Software Engineering

Fachhochschule Bielefeld

Campus Minden

Wintersemester 2019/2020

Prof. Dr. Jörg Brunsmann

Inhalte und Lernziele

- Was ist Softwaretechnik und warum ist Softwaretechnik wichtig?
- Lernen, methodisch und strukturiert im Team mit agilen Vorgehensmodellen Software zu entwickeln
- Unterschiedliche Vorgehensmodelle und deren Phasen beschreiben können
- Die Phasen "Anforderungsanalyse", "Systemanalyse", "Softwareentwurf", "Implementierung" und "Testen" beherrschen
- Architekturmuster und Entwurfsmuster beherrschen und anwenden können.
- Was versteht man unter Softwareprüfung und welche Verfahren der Softwareprüfung existieren?
- UML-Diagrammtypen beherrschen und in den richtigen Phasen anwenden können
- Qualitätsmerkmale von Software kennen und beschreiben können
- Software spezifizieren können

Organisation

Vorlesung

- Jeder erhält ein Thema und präsentiert das Thema vor den anderen im Vorlesungsunterricht
- Länge der Präsentation: 15 Minuten
- Max. sechs Studierende pro Vorlesungsunterricht (insgesamt 90 Minuten)
- Die Präsentation ist Bestandteil der Praktikumsnote
- Die Präsentation ist vor dem Präsentationstermin in einen ILIAS-Ordner namens "Präsentationen" hochzuladen
- Inhalte sind mit mir im Vorfeld abzusprechen (z.B. im Praktikum)
- Format der Präsentation: PPT
- Format des Dateinamens: x.y.Thema.ppt
  - "x" und "y" gemäß Ablauf der Vorlesung
  - Sollte x kleiner sein als 10, dann eine 0 davor

Praktikum

- Softwareprodukt planen, entwerfen und implementieren
- Gültige Gruppengrößen: 1-3 Teilnehmer
- Bewertungkritierien (mindestens 4.0 zum Bestehen)
    - Problemlösungskompetenz
    - Umfang der Funktionalität
    - Qualität
    - Sorgfalt
    - Selbstinitiative und Einsatz
    - Kontinuierlicher Fortschritt
    - Komplexität
    - Kreativität
    - Teamarbeit
    - gleichverteilte Aufgabenverteilung
- Theorie: Klausur (120 Minuten). Min. 50% der Punkte für Bestehen. Keine Hilfsmittel erlaubt.
- Gesamtnote ist das Mittel aus Praktikumsnote und Klausurnote

Themen

1. Einleitung und Grundlagen

Di. 22.10.2019

1. Einleitung [Kai Penner]
    - Was ist Software? Welche Eigenschaften besitzt Sofware?
    - Welche Arten von Software existieren? (Systemsoftware, Anwendungssoftware etc.)
    - Was ist Auftragssoftware?
    - Was ist der Unterschied zwischen einem Softwaresystem vs. Softwareprodukt?
2. Softwarequalität [Florian Löwen]
    - Was ist Softwarequalität? (Qualitätsmerkmale)
    - Was ist die ISO 25010?
    - Welche unterschiedlichen Perspektiven auf Softwarequalität existieren? (Kunde, Informatiker, etc.)
    - Was ist das Qualitätsdreieck?
    - Welche Arten von Dokumentation existieren? (Systemdokumentation, Projektdokumentation, Benutzerdokumentation)
3. Softwareprojekte [Yannic Döll]
    - Was ist ein Softwareprojekt?
    - Welche Projektgrößen existieren?
    - Welche Kostenarten existieren in einem Softwareprojekt?
    - Welche Beteiligte existieren in Softwareprojekten? (Informatiker, Benutzer, Kunde, etc.)
    - Welche Rollen existieren aus Informatiksicht? (Projektleiter, Programmierer, Architekt, etc.)
    - Welche Gründe existieren für das Scheitern von Projekten?
    - Welche Gründe gibt es für den Erfolg von Projekten?
4. Software Engineering [Bjarne Küper]
    - Was ist Software Engineering? (Vergleich von Definitionen)
    - Was sind die Inhalte von Software Engineering? (Vorgehensmodelle, Methoden, Werkzeuge und Architekturen)
5. Software-Ingenieur [Gysar Flegel]
    - Was ist ein Software-Ingenieur?
    - Eigenschaften eines idealen Software-Ingenieurs (Kommunikationsfähigkeit, Neugier, etc.)

2. Klassische Vorgehensmodelle

Di. 29.10.2019

1. Klassische Vorgehensmodelle [Lukas Hitzemann]
    - Was ist ein Vorgehensmodell?
    - Bestandteile von Vorgehensmodellen (Phase, Aktivität, Artefakt, Rolle)
2. Phasenmodell [Sascha Rührup]
    - Was ist ein Phasenmodell?
    - Eigenschaften typischer Phasen: Anforderungsanalyse, Systemanalyse, Entwurf, Implementierung, Test, Wartung
    - Aufgaben typischer Rollen: Projektmanager, Anforderungsanalytiker, Software-Architekt, Software-Designer, Software-Entwickler, Tester, Administrator
3. Wasserfall-Modell [Marco Schwier]
    - Eigenschaften, Vorteile, Nachteile
4. V-Modell [Niklas Lange]
    - Eigenschaften, Vorteile, Nachteile
5. Inkrementelle und iterative Vorgehensmodelle und Prototypen [Akin Sansar]
    - Eigenschaften, Vorteile, Nachteile

3. Agile Vorgehensmodelle

Di. 5.11.2019

1. Agile Vorgehensmodelle [Koch]
    - Was ist ein agiles Vorgehensmodell?
    - Was ist das agile Manifest?
    - Eigenschaften, Werte, Prinzipien und Praktiken agiler Prozesse
    - Vergleich von klassischen und agilen Vorgehensmodellen
2. Extreme Programming [Christoph Raitzig]
    - Inhalte und Ablauf
    - Praktiken (Paarprogrammierung, TDD)
3. Scrum [Wortmann]
    - Ablauf
    - Meetings, Rollen, Artefakte
4. Kanban [Rafael Berger]
    - Inhalte, Ablauf
    - Sechs Kernpraktiken
5. Vergleich von Kanban und Scrum [Göker]
    - Vergleich (Rollenverteilung, Arbeitsbegrenzung, Iterationsdauer et.)
    - Auswahlhilfe (Wann ist was besser geeignet)

4. Objektorientierte Modellierung mit UML-Diagrammen

Di. 12.11.2019

1. Objektorientierte Modellierung [Mathis Block]
    - OOA/OOD
2. 4+1-Sichten-Softwarearchitekturmodell [Nick Kötter]
3. Modellierung mit UML-Diagrammen [Philipp Fricke]
    - Was ist UML?
    - Was sind statische und dynamische Diagrammtypen?
    - Zuordnung der UML-Diagrammtypen zu Phasen der Softwareentwicklung
4. Vorlesungsrelevante Diagrammtypen [Ruben Klinksiek]
    - Kurze Darstellung des Zwecks und der Inhalte von: Klassendiagramm, Paketdiagramm, Objektdiagramm, Use-Case-Diagramm, Aktivitätsdiagramm, Zustandsdiagramm, Sequenzdiagramm, Komponentendiagramm, Verteilungsdiagramm
5. Modellgetriebene Softwareentwicklung [Sebastian Peltner]

5. Anforderungsanalyse

Di. 19.11.2019

1. Anforderungsanalyse [Nicolas Guzik]
    - Was ist Anforderungsanalyse?
    - Was sind Anforderungen? (funktional, nicht-funktional)
    - Probleme bei der Anforderungsermittlung
    - Stakeholder
    - Wege zu und Quellen von Anforderungen
    - Dokumentation mittels Lasten- und Pflichtenheft
2. Satzschablonen [Lukas von der Heide]
    - Satzschablonen (erfunden von Chris Rupp)
    - Aufbau
    - Typen von Systemaktivitäten
    - Schlüsselworte und Verbindlichkeiten
    - Prozessworte und Bedingungen
3. Textuelle Use Case Definition [Joshua Martin]
    - Schablone zur textuellen Definition von Use Cases
    - Abläufe und Szenarien
    - Normalabläufe, Alternativabläufe, Ausnahmeabläufe
4. UML-Use Case Diagramme [Marten Daniel]
    - Was ist ein Use Case?
    - Notationselemente: Akteure, Use Cases, Beziehungen
5. User Stories und GUI-Mockups [Fabian Engelke]
    - Aufbau von User Stories
    - Card, Conversation, Confirmation
    - INVEST-Kriterien
    - User-Stories vs. Use Case

6. Systemanalyse mit Aktivitätsdiagrammen

Di. 26.11.2019

1. Aktivitätsdiagramme 1 [Reckmann]
    - Grundlegende Notationselemente
    - Aktionen, Swimmlanes, Knoten
2. Aktivitätsdiagramme 2 [Peters]
    - Tokenmodell
    - Datenfluss vs. Kontrollfluss
3. Aktivitätsdiagramme 3 [Sebastian Steinmeyer]
    - Subaktivitäten
    - Signale und asynchrone Ereignisse
    - Unterbrechungsbereiche
4. BPMN [Karl Neitmann]
5. Ereignisgesteuerte Prozessketten [Umut Güngör]
6. CRC-Karten vs. Verb-Substantiv-Methode [Efthimia Niaka]
    - Class-Responsibility-Collaboration-Karte
    - Identifikation von Klassen und Objekte und deren Relationen
    - Beispiele

7. Modellierung mit Klassen-, Sequenz- und weiteren UML-Diagrammen

Di. 3.12.2019

1. Domain Object Modellierung mit Klassendiagrammen 1 [Jonas Strunk]
    - Grundlegende Notationselemente: Klassen, Attribute, Sichtbarkeit, Multiplizität, Assoziation, Aggregation, Komposition, Vererbung, Implementierung
    - Beispiele
2. Domain Object Modellierung mit Klassendiagrammen 2 [Schnittger]
    - Weitere Notationselemente: Stereotypen, Merkmale, Verantwortlichkeiten, Rollen, Navigierbarkeit, Assoziationsklassen
    - Beispiele
3. Sequenzdiagramme und Kommunikationsdiagramme [Dominik Scheffler]
    - Grundlegende Notationselemente: Objekte, Lebenslinie, Aktivierungsbalken, Nachrichten (synchron, asynchron)
    - Weitere Notationselemente: Steuerungsoperatoren
    - alt, opt, loop, break
    - seq, strict, par, critical
    - ignore, consider, assert, neg
    - Beispiele
    - Kommunikationsdiagramme als Alternative zu Sequenzdiagrammen
4. Zustandsdiagramme [Markus Herbusch]
    - Grundlegende Notationselemente: Zustand, Transition, Startzustand, Endzustand, Entscheidungsknoten
    - Weitere Notationselemente: Komplexe Zustände, Ereignistypen (CallEvent, TimeEvent, etc.), Parallelisierungknoten, Synchronisierungsknoten
    - Beispiele
5. Komponentendiagramme und Paketdiagramme [Alexander Nolting]
    - Notationselemente
    - Beispiele
6. Verteilungsdiagramme und Objektdiagramme [Waldemar Reger]
    - Notationselemente
    - Beispiele

8. Softwareentwurf

Di. 10.12.2019

1. Softwareentwurf [Alexander Del]
    - Tätigkeiten im Softwareentwurf
    - Qualitätsmerkmale für "gute" Software
2. Entwurfsprinzipien [Born]
    - Separation of Concerns
    - Keep It Stupid Simple.
    - You Ain’t Gonna Need It.
    - Don't repeat yourself.
    - Composition Over Inheritance
3. Entwurfsziele [Wilczek]
    - Kopplung und Kohäsion
    - Beispiele
4. Bad Code Smells und Anti Patterns [Fabian List]
    - Duplicate Code, Long Method, etc.
    - Dead Code, Copy Paste, Blob (God-Class), Poltergeist
    - Beispiele
5. Domain Driven Design [Jonas vom Braucke]
    - Domain Object Model
    - Entitäten, Value objects, Aggregates, Services

9. MV*-Architekturmuster

Di. 17.12.2019

1. Softwaremuster [Jannis Herbertz]
    - Was sind Architekturmuster?
    - Was sind Entwurfsmuster?
2. Model View Controller [Windhorn]
    - Eigenschaften
    - Vor- und Nachteile
    - Active MVC, Passive MVC
    - Ablauf
    - Beispiel
3. Web MVC [Madlehn]
    - Eigenschaften
    - Vor- und Nachteile
    - Active MVC, Passive MVC
    - Ablauf
    - Beispiel
4. Model View Presenter [Maximilian Frömelt]
    - Eigenschaften
    - Vor- und Nachteile
    - Passive View, Supervising Presenter
    - Ablauf
    - Beispiel
5. Model View ViewModel [Kevin Gerzen]
    - Eigenschaften
    - Vor- und Nachteile
    - Databinding
    - Ablauf
    - Beispiel

10. Entwurfsmuster

Di. 7.1.2020

1. Entwurfsmuster 1 [Dennis Petana]
    - Welche Entwurfsprobleme lösen Entwurfsmuster?
    - Vergleich von Mustern, Algorithmen und Frameworks
    - Vor- und Nachteile
    - Arten von Entwurfsmustern (Verhalten, Struktur, Erzeugung)
    - Gang of Four Patterns
2. Entwurfsmuster 2 [Timo Kohlmeier]
    - Proxy, Abstract Factory, Factory Method
    - UML-Diagramme
    - Anwendungsbeispiele
    - Implementierungsbeispiel
3. Entwurfsmuster 3 [Rostek]
    - Adapter, Observer, Facade
    - UML-Diagramme
    - Anwendungsbeispiele
    - Implementierungsbeispiel
4. Entwurfsmuster 4 [Buschendorf]
    - Composite, State
    - UML-Diagramme
    - Anwendungsbeispiele
    - Implementierungsbeispiel
5. Entwurfsmuster 5 [Dümke]
    - Visitor
    - UML-Diagramme
    - Anwendungsbeispiele
    - Implementierungsbeispiel

11. Implementierung und Continuous Integration

Di. 14.1.2020

1. Übersetzen von UML Klassendiagramme in objektorientierte Programmiersprachen [Michael Dering]
    - Klasse, Schnittstellen, Attribute, Methoden, Sichtbarkeiten
    - Aggregation, Komposition, Assoziation
    - Beispiel
2. Continuous Integration und CI-Server [Leon Brandt]
3. Continuous Delivery / Deployment [Luca Berneking]
4. DevOps [Justin Drögemeier]

12. Softwareprüfung

Di. 21.1.2020

1. Grundlagen [Leon Stümpeley]
    - Dynamische Tests (Unit, Integrations, Systemtest, etc.)
    - Statische Tests (Review, Walkthrough, etc.)
2. Metriken [Marcell Wach]
    - Kontrollflussgraph
    - Normalisierung
    - Konventionelle und objektorientierte Metriken
    - Zyklomatische Komplexität
    - Lack of Cohesion in Methods
3. Funktionsorientierter Test (Black Box Test) [Philipp Steib]
    - Äquivalenzklassenbildung
    - Grenzwertanalyse
4. Strukturorientierter Test (White-Box-Test) [Oliver Maciejewski]
    - Statement Coverage, Branch Coverage, Path Coverage, Condition Coverage