Erinnerung: UML-Aufgaben im Praktomaten

- einführende Aufgaben zur Java-Programmierung
- Aufgaben zum Übungsmaterial
- zusätzliche, komplexere Aufgaben
- Klausurrelevante Aufgaben (Implementierungsteil)

Teil IV: Objektorientierter Entwurf (OOD)

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
Version 13-1.0, 06.07.13

Teil IV - Objektorientierter Entwurf (Object-Oriented Design, OOD)

Vorbemerkungen über Verfeinerung

1) Einführung in die objektorientierte Softwarearchitektur
   1) Modularität und Geheimnisprinzip
   2) Entwurfsmuster für Modularität
   3) BCD-Architekturstil (3-tier architectures)
2) Verfeinerung des Entwurfsmodells zum Implementierungsmodell
   1) Verfeinerung von Klassendiagrammen
   2) Verfeinerung von Lebenszyklen
   3) Verfeinerung mit Object Fattening
3) Objektorientierte Rahmenwerke (frameworks)
4) Softwarearchitektur mit dem Quasar-Architekturstil

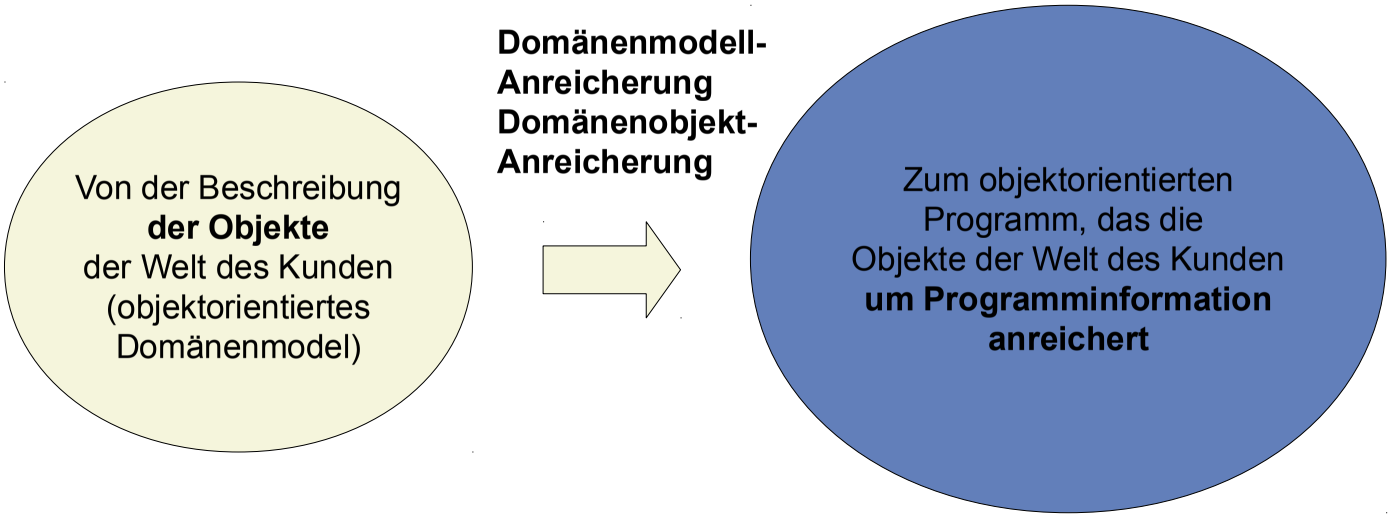
The Engineer's Ring (Canada)

- The ring of engineering of Canadian engineers
  - The broken bridge of Pont de Québec
  - The engineer's oath
- Software Engineering is an “science of engineering”
- Engineers solve problems based on laws of design, construction, production, prediction

> "Gold is for the mistress - silver for the maid!
> Copper for the craftsman cunning at his trade."
> "Good!" said the Baron, sitting in his hall.
> "But Iron, Cold Iron - is master of them all!"
>
> Rudyard Kipling

Die zentralen Fragen des objektorientierten Ansatzes

Wie kommen wir vom Problem des Kunden zum Programm (oder Produkt)?



Anreicherung/Verfettung: Anreicherung durch technische Programminformation
„object fattening“: Anreicherung von Objekten des Domänenmodells

Verfeinerung von UML von der Analyse zum Entwurf zur Implementierung

	Analyse-Modell	Entwurfs-Modell (Architektur)	Implementierungsmodell (Feinentwurf)	Implementierung (Programm)
vollständig	nein	nein	beinahe	ja
Sprache	aUML	dUML	jUML	Java
Charakter	Fachlichkeit; Domäne	Fachlichkeit und System	Fachlichkeit, System und Technologie	Fachlichkeit, System, Technologie und Resourcen
Technologie	Annahme perfekter Technologie; funktionale Essenz	perfekt; funktionale Essenz und interne Aktivitätäten	technologieabhängig (plattformabhängig)	technolgieabhängig (plattformabhängigg), resourcenabhängig
Struktur	noch wenig	Architektur des Systems	Verhalten des Systems	Lauffähiges Programm

Verfeinerung der Sprachkonzepte

	Analyse-Modell	Entwurfs-Modell (Architektur)	Implementierungsmodell (Feinentwurf)	Implementierung
Klassen	Begriffe und Domänenkonzepte; Facetten, Rollen	Systemkonzepte; Facetten, Rollen; Komponenten	Systemkonzepte nur noch in einfachen Klassen; Rollen aufgelöst durch Entwurfsmuster	Systemkonzepte
Objekte	Domänenobjekte	Systemobjekte auf Architekturebene	Alle Systemobjekte	Alle Systemobjekte
Vererbung	Begriffshierarchien Mehrfachvererbung, Produktverbände	ad-hoc Vererbung	konform, 1-D-Vererbungshierarchie	konform
Assoziationen	oft ungerichtet	mit Kardinalitäten	abgeflacht	

- Verfeinerung vom Entwurfsmodell zur Implementierungsmodell und Implementierung
  - Umorganisieren von nicht-konformer in konforme Vererbung
  - Transformation in die Implementierungssprache

Verfeinerungsschritte beim Übergang vom Analyse- zum Entwurfsmodell

- **Vervollständigung** fehlender Angaben
  - Typisierung
  - Ausarbeitung von Objekt-Lebenszyklen (punktweise Verfeinerung)
- **Detaillierung**
  - Querschneidende Objektanreicherung durch Kollaborationen
  - Assoziationen:
    - Einziehen von Navigationsrichtungen, um Platz zu sparen
    - Einziehen von Qualifikationen, um Suchen zu beschleunigen
    - Annotation von geordneten und sortierten Assoziationen
  - Einziehen von Schnittstellen zur Sicherstellung von homogenem Verhalten
  - Einziehen von Materialbehälterklassen (Verwaltungsklassen)

Verfeinerungsschritte beim Übergang zum Entwurfsmodell (II)

- **Strukturierung** und Restrukturierung
  - Refactoring (Restrukturierung unter Beibehaltung der Semantik)
  - Erhöhung von Wiederverwendung:
    - Einziehen von Komponenten mit angebotenen und benötigten Schnittstellen
    - Einziehen von Paketen zur Strukturierung
    - Einschränkung durch Sichtbarkeiten zur loseren Kopplung (Datenkapselung und Austauschbarkeit)
  - Identifikation von abgeleiteten Modellelementen zur Elimination von Redundanz (Verschlankung)
- **Flachklopfen** (Abflachen, Realisierung)
  - Integration von Unterobjekten (Rollen, Facetten, Teile) mit Hilfe von Entwurfsmustern: Compositum, Brücke, u.v.m.
  - Abflachen der Assoziationen
- **Erhöhung Zuverlässigkeit**
  - Verfeinerung der Vererbungsrelation zu konformer Vererbung zur Elimination von Fehlern
  - Schreiben von Verträgen mit Vor- und Nachbedingungen

Verfeinerungsschritte beim Übergang zum Entwurfsmodell (III)

- Entwickeln des **Architekturaspektes**
  - Entwurfsmuster anwenden
  - Schichtung
  - Architekturprinzipien wie Geheimnisprinzip oder lose Kopplung
- Erhöhe **Wiederverwendbarkeit**
  - Verwende **Variabilitätsmuster**
  - Verwende **Erweiterungsmuster**
  - Integriere externe Komponenten durch **Klebemuster**
  - Ausnutzen des **Quasar**-Prinzips zur Identifikation von wiederverwendbaren Einheiten

Schritte beim Übergang zum Implementierungsmodell (Feinentwurf)

- Vervollständigung fehlender Angaben
  - Implementierung von Methoden
- Strukturierung und Restrukturierung
  - Auflösen von Mehrfachvererbung
  - Einziehen von abgeleiteten Relationen, um Zugriffe, Navigationen und Abfragen zu beschleunigen (Verfettung)
- Detaillierung (Implementierungsmuster anwenden)
  - Assoziationen:
    - Einziehen von Navigationsrichtungen, um Platz zu sparen; Ersetzen durch Suchalgorithmen
    - Abbildung von qualifizierten Assoziationen, um Suchen zu beschleunigen
    - Annotation von geordneten und sortierten Assoziationen
  - Verfeinerung der Vererbungsrelation zu *konformer Vererbung* zur Elimination von Fehlern
- Verhalten angeben
  - Lebenszyklen modellieren
  - Implementierung von Schnittstellen auswählen

Schritte beim Übergang zum Code

- **Codegenerierung**
  - Nutzung von Codegenerator-Werkzeugen, um Lebenszyklen in Programm zu übersetzen
  - Nutzung von Webe-Werkzeugen, um querschneidende Kollaborationen in Klassen einzuweben
  - Umsetzung der Modelle in Code von Hand

The End

Erinnerung: UML-Aufgaben im Praktomaten

- einführende Aufgaben zur Java-Programmierung
- Aufgaben zum Übungsmaterial
- zusätzliche, komplexere Aufgaben
- Klausurrelevante Aufgaben (Implementierungsteil)

Teil IV: Objektorientierter Entwurf (OOD)

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
Version 13-1.0, 06.07.13

- Parallelen zum Fachgebiet der Architektur:
  - Architekten sind an der Nahtstelle zwischen Kunde und Baufirma.
  - Schlechter Architekturentwurf kann nicht durch gute Bauqualität kompensiert werden.
  - Es gibt Architektur-Spezialisten für bestimmte Anwendungsgebiete.
  - Es gibt "Schulen", die bestimmte Grundprinzipien vertreten.
  - Es gibt bestimmte

Teil IV - Objektorientierter Entwurf (Object-Oriented Design, OOD)

Vorbemerkungen über Verfeinerung

1) Einführung in die objektorientierte Softwarearchitektur
   1) Modularität und Geheimnisprinzip
   2) Entwurfsmuster für Modularität
   3) BCD-Architekturstil (3-tier architectures)
2) Verfeinerung des Entwurfsmodells zum Implementierungsmodell
   1) Verfeinerung von Klassendiagrammen
   2) Verfeinerung von Lebenszyklen
   3) Verfeinerung mit Object Fattening
3) Objektorientierte Rahmenwerke (frameworks)
4) Softwarearchitektur mit dem Quasar-Architekturstil

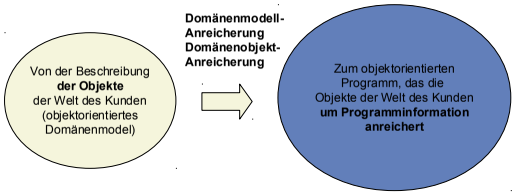
The Engineer's Ring (Canada)

- The ring of engineering of Canadian engineers
  - The broken bridge of Pont de Québec
  - The engineer's oath
- Software Engineering is an “science of engineering”
- Engineers solve problems based on laws of design, construction, production, prediction

> "Gold is for the mistress - silver for the maid!
> Copper for the craftsman cunning at his trade."
> "Good!" said the Baron, sitting in his hall.
> "But Iron, Cold Iron - is master of them all!"
>
> Rudyard Kipling

Die zentralen Fragen des objektorientierten Ansatzes

Wie kommen wir vom Problem des Kunden zum Programm (oder Produkt)?



Anreicherung/Verfettung: Anreicherung durch technische Programminformation
„object fattening": Anreicherung von Objekten des Domänenmodells

Verfeinerung von UML von der Analyse zum Entwurf zur Implementierung

	Analyse-Modell	Entwurfs-Modell (Architektur)	Implementierungsmodell (Feinentwurf)	Implementierung (Programm)
vollständig	nein	nein	beinahe	ja
Sprache	aUML	dUML	jUML	Java
Charakter	Fachlichkeit; Domäne	Fachlichkeit und System	Fachlichkeit, System und Technologie	Fachlichkeit, System, Technologie und Resourcen
Technologie	Annahme perfekter Technologie; funktionale Essenz	perfekt; funktionale Essenz und interne Aktivitätäten	technologieabhängig (plattformabhängig)	technolgieabhängig (plattformabhängigg), resourcenabhängig
Struktur	noch wenig	Architektur des Systems	Verhalten des Systems	Lauffähiges Programm

Verfeinerung der Sprachkonzepte

	Analyse-Modell	Entwurfs-Modell (Architektur)	Implementierungsmodell (Feinentwurf)	Implementierung
Klassen	Begriffe und Domänenkonzepte; Facetten, Rollen	Systemkonzepte; Facetten, Rollen; Komponenten	Systemkonzepte nur noch in einfachen Klassen; Rollen aufgelöst durch Entwurfsmuster	Systemkonzepte
Objekte	Domänenobjekte	Systemobjekte auf Architekturebene	Alle Systemobjekte	Alle Systemobjekte
Vererbung	Begriffshierarchien Mehrfachvererbung, Produktverbände	ad-hoc Vererbung	konform, 1-D-Vererbungshierarchie	konform
Assoziationen	oft ungerichtet	mit Kardinalitäten	abgeflacht	

- Verfeinerung vom Entwurfsmodell zur Implementierungsmodell und Implementierung
  - Umorganisieren von nicht-konformer in konforme Vererbung
  - Transformation in die Implementierungssprache

Verfeinerungsschritte beim Übergang vom Analyse- zum Entwurfsmodell

- **Vervollständigung** fehlender Angaben
  - Typisierung
  - Ausarbeitung von Objekt-Lebenszyklen (punktweise Verfeinerung)
- **Detaillierung**
  - Querschneidende Objektanreicherung durch Kollaborationen
  - Assoziationen:
    - Einziehen von Navigationsrichtungen, um Platz zu sparen
    - Einziehen von Qualifikationen, um Suchen zu beschleunigen
    - Annotation von geordneten und sortierten Assoziationen
  - Einziehen von Schnittstellen zur Sicherstellung von homogenem Verhalten
  - Einziehen von Materialbehälterklassen (Verwaltungsklassen)

Verfeinerungsschritte beim Übergang zum Entwurfsmodell (II)

- **Strukturierung** und Restrukturierung
  - Refactoring (Restrukturierung unter Beibehaltung der Semantik)
  - Erhöhung von Wiederverwendung:
    - Einziehen von Komponenten mit angebotenen und benötigten Schnittstellen
    - Einziehen von Paketen zur Strukturierung
    - Einschränkung durch Sichtbarkeiten zur loseren Kopplung (Datenkapselung und Austauschbarkeit)
  - Identifikation von abgeleiteten Modellelementen zur Elimination von Redundanz (Verschlankung)
- **Flachklopfen** (Abflachen, Realisierung)
  - Integration von Unterobjekten (Rollen, Facetten, Teile) mit Hilfe von Entwurfsmustern: Compositum, Brücke, u.v.m.
  - Abflachen der Assoziationen
- **Erhöhung Zuverlässigkeit**
  - Verfeinerung der Vererbungsrelation zu konformer Vererbung zur Elimination von Fehlern
  - Schreiben von Verträgen mit Vor- und Nachbedingungen

Verfeinerungsschritte beim Übergang zum Entwurfsmodell (III)

- Entwickeln des **Architekturaspektes**
  - Entwurfsmuster anwenden
  - Schichtung
  - Architekturprinzipien wie Geheimnisprinzip oder lose Kopplung
- Erhöhe **Wiederverwendbarkeit**
  - Verwende **Variabilitätsmuster**
  - Verwende **Erweiterungsmuster**
  - Integriere externe Komponenten durch **Klebemuster**
  - Ausnutzen des **Quasar**-Prinzips zur Identifikation von wiederverwendbaren Einheiten

Schritte beim Übergang zum Implementierungsmodell (Feinentwurf)

- Vervollständigung fehlender Angaben
  - Implementierung von Methoden
- Strukturierung und Restrukturierung
  - Auflösen von Mehrfachvererbung
  - Einziehen von abgeleiteten Relationen, um Zugriffe, Navigationen und Abfragen zu beschleunigen (Verfettung)
- Detaillierung (Implementierungsmuster anwenden)
  - Assoziationen:
    - Einziehen von Navigationsrichtungen, um Platz zu sparen; Ersetzen durch Suchalgorithmen
    - Abbildung von qualifizierten Assoziationen, um Suchen zu beschleunigen
    - Annotation von geordneten und sortierten Assoziationen
  - Verfeinerung der Vererbungsrelation zu *konformer Vererbung* zur Elimination von Fehlern
- Verhalten angeben
  - Lebenszyklen modellieren
  - Implementierung von Schnittstellen auswählen

Schritte beim Übergang zum Code

- **Codegenerierung**
  - Nutzung von Codegenerator-Werkzeugen, um Lebenszyklen in Programm zu übersetzen
  - Nutzung von Webe-Werkzeugen, um querschneidende Kollaborationen in Klassen einzuweben
  - Umsetzung der Modelle in Code von Hand

The End