DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Zielgruppe: **Wirtschaftsinformatik**

Modul: **Software Engineering**

Foliensatz: # Organisatorisches

Prof. Dr. Malte Prieß

d-fine

GEOMAR
Helmholtz-Zentrum für Ozeanforschung Kiel

future ocean
KIEL MARINE SCIENCES

C A U
Christian-Albrechts-Universität zu Kiel

Leibniz Universität Hannover

Max-Planck-Institut für Gravitationsphysik
Albert-Einstein-Institut

DREILINDEN GYMNASIUM

Lehrveranstaltung

Modul	Software Engineering (3. Semester Sep. bis Nov.)
Modulverantwortlicher	Prof. Dr. Malte Prieß
ECTS-Credits	5 ECTS-Credits
SWS	6 SWS
Workload (je Theorieblock)	Vorlesung + Übung: 9 * 6 SWS * 0,75 h = 40,5 h Präsenzstudium: 40,5 h Selbststudium (Vor- und Nachbereitung): 84,5 h Workload: 40,5 + 84,5 h = 125 h (Präsenzstudium ~32%)
Prüfungsleistungen	Klausur, voraussichtlich 120 Minuten

Software Engineering

- Vorgehensmodelle und agile Softwareentwicklung
- Requirements-Engineering
- Entwurf und Implementierung
- Grundlagen der Softwarearchitektur
- Cloudbasierte Software, Serviceorientiertes Software Engineering
- Zuverlässige Programmierung
  - Prinzipien des Software Engineering
  - Entwurfsmuster
  - Refactoring
  - Testen von Software
- Code- und Konfigurationsmanagement
- DevOps-Automatisierung

Modulplanung

Woche	Vorlesung	Übung
Woche 1	Prinzipien des Software Engineering	Warmup Ticketsystem, Versionssystem Git (Teil 1) – Git Basics
Woche 2	Vorgehens-/Prozessmodelle und Agile Softwareentwicklung	Git (Teil 2) – Git Branching
Woche 3	Workshop Scrum	
Woche 4	*Studienfahrt*	
Woche 5	Anforderungsanalyse und Spezifikation	Anforderungsanalyse und Spezifikation
Woche 6	Entwurf	Entwurf
Woche 7	Architektur- und Entwurfsmuster	Architektur- und Entwurfsmuster
Woche 8	Implementierung und Test	Implementierung und Test
Woche 9	Voraussichtlich Wiederholung	
Woche 10	*Prüfungswoche*	

Zu beachten: Inhalte können abweichen; die Aufteilung kann sich verschieben

Astah UML

http://astah.net/student-license-request

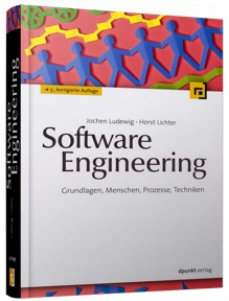
Unterstütze UML Diagramme:

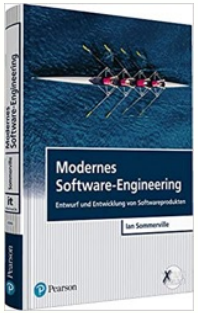
- Class
- UseCase
- Sequence
- Activity
- Communication
- Statemachine
- Component
- Deployment
- Composite Structure
- Object and Package Diagrams

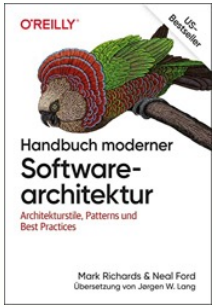
Literatur

Software Engineering (*es gilt jeweils die aktuelle Auflage*)

- *Ludewig, Lichter*: Software Engineering: Grundlagen, Menschen, Prozesse, Techniken
- *Sommerville: Software Engineering*
- *Sommerville: Modernes Software-Engineering: Entwurf und Entwicklung von Softwareprodukten*
- *Witte: Testmanagement und Softwaretest, Theoretische Grundlagen und praktische Umsetzung*
- *Richards, Ford:* Handbuch moderner Softwarearchitektur
- *Becker, Kugeler, Rosemann*: Prozessmanagement: Ein Leitfaden zur prozessorientierten Organisationsgestaltung
- *Allweyer:* BPMN 2.0 Business Process Model and Notation. Einführung in den Standard für die Geschäftsprozessmodellierung



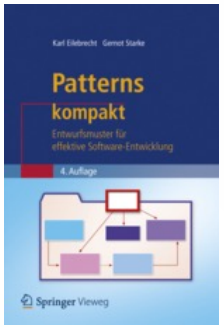


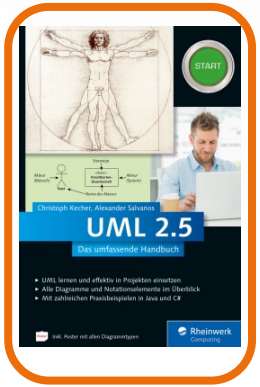


Literatur

UML und Entwurfsmuster (*es gilt die aktuelle Auflage*)

- *Oesterreich*: Analyse und Design mit der UML 2.5: Objektorientierte Softwareentwicklung, Oldenbourg.
- *Gamma, Helm, Johnson, Vlissides*: Entwurfsmuster, Addison-Wesley.
- *Eilebrecht, Starke*: Patterns kompakt - Entwurfsmuster für effektive Software-Entwicklung, Springer.
- *Kecher, Salvanos*: UML 2.5 Das umfassende Handbuch, Rheinwerk Verlag GmbH.

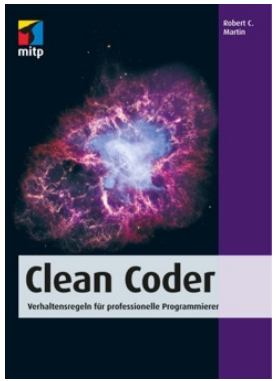


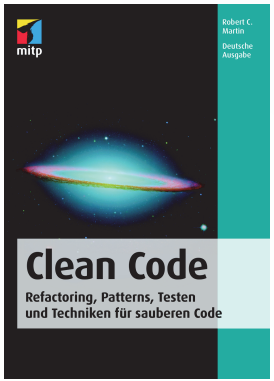


Literatur Zusatz : Clean Architecture, Clean Code, Clean Coder

- Robert C. Martin („Uncle Bob“)
  - Clean Code: Refactoring, Patterns, Testen und Techniken für sauberen Code / Clean Code: A Handbook of Agile Software Craftsmanship (Robert C. Martin)
  - Clean Coder: Verhaltensregeln für professionelle Programmierer / The Clean Coder: A Code of Conduct for Professional Programmers
  - Clean Architecture: Das Praxis-Handbuch für professionelles Softwaredesign, Regeln und Paradigmen für effiziente Softwarestrukturen / Clean Architecture: A Craftsman's Guide to Software Structure and Design

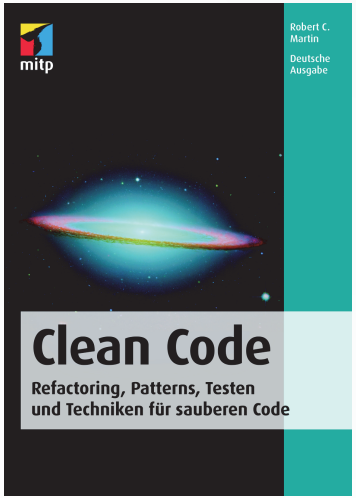
- https://cleancoders.com





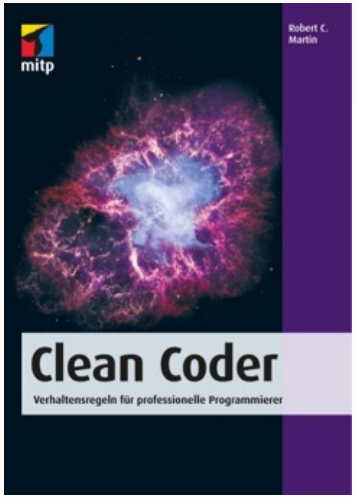
Literatur Zusatz: Clean Code

- Clean Code
  - Guten Code schreiben und schlechten Code überarbeiten.
  - Welche Prinzipien, Patterns und Praktiken sind anzuwenden, um sauberen Code zu schreiben?
  - Wie kann schlechter Code in guten Code umgewandelt werden?
  - Saubere Fehlerbehandlung sowie die Anwendung sauberen Codes während der Testphase.



Literatur Zusatz: Clean Coder

- Clean Coder
  - Es geht um Disziplinen, Techniken, Tools und Methoden.
  - Mit Konflikten, knappen Zeitplänen und unvernünftigen Managern umgehen.
  - Beim Programmieren im Fluss bleiben und Schreibblockaden überwinden.
  - Mit unerbittlichem Druck umgehen und Burnout vermeiden.
  - Zeitmanagements optimieren.
  - Für Umgebungen sorgen, in denen Programmierer und Teams wachsen und sich wohlfühlen.
  - Wann Sie Nein sagen sollten und wie Sie das anstellen.
  - Wann Sie Ja sagen sollten und was ein Ja wirklich bedeutet.
  - ...



Ausblick: Clean Architecture

- Clean Architecture
  - Praktische Lösungen für den Aufbau von Softwarearchitekturen.
  - Allgemeingültige Regeln für die Verbesserung der Produktivität in der Softwareentwicklung über den gesamten Lebenszyklus.
  - Wie Softwareentwickler wesentliche Prinzipien des Softwaredesigns meistern, warum Softwarearchitekturen häufig scheitern und wie man solche Fehlschläge verhindern kann.