Qualität von Programmen

- Brauchbarkeit:
  - Zweckerfüllung
  - Bedienbarkeit
  - Effizienz des Programmablaufs
- Zuverlässigkeit
- Wartbarkeit:
  - Einfachheit
  - Lesbarkeit
  - Lokalität
  - Faktorisierung

Programmerstellung und Wartung

- Schritte in Softwareentwicklungsprozessen:
  - Analyse
  - Entwurf
  - Implementierung
  - Verifikation und Validierung
- Arten von Softwareentwicklungsprozessen:
  - Wasserfallmodell
  - zyklische Prozesse mit schrittweiser Verfeinerung
- Qualitätsunterschiede zwischen Prozessen:
  - von vielen Faktoren abhängig
  - Anpassungen wichtig
  - Überprüfung der Prozesse auf Eignung

Qualität von Programmiersprachen

- Zuverlässigkeit der Programme:
  - Schreibbarkeit
  - Lesbarkeit
  - Einfachheit
  - Sicherheit
  - Robustheit
- Wartbarkeit der Programme
- Effizienz
  - der Werkzeuge und der generierten Programme
  - der Programmerstellung und Wartung

Rezept für gute Programme

- Softwareentwicklung ist sehr komplex (unvollständiges Wissen, widersprüchliche Ziele, Zeitdruck)

  $\Rightarrow$ einfache Rezepte scheitern

- umfangreicher Erfahrungsschatz

- gezielter Einsatz von Erfahrung erhöht Erfolgsaussichten

- objektorientierte Programmierung bietet mehr Faktorisierungsmöglichkeiten als andere Paradigmen

  $\Rightarrow$ erleichtert gezielten Einsatz von Erfahrung

  $\Rightarrow$ überfordert Anfänger

Verantwortlichkeiten einer Klasse

- definiert durch drei w-Ausdrücke (Ich ist Instanz):
  - was ich weiß (Zustand der Instanzen)
  - was ich mache (Verhalten der Instanzen)
  - wen ich kenne (sichtbare Objekte, Klassen)
- EntwicklerInnen der Klasse zuständig für Änderungen in den Verantwortlichkeiten der Klasse

Klassen-Zusammenhalt

- Klassen-Zusammenhalt (class coherence) = Grad der Beziehungen zwischen den Verantwortlichkeiten der Klasse
- hoch, wenn
  - Variablen und Methoden eng zusammenarbeiten
  - und durch Klassenname gut beschrieben
- Klassen-Zusammenhalt soll hoch sein

  $\Rightarrow$ Hinweis auf gute Faktorisierung
  (wenig Änderungen nötig)

Objekt-Kopplung

- Objekt-Kopplung (object coupling) = Abhängigkeit der Objekte von einander
- stark, wenn
  - viele sichtbare Methoden und Variablen
  - viele Nachrichten im laufenden System
  - viele Parameter in Methoden
- Objekt-Kopplung soll schwach sein

  $\Rightarrow$ Hinweis auf gute Kapselung
  (wenig unnötige Beeinflussung bei Änderungen)

Klassenzusammenhalt, Objektkopplung

- hängen eng zusammen
- bereits in früher Entwicklungsphase abschätzbar
- hilfreich bei der Bewertung von Alternativen

Refaktorisierung

- erste Faktorisierung oft nicht optimal
- Refaktorisierung = Änderung der Programmstruktur (ohne Änderung der Funktionalität)
- Refaktorisierung in Anfangsphase oft billig
- wenige gezielte Refaktorisierungen führen zu stabiler Zerlegung in Objekte (braucht nicht mehr geändert werden)
- refaktorisieren bevor Probleme sich über das ganze Programm ausbreiten

Wiederverwendung

- Programme
- Daten
- Erfahrungen
- Code
  - globale Bibliotheken
  - fachspezifische Bibliotheken
  - projektinterne Wiederverwendung
  - programminterne Wiederverwendung

Voraussetzungen

- Kosteneinsparungen durch Codewiederverwendung nur erzielbar wenn
  - EntwicklerInnen ausreichend erfahren
  - Zeit in Wiederverwendbarkeit investiert
- Fehlentscheidungen $\Rightarrow$ lange Entwicklungszeiten, Scheitern
- im Zweifelsfall weniger in Wiederverwendbarkeit investieren, bei Bedarf Refaktorisierungen nachholen
- nötige Refaktorisierungen nicht verzögern

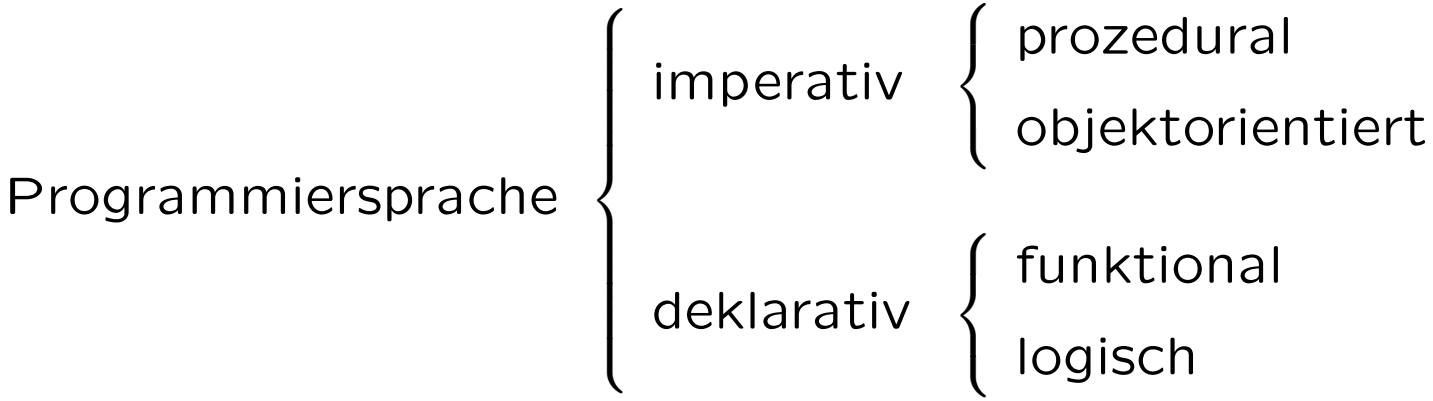
Entwurfsmuster

- dienen dem Austausch kollektiver Erfahrung
- man wählt beste Lösung aus Katalog
- ein Entwurfsmuster besteht aus
  - Name
  - Problemstellung
  - Lösung
  - Konsequenzen

Einsatz von Entwurfsmustern

- Entwurfsmuster so kombinierbar, dass viele erwünschte Eigenschaften erzielbar, negative Eigenschaften minimierbar
- aber übermäßiger Einsatz führt zu unnötig komplizierten Programmen – schwer wartbar
- auch Einsatz von Entwurfsmustern setzt Erfahrung voraus

Paradigmen der Programmierung



Eignung der Programmierparadigmen

- wenn Komplexität des Systems von Komplexität der Algorithmen dominiert

  $\Rightarrow$ prozedurale oder deklarative Programmierung

- wenn Komplexität des Systems deutlich höher als Komplexität der einzelnen Algorithmen (großes System)

  $\Rightarrow$ objektorientierte Programmierung

Paradigmen für Modularisierung

- Programmierung mit Modulen
- Programmierung mit abstrakten Datentypen
- objektorientierte Programmierung
- generische Programmierung