

Bachelor of Science (BSc) in Informatik
Modul Software-Entwicklung 1 (SWEN1)

LE 14 – Wrap-up der Vorlesung

SWEN1/PM3 Team:
R. Ferri (feit), D. Liebhart (lieh), K. Bleisch (bles), G. Wyder (wydg)

Ausgabe: HS24

Um was geht es?

- Was sind die wichtigen Themen und Schwerpunkte der Vorlesung?
- Wieviel weiss ich noch davon?
- Was ist genau der Inhalt der Semesterendprüfung?
- Was muss ich allenfalls nochmals Repetieren für die Semesterendprüfung?



Lernziele LE 14 – Wrap up der Vorlesung

- Sie sind in der Lage:
 - die wichtigsten Themen in der Vorlesung im Kontext eines iterativen Softwareentwicklungsprozesses kurz in eigenen Worten zu erläutern (Zweck, Prozesse und Artefakte).
 - den Ablauf, Inhalt und die Prüfungsmodalitäten der Semesterendprüfung (SEP) zu erklären.
 - besser einschätzen zu können, was Sie für die SEP noch repetieren müssen.

Agenda

1. Überblick
2. Recap Anforderungsanalyse und Domänenmodellierung
3. Recap Softwarearchitektur und Design
4. Recap Entwurf mit Design Patterns
5. Info zur Semesterendprüfung
6. Bearbeitung einer Fallstudie (alte SEP)
7. Wrap-up

Lernziele SWEN1 (1/2)

- Sie sind in der Lage:
 - für einen vorgegebenen, **iterativ-inkrementellen Softwareentwicklungsprozess** den Ablauf und die **Artefakte** zur **Entwicklung einer objektorientierten Softwareapplikation** zu erläutern,
 - die **Begriffswelt des Anwenders** durch geeignete Vorgehensweisen erfassen und zu einer fachlichen Terminologie **zu verdichten (Domänenmodell)**,
 - eine **Softwareapplikation** sinnvoll abzugrenzen,
 - **systematisch die funktionalen Anforderungen** mit Use Cases sowie **Qualitätsanforderungen und Randbedingungen** zu erheben und zu kommunizieren,

Lernziele SWEN1 (2/2)

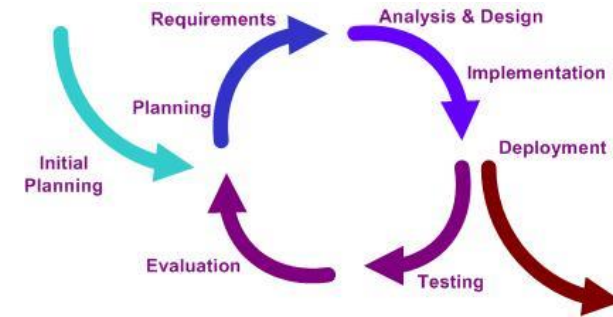
- basierend auf den Anforderungen eine **geeignete Softwarearchitektur** und ein **objektorientiertes Design** - Klassen mit Verantwortlichkeiten - für die darin enthaltenen Komponenten **der fachlichen Logik zu entwerfen**,
- für die **Modellierung und Kommunikation von Artefakten** im Softwareentwicklungsprozess **standardisierte Notationen** (wie UML) zu benutzen,
- bewährte **Analyse, Architektur und Design Patterns** adäquat für eine Problemstellung einzusetzen.

Themen und Ablauf des Moduls SWEN1

SW#	Thema
01	Einführung und Überblick
02	Anforderungsanalyse I
03	Anforderungsanalyse II
04	Domänenmodellierung
05	Quizzzy 1: Analyse Softwarearchitektur und Design I
06	Softwarearchitektur und Design II
07	UseCase Implementation
08	Entwurf mit Design Patterns I
09	Entwurf mit Design Patterns II
10	Refactoring und Testing
11	Quizzzy 2: Design Vertiefung 1: Verteilte Systeme
12	Vertiefung 2: Persistenz
13	Vertiefung 3: Framework Design
14	Wrap-up der Vorlesung

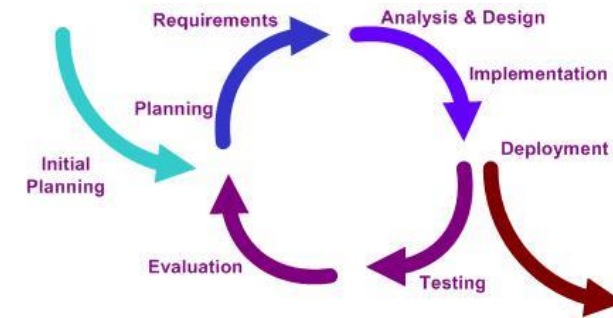
Angewendeter iterativ-inkrementeller Softwareentwicklungsprozess in SWEN1/PM3

- Der Softwareentwicklungsprozess wurde so angepasst (engl. tailoring), dass die **wesentlichen Artefakte** in einem **Softwareprojekt** im Kontext eingeführt werden können.
- Die Software wird in **Iterationen** entwickelt (2 Wochen Rhythmus).
- Jede Iteration hat ein Ziel und wird nach Abschluss reviewed.
- Es gibt **drei Meilensteine**, die im Projektverlauf ein besonderes Ereignis darstellen bzw. den Abschluss einer Phase: Projektskizze (M1), Lösungsarchitektur (M2) und Beta-Release (M3)
- In **jeder Iteration** werden **Anforderungen, Analyse & Design, Implementation und Testing** gemacht (Software entsteht in Inkrementen).
- Der angewendete Softwareentwicklungsprozess und das **Projektmanagement eines iterativ-inkrementellen Projektes** wird in PM3 noch detaillierter erklärt.



Wesentliche Resultate bzw. Artefakte

- **Anforderungsanalyse**
 - Funktionale Anforderungen mit Use Cases
 - Qualitätsanforderungen und Randbedingungen
 - Domänenmodell
- **Design**
 - Softwarearchitektur
 - Use Case Realisierung (statische und dynamische Modelle)
- **Implementation**
 - Quellcode (inkl. Javadoc)
- **Testing**
 - Unit-Tests
 - Integrations- und Systemtests

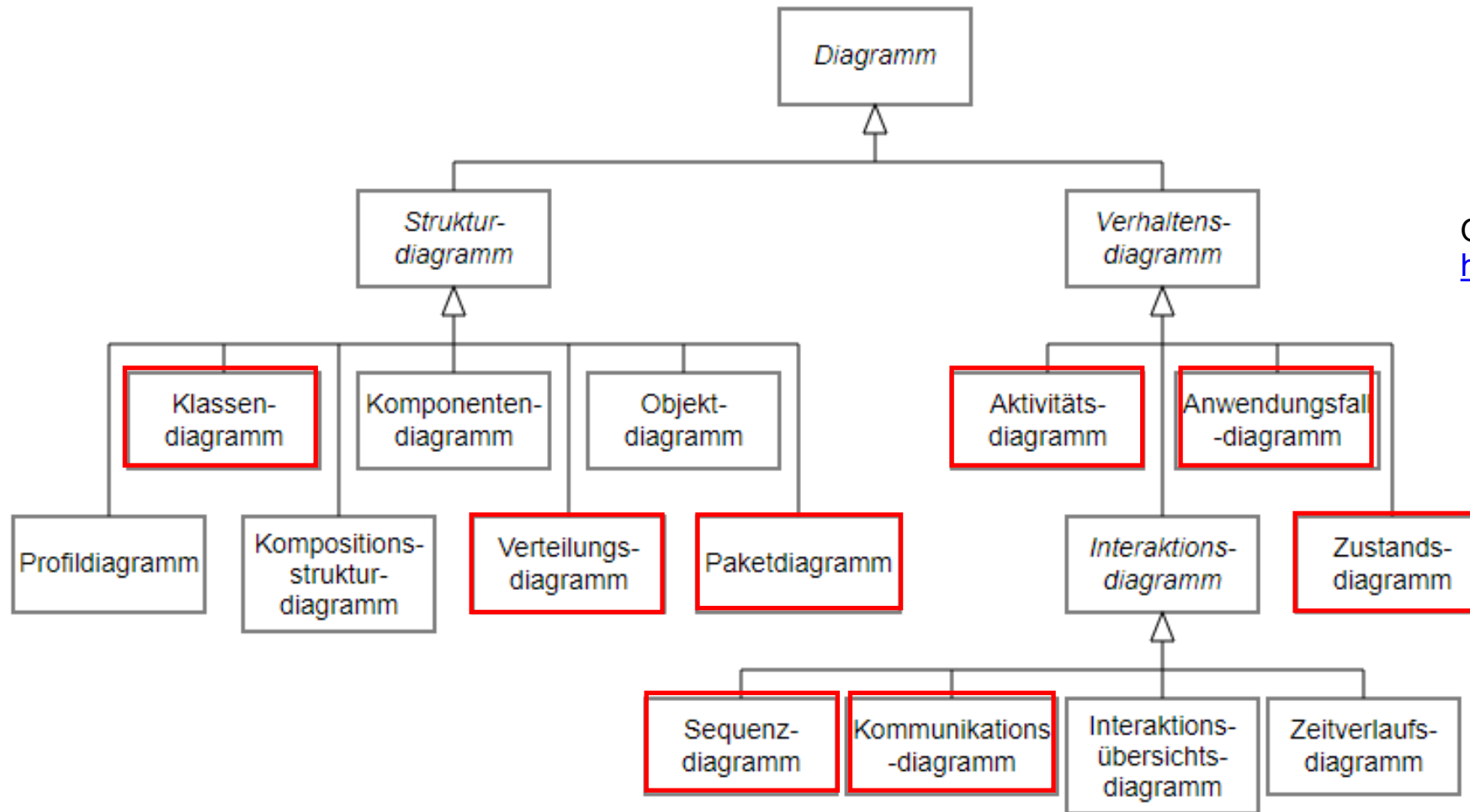


Wichtig:
Diese Artefakte entstehen inkrementell, d.h. werden
schrittweise in jeder Iteration verfeinert und ergänzt!

Modellierung und Modelle mit der UML



Quelle: UML Specification,
<https://www.omg.org/spec/UML/>



 für die Modellierung in SWEN1 relevant

Gebrauch der UML (nach Martin Fowler)

- **UML as a Sketch**

- Informelle und unvollständige Diagramme (z.T. von Hand gezeichnet), um schwierige Teile des Problems oder der Lösung zu verstehen und zu kommunizieren
- Die agile Community bevorzugt diese Anwendungsart von UML



- **UML as a Blueprint**

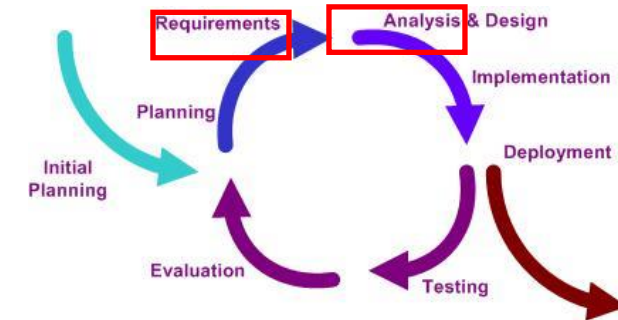
- Relativ detaillierte Analyse und Design-Diagramme für Code-Generierung oder um existierenden Code besser zu verstehen
- Klassische UML-Tools für ein Forward- und Reverse-Engineering (Round-trip)

- **UML as a Programming Language**

- Komplete, ausführbare Spezifikation eines Software-Systems in UML
- MDA-Tools zur Modellierung und Generierung

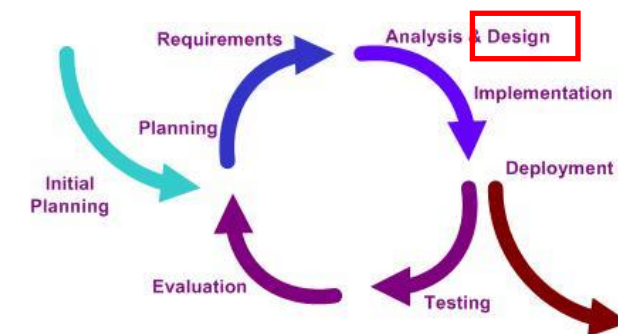
Überblick Anforderungen & Analyse

- **User Research** (Personas und Szenarien, Contextual Inquiry)
- Sketching und Prototyping
- Ableiten und Modellieren von **Use Cases** (dt. Anwendungsfälle)
- **Detaillierung der Use Case** (UML-Use-Case-Diagramm, Use-Case-Spezifikationen, UI-Sketching)
- **Qualitätsanforderungen** und **Randbedingungen** erheben und festhalten.
- Modellierung der Fachlichkeit und Begriffe des Anwenders in einem **Domänenmodell** (konzeptuelles UML-Klassendiagramm)
- Bei der **objektorientierten Analyse (OOA)** liegt die Betonung darauf, die Objekte – oder Konzepte in dem Problembereich zu finden und zu beschreiben!



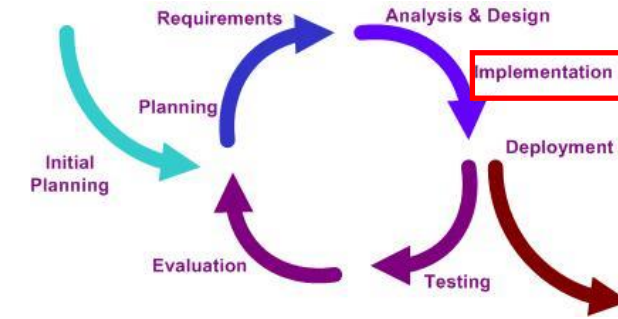
Überblick Design

- Design und Modellierung einer für die Problemstellung geeigneten **Softwarearchitektur** (UML-Paketdiagramm, UML-Verteilungsdiagramm)
- **Use-Case-Realisierung** und **Klassendesign** mit Verantwortlichkeiten (UML-Klassendiagramm, UML-Sequenzdiagramm, UML-Kommunikationsdiagramm, UML-Zustandsdiagramm, UML-Aktivitätsdiagramm)
- Entwurf mit bewährten **Design Patterns**
- Beim **objektorientierten Design (OOD)** liegt die Betonung darauf, geeignete Softwareobjekte und ihr Zusammenwirken (engl. collaboration) zu definieren, um die Anforderungen zu erfüllen!



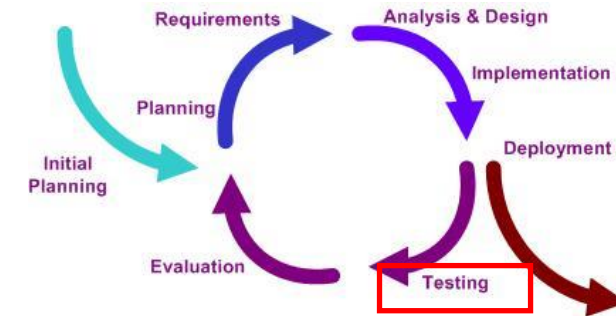
Überblick Implementation

- Umsetzung des Designs in **Code** der entsprechenden (objektorientierten) Programmiersprache
- Verwendung von geeigneten **Algorithmen und Datenstrukturen** zur Implementierung des Designs
- Code Smells sofort bei deren Aufdeckung verbessern (**Refactoring**)
- **Laufende Dokumentation** des **Quellcodes** (nach Clean Code-Prinzipien)



Überblick Testing

- Laufendes Design und Implementierung von **Unit-Tests**
- Planung, Design und Durchführung von weiteren Tests auf den **Teststufen Integration** und **System** je nach Problemstellung
- **Dokumentation** des Testkonzepts und der Tests



Agenda

1. Überblick
- 2. Recap Anforderungsanalyse und Domänenmodellierung**
3. Recap Softwarearchitektur und Design
4. Recap Entwurf mit Design Patterns
5. Info zur Semesterendprüfung
6. Bearbeitung einer Fallstudie (alte SEP)
7. Wrap-up

Aufgabe 14.1 (10')

Diskutieren Sie in Murmelgruppen folgende Fragen:

- Was ist Usability und wieso ist das wichtig für die Softwareentwicklung?
- Wie findet man die wichtigen Anforderungen an ein Softwareprodukt?
- Was sind Prozesse und Artefakte in der Anforderungsanalyse für die Softwareentwicklung?
- Was ist Domänenmodellierung und was ihr Zweck in der Anforderungsanalyse?
- Wieviel Anforderungsanalyse und Domänenmodellierung braucht es für ein Projekt?

Agenda

1. Überblick
2. Recap Anforderungsanalyse und Domänenmodellierung
3. **Recap Softwarearchitektur und Design**
4. Recap Entwurf mit Design Patterns
5. Info zur Semesterendprüfung und Q&A
6. Bearbeitung einer Fallstudie (alte SEP)
7. Wrap-up

Aufgabe 14.2 (10')

Diskutieren Sie in Murmelgruppen folgende Fragen:

- Was ist eine Softwarearchitektur und wie beschreibt man sie?
- Was sind die Ziele einer geschichteten Softwarearchitektur und wie beschreibe ich sie?
- Wie realisiere ich einen Use Case mit Klassen, die klare Verantwortlichkeiten haben, wartbar und einfach erweiterbar sind?
- Wie modelliere ich mein Design (statisches und dynamisches Modell) mit der UML, um es diskutieren und evaluieren zu können?
- Wozu sind die GRASP Prinzipien und Patterns im Design nützlich?

Agenda

1. Big Picture
2. Recap Anforderungsanalyse und Domänenmodellierung
3. Recap Softwarearchitektur und Design
4. **Recap Entwurf mit Design Patterns**
5. Info zur Semesterendprüfung
6. Bearbeitung einer Fallstudie (alte SEP)
7. Wrap-up

Aufgabe 14.3 (10')

Diskutieren Sie in Murmelgruppen folgende Fragen:

- Was sind Design Patterns und wozu sind sie nützlich?
- Wie wird ein Design Pattern beschrieben?
- Welche GoF Design Pattern kennen Sie und was ist ihr Zweck?
- Was sind Trade-offs bei der Anwendung eines Design Patterns?

Agenda

1. Überblick
2. Recap Anforderungsanalyse und Domänenmodellierung
3. Recap Softwarearchitektur und Design
4. Recap Entwurf mit Design Patterns
- 5. Info zur Semesterendprüfung**
6. Bearbeitung einer Fallstudie (alte SEP)
7. Wrap-up

Leistungsnachweise (1/2)

- **2 Quizzes** in der Vorlesung während dem Semester zur formativen Lernkontrolle (jeweils 10 Multiple-Choice-Fragen, 15 Min.)
 - 1 Quiz zur Anforderungsanalyse
 - 1 Quiz zur Softwarearchitektur und Design
 - Die erreichte Note zählt zu 10% für die Gesamtnote
- **Unterrichtsaufgaben** während des Semesters
 - Max. 3 Punkte pro LE
 - Die erreichte Note zählt zu 20% für die Gesamtnote
- **Semesterendprüfung (SEP)** (90 Min.)
 - Umfang: Vorlesung, abgegebene Unterlagen, Aufgaben aus dem integrierten Praktikum und der Wissenssicherung
 - Die erreichte Note zählt zu 70% für die Gesamtnote



Leistungsnachweise (2/2)

- **Inhalte der Semesterendprüfung (SEP)**
 - 25-30% Anforderungsanalyse und Domänenmodellierung
 - 10% Fragen zu Diagrammen und Design mit der UML
 - 50-65% Softwarearchitektur und Design, Design Patterns
 - Umfang ist alles aus den Lerneinheiten 1-9 und zusätzliche GoF Design Patterns aus der Vertiefung LE 13 (Abstract Factory, Factory Method, Command, Template Method)
 - Keine Fragen oder Aufgaben zu diskutierter Technologie in den Vertiefungen 1-4 (z.B. zu JavaFX, RMI, REST, WebSocket, JDBC, JPA, Java Reflection etc.)
 - Inhalte aus den Lernaufgaben und Wissenssicherungen.



Agenda

1. Überblick
2. Recap Anforderungsanalyse und Domänenmodellierung
3. Recap Softwarearchitektur und Design
4. Recap Entwurf mit Design Patterns
5. Info zur Semesterendprüfung
6. **Bearbeitung einer Fallstudie (alte SEP)**
7. Wrap-up

Agenda

1. Überblick
2. Recap Anforderungsanalyse und Domänenmodellierung
3. Recap Softwarearchitektur und Design
4. Recap Entwurf mit Design Patterns
5. Info zur Semesterendprüfung
6. Bearbeitung einer Fallstudie (alte SEP)
- 7. Wrap-up**

Besprechung Ihres Feedbacks zur Vorlesung

- Gerne nehmen wir noch weiteres Feedback zur Vorlesung entgegen:
 - Was hat gefallen?
 - Was hat nicht gefallen?
 - Was muss geändert werden?



Herzlichen Dank und weiterhin viel Erfolg im Studium!