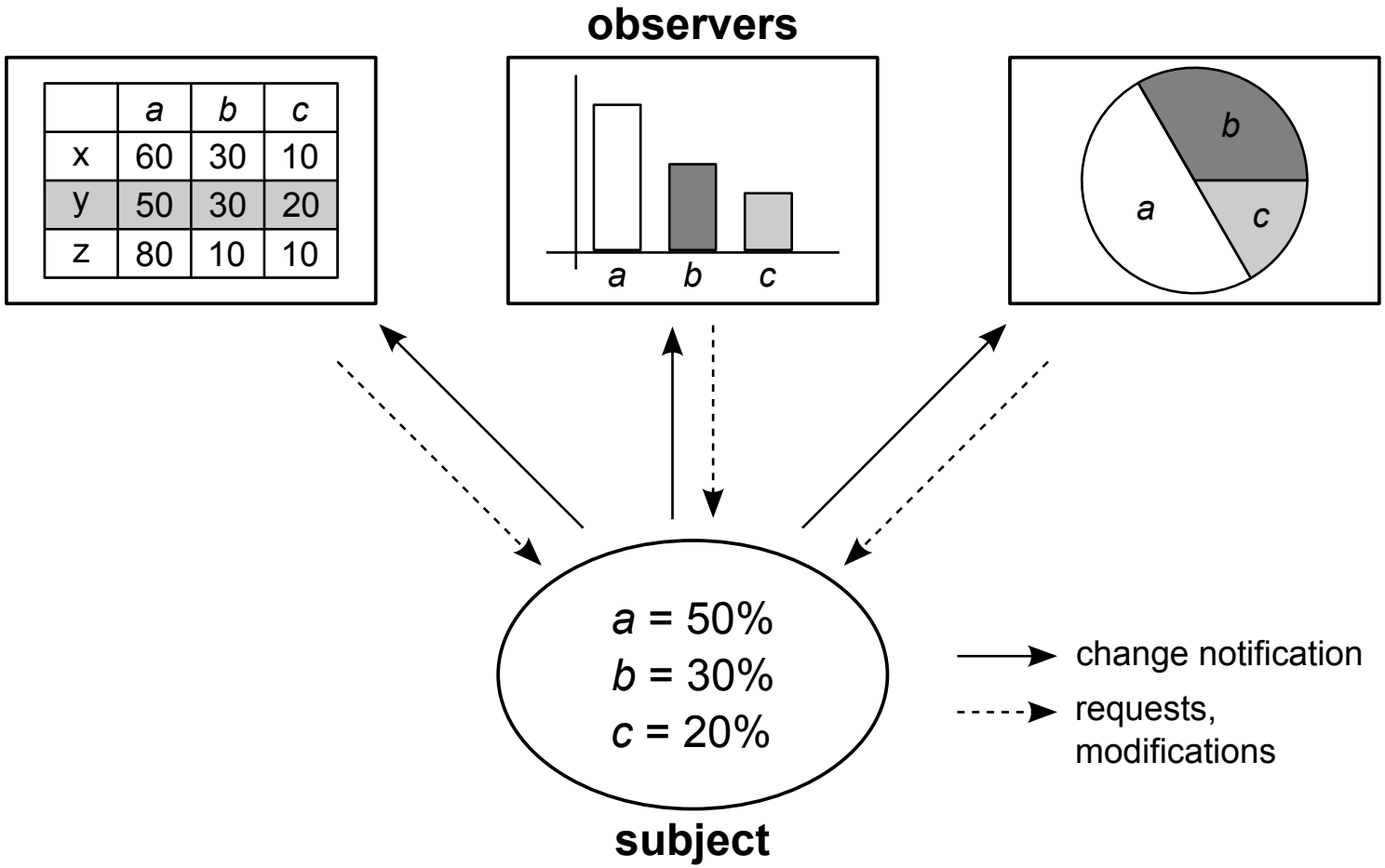
Konsequenzen

Entwurfsmuster haben großen Einfluss auf die Struktur des Codes.

- mehr Klassen und Objekte
  - Entwurfsmuster zielen auf kleinere, kompaktere Klassen.
  - Verhalten wird in eigene Klassen/Objekte ausgelagert.
  - erhöht richtig eingesetzt die Wiederverwendbarkeit
- flachere Hierarchien
  - Zusammensetzung wird gegenüber Vererbung bevorzugt.
  - entkoppelt durch Erhalt der Kapselung
  - führt zu kleineren, aber zahlenmäßig mehr Klassen
- auf den ersten Blick komplexerer Code
  - Entwurfsmuster sind relativ komplex.
  - Unkenntnis ist kein Argument gegen ihre Verwendung.
  - Vertrautheit sorgt für Wiedererkennungseffekte.

Beispiel: Das „Observer“-Entwurfsmuster

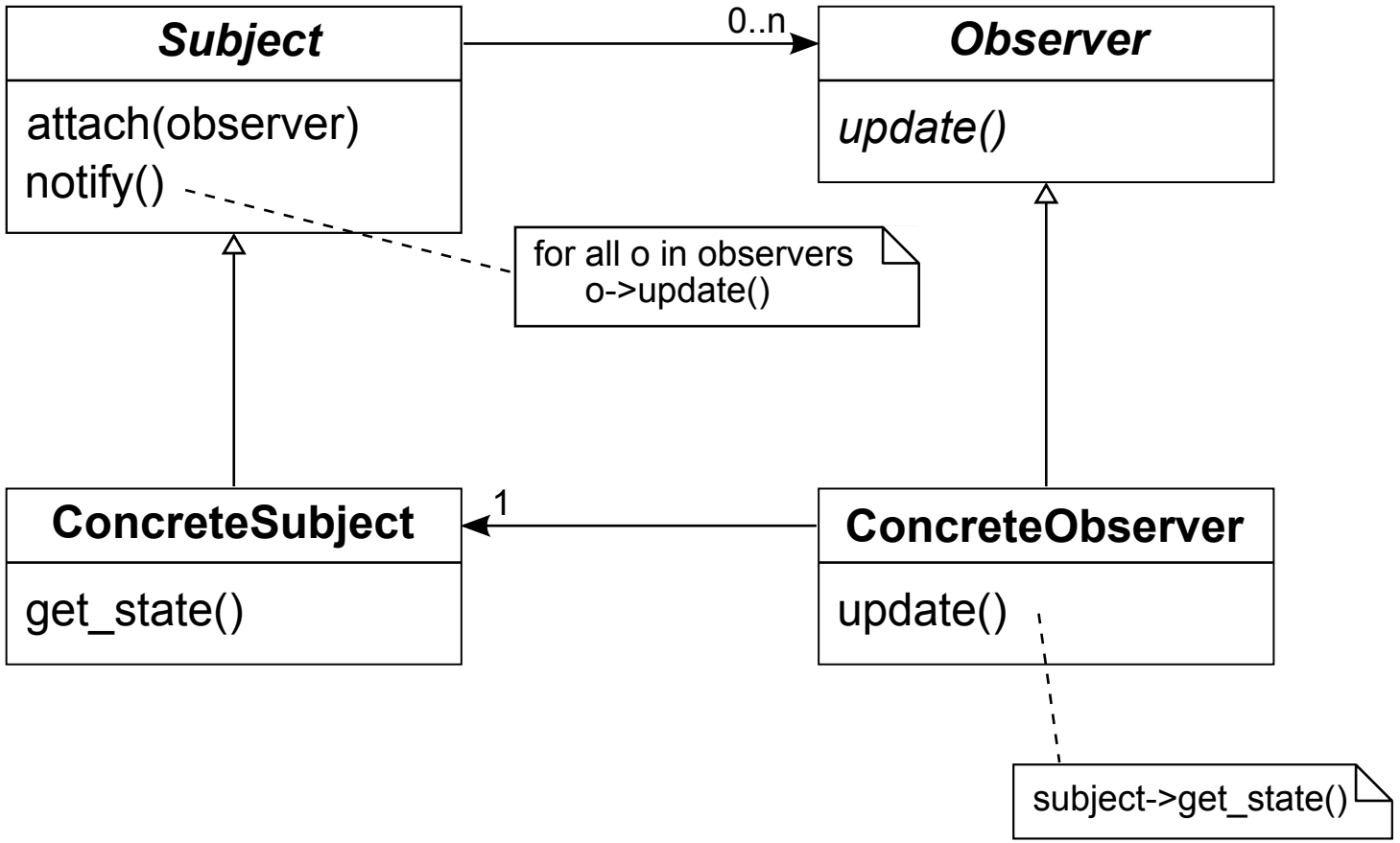
Motivation



verändert nach: Gamma, Helm, Johnson, Vlissides: Design Patterns. Addison-Wesley, Boston 1995, S. 293

Beispiel: Das „Observer“-Entwurfsmuster

(Vereinfachtes) UML-Diagramm



verändert nach: Gamma, Helm, Johnson, Vlissides: Design Patterns. Addison-Wesley, Boston 1995, S. 294

Beispiel: Das „Observer“-Entwurfsmuster

Implementation der abstrakten Klassen in Python

Listing: Abstrakte Klassen des Observer-Entwurfsmusters

```
class Subject():
    def __init__(self):
        self.observers = []

    def attach(self, observer=None):
        self.observers.append(observer)

    def notify(self):
        for observer in self.observers:
            observer.update()

class Observer():
    def __init__(self):
        self.observed_subject = None

    def update(self):
        self.observed_subject.get_state()
```

☛ Nur eine mögliche Implementierung, eng am UML-Diagramm

Beispiel: Das „Observer“-Entwurfsmuster

Implementation der konkreten Klassen in Python und Anwendung

Listing: Konkrete Klassen des Entwurfsmusters und deren Einsatz

```
class ConcreteSubject(Subject):
    def __init__(self):
        super.__init__()
        self.state = None

    def get_state(self):
        return self.state

class ConcreteObserver(Observer):
    def update(self)
        super.update()
        # Update own internals

data_source = ConcreteSubject()
histogram_view = ConcreteObserver()
data_source.attach(observer=histogram_view)
```

- `get_state` auch in der abstrakten Klasse definierbar
- Praxis: mehrere Beobachter für ein Subjekt

Übersicht

Ein gemeinsames Vokabular

Sprache beeinflusst unsere Art zu denken.

> *Patterns give a name and form to abstract heuristics, rules, and best practices of object-oriented techniques. No reasonable engineer wants to start from a blank slate . . .*
>
> – Philippe Kruchten

- gemeinsame Sprache als Grundlage der Kommunikation
  - Worte abstrahieren Konzepte in einer kompakten Weise.
  - Sprache beeinflusst, was und wie wir denken.
  - Verwendete Muster sollten klar benannt werden.
- das Rad nicht neu erfinden
  - Arbeitsteilung steigert wesentlich die Produktivität.
  - Uns fehlt meist das Wissen für eigene Entwicklungen.

Vorwort in: Craig Larman: Applying UML and Patterns. Prentice Hall, Upper Saddle River, 2005

Praktische Erfahrung von Experten

Auf diese Erfahrung aufbauen zu können ist extrem hilfreich.

> *If you don't know patterns,*
> *you're less likely to evolve great designs.*
> *Patterns capture wisdom.*
> *Reusing that wisdom is extremely useful.*
>
> – Joshua Kerievsky

- Entwurfsmuster transportieren praktische Erfahrung.
  - Jedes Muster im Katalog hat sich in der Praxis bewährt.
  - Es lohnt sich, Experten zuzuhören und zuzuschauen.
- Die Katalogisierung ermöglicht systematischen Zugang.
  - Muster sind abstrakte Lösungsvorschläge.
  - Vertrautheit mit den Mustern und genaue Analyse sind essentiell für den gewinnbringenden Einsatz.

Joshua Kerievsky: Refactoring to Patterns. Addison-Wesley, Boston 2005, S. 32

Vorteile beim Verwenden von Entwurfsmustern

Zusammenfassung und Überblick

- gemeinsames Vokabular
  - einfache Verständigung über relativ abstrakte Konzepte
  - Ein Muster transzendiert seine konkrete Umsetzung.
- Hilfe beim Verständnis bestehender Systeme
  - Objektorientiert programmierte Systeme sind oft komplex.
  - Die Komplexität spiegelt die Realität wider.
  - Kenntnis der verwendeten Muster hilft beim Verständnis.
- Brücke zwischen Analyse und Implementierung
  - Entwurfsmuster vermitteln zwischen der komplexen realen Fragestellung und der Umsetzung der Lösung in Software.

☛ Beherrschung von Entwurfsmustern hilft wesentlich bei der Entwicklung wiederverwendbarer hochwertiger Software.

Übersicht

Einsatz: im Entwurf oder in der Entwicklung?

Vor- und Nachteile des frühen Einsatzes von Entwurfsmustern

- zwei grundsätzliche Entwicklungsstrategien
  - kompletter detaillierter Entwurf vor der Implementierung „*Big Design Up-Front*“ (BDUF)
  - iterative Entwicklung von unten nach oben
- Kenntnis von Entwurfsmustern im Entwurfsprozess
  - manche Muster nur mühsam im Nachhinein umsetzbar
  - Kenntnis der Muster und genaue Anforderungsanalyse
  - BDUF bei überschaubaren Projekten und genug Erfahrung
- bestehenden Code zu Mustern weiterentwickeln
  - Regelfall für die Anwendung von Mustern
  - Code ist nicht statisch, Muster lassen sich später einführen.
  - „*Refactoring to Patterns*“

Joshua Kerievsky: Refactoring to Patterns. Addison-Wesley, Boston 2005

„You ain't gonna need it“ (YAGNI)

Prognosen sind schwierig, besonders wenn sie die Zukunft betreffen.

- zukünftige Anforderungen schwer vorhersehbar
  - Anforderungen an Software ändern sich ständig.
  - Die Richtung der Änderung ist (fast) nicht vorhersehbar.
  - intrinsisches Problem: Fragestellungen meist zu komplex
- immer nur die aktuellen Anforderungen erfüllen
  - Jeder Codeteil sollte real verwendet werden.
  - Flexibilität, Modularität und Wiederverwendbarkeit sind davon ungeachtet wichtige Ziele.
- „Vorauseilender Gehorsam“ bläht den Code auf.
  - Das Ziel ist einfacher und lesbarer Code.
  - Flexibilität erfordert selten komplexen Code, aber das Wissen um flexible und erweiterbare Lösungen.

Entwurfsmuster sind kein Selbstzweck

Das „Hammer-Nagel-Problem“

- generelle Zielstellung für Code
  - so einfach und lesbar wie möglich
  - so flexibel wie nötig
- Entwurfsmuster zielen auf Flexibilität und Erweiterbarkeit.
  - Beschreibungen enthalten konkrete Anwendungsszenarien.
  - Umsetzungen können unterschiedlich komplex ausfallen.
- Problem gerade von „Novizen“:
  - Entwurfsmuster werden um ihrer selbst willen eingesetzt.
  - Alles wird in die Form von Mustern „gepresst“.

☛ Entwurfsmuster sind nicht immer die beste Lösung.

☛ Der Einsatz von Entwurfsmustern braucht Erfahrung.

Wie eignet man sich Entwurfsmuster an?

Literatur, guter Code und praktische Erfahrung

- Entwurfsmuster wurden/werden katalogisiert.
  - „Klassiker“: Gamma *et al.*, Design Patterns (1995)
  - viele weitere (gute) Bücher
  - unterschiedliche Schwerpunkte und Anwendungsfälle
- guter Code professioneller Entwickler
  - Es gibt viele quelloffene Projekte mit guter Codequalität.
  - Lesen von Code anderer Entwickler erweitert den Horizont.
  - Code sollte immer kritisch hinterfragt werden.
- praktische Erfahrung
  - Programmierung lernt man nur über die Praxis.
  - Muster sollten systematisch ausprobiert werden.
  - Austausch mit anderen Programmierern

Zentrale Aspekte

- Der Entwurf objektorientierter Software ist schwierig,
  und der wiederverwendbarer Software noch schwieriger.
- Muster sind bewährte (abstrakte) Lösungen
  für wiederkehrende Fragestellungen.
- Entwurfsmuster beschreiben interagierende Objekte und Klassen
  als maßgeschneiderte Lösung eines generellen Problems.
- Klar benannte Entwurfsmuster ermöglichen die Entwicklung
  von Software auf einer höheren Abstraktionsebene.
- Entwurfsmuster sind kein Selbstzweck.
  Sie können zu sparsam und zu freigiebig eingesetzt werden.