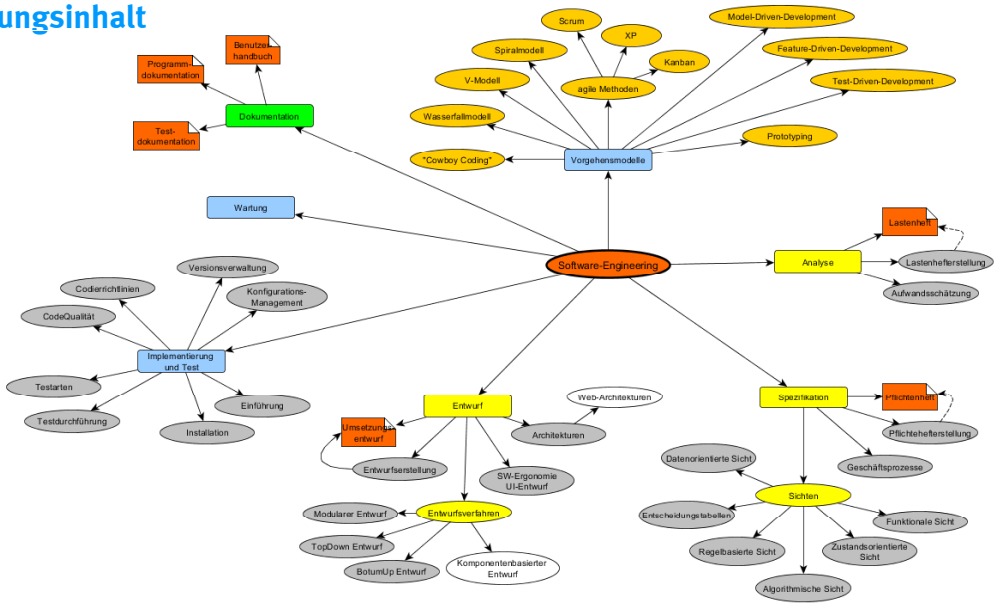
Vorlesung Software-Engineering I

... im 3. und 4. Semester.

00. Einführungsteil – Vorlesungsinhalt



Dipl. Ing.(FH) Frank-Michael Hoyer

hoyer@lehre.dhbw-stuttgart.de

http://www.dhbw-stuttgart.de/~hoyer

Vorlesung
Software-Engineering I
seit WS2010
an der DHBW-Stuttgart

Seit 1994 bei Fa. Festo AG & Co. KG, Esslingen-Berkheim

Dipl.Ing.(FH)
Frank M. Hoyer
frank.hoyer@festo.com
0711/347-3436

Bereich: Pneumatic Automation
Abteilung: Project Office Portfoliomenagement
Team : agile Product Development

Aufgaben:
- Projektleiter agile Software Entwicklung
- Sprecher Arbeitskreis Datenverarbeitung in der Entwicklung
- Koordinator DV-Beschaffung Bereich Entwicklung

http://www.dhbw-stuttgart.de/themen/bachelor/fakultaet-technik/informatik/profil/

Ziele des Studiengangs Informationstechnik

- Die fundierte und kontinuierlich aktualisierte Vermittlung **informationstechnischer, betriebswirtschaftlicher und interdisziplinärer Kompetenzen**.
- Befähigung zur Mitgestaltung und Weiterentwicklung **zukünftiger IT-Systeme** und -Anwendungen
- theoretische und praxisintegrierte Vorbereitung auf die **Mitwirkung und Leitung** von firmen- wie auch länderübergreifenden IT-Projekten
- Steigerung der **Sprach- und Sozialkompetenz**, u.a. durch englischsprachige Vorlesungen und studienbegleitende Theoriephasen und Praxiseinsätze im Ausland

http://www.dhbw.de/fileadmin/user/public/SP/Studienbereich_Technik/Kernmodul/T2INF2003.pdf

SW-Engineering I – Qualifikationsziele und Kompetenzen

- Die Studierenden kennen die **Grundlagen des Softwareerstellungsprozesses**.
- Sie können eine vorgegebene Problemstellung **analysieren** und rechnergestützt Lösungen **entwerfen, umsetzen** und **dokumentieren**.
- Sie kennen die **Methoden** der jeweiligen Phasen und können sie **anwenden**.
- Sie können Lösungsvorschläge für ein gegebenes Problem konkurrierend **bewerten** und **korrigierende Anpassungen** vornehmen.
- Die Studierenden können sich mit Fachvertretern über Problemanalysen und Lösungsvorschläge, sowie über die Zusammenhänge der einzelnen Phasen **austauschen**.
- Sie können einfache Softwareprojekte **autonom** entwickeln oder bei komplexen Projekten effektiv in einem **Team** mitwirken.
- Sie können ihre Entwürfe und Lösungen **präsentieren** und **begründen**.
- In der Diskussion im Team können sie sich kritisch mit **verschiedenen Sichtweisen** auseinandersetzen und diese erläutern.
- Sie **bewerten** die eingesetzten Technologien und schätzen ihre Folgen ab.
- Die Studierenden können sich selbsständig in **Werkzeuge** einarbeiten.
- Sie verbinden den Softwareentwicklungsprozess mit Techniken des Projektmanagement und beachten während des Projekts **Zeit- und Kostenfaktoren**.

http://www.dhbw.de/fileadmin/user/public/SP/Studienbereich_Technik/Kernmodul/T2INF2003.pdf

Lehr- und Lehrinhalte **SW-Engineering I**

- **Vorgehensmodelle**
- **Phasen** des SW-Engineering und deren Zusammenhänge
- **Analyse**: Lastenheft
- **Spezifikation:** Pflichtenheft, Anwendungsfälle
- **Methoden** zur Repräsentation von Algorithmen
- Datenmodellen, Funktionsweisen, Zustands- und Regelabhängigkeiten
- **Entwurf:** SW-Architektur, Systementwurf, Schnittstellenentwurf, Klassendiagramme
- **Implementierung** und **Test**
- Codierrichtlinien und **Codequalität**, systematisches Testen, Testarten und Testdurchführung, Installation und Einführung
- **Betrieb** und **Wartung**.
- Phasenspezifische werden die verschiedenen Arten der **Dokumentation** behandelt.

Labor/Gruppenarbeit:

Ein **komplexes Problem** wird als **Projekt** mit allen Phasen von den Studierenden erarbeitet und dokumentiert.

Inhaltsübersicht **Software-Engineering I** (3. und 4. Semester)

Vorlesung und Übung:

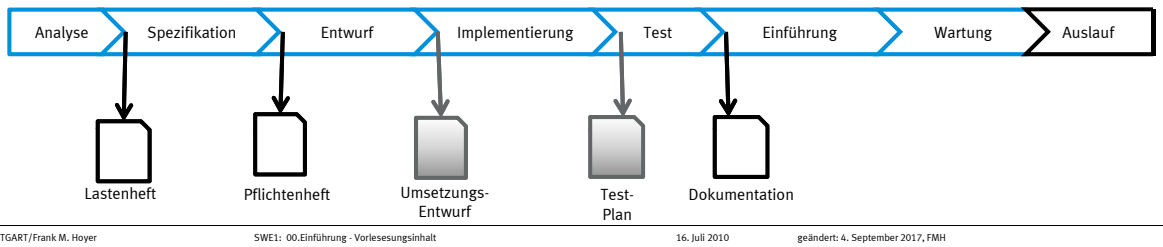
1. Vorgehensmodelle
2. Analyse
3. Spezifikation
4. Entwurf
5. Implementierung und Test
6. Wartung und Pflege
7. Dokumentation

FMH (44h)

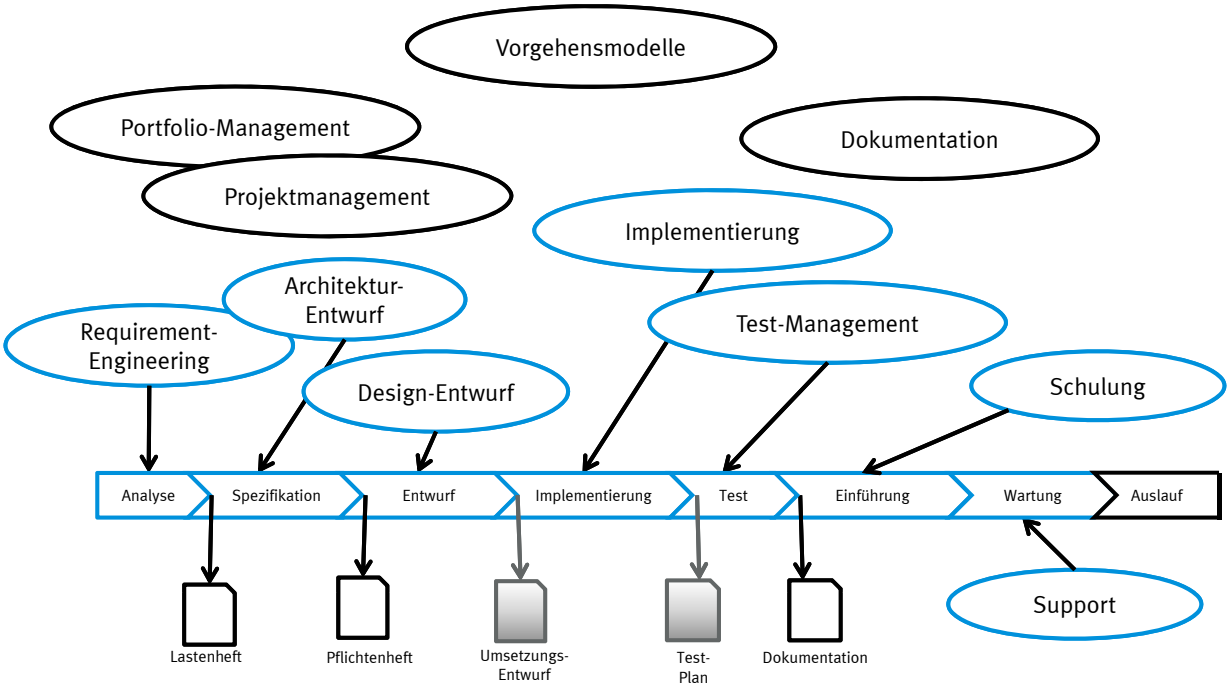
DNR (0h)

Dipl.Ing.(FH)
Frank M.Hoyer

Prof. Dr.
Doris Nitsche-Ruhland

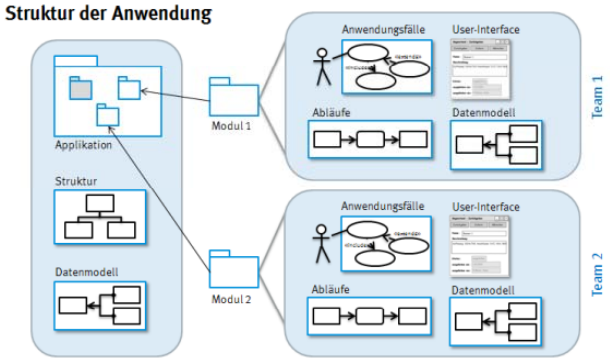


Themenüberblick Software-Engineering I

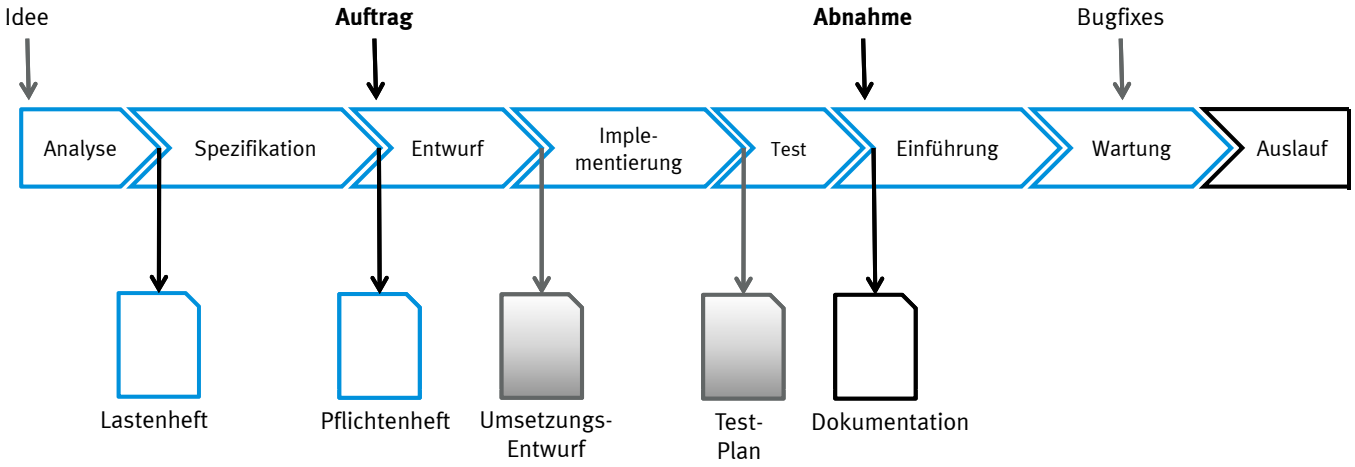


Kenntnisse und Fertigkeiten Software-Engineering I

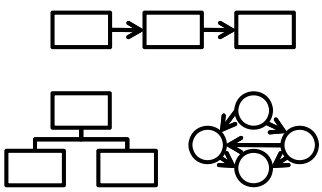
- Struktur und Inhalt von **Lasten-** und **Pflichtenheften** kennen
- Verfahren zur **Aufwandsschätzung** kennen
- **Basiskonzepte** (Funktionen, Daten, Abläufe) kennen
- **Strukturierte- / Objektorientierte Analyse** (SA/OO) anwenden
- **Geschäftsprozesse** analysieren, **Anwendungsfälle** erstellen
- Modularer, Top-Down und Bottom-Up **Entwurf** kennen
- **SW-Architekturen** kennen
- **UML** Diagramme kennen
- ***Entity-Relationship*** *(ER) Diagramme kennen*
- **Data Dictionary** (DD) erstellen
- **Kontrollstrukturen, Entscheidungstabellen** kennen
- **Zustandorientierte** u. **regelbasierte Sichten** kennen
- **Testebenen, Testverfahren** kennen
- Grundlagen der **Dokumentation** kennen



Phasenmodell der Software-Entwicklung

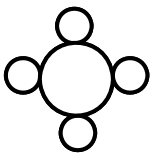


Parallele Inhaltsebenen



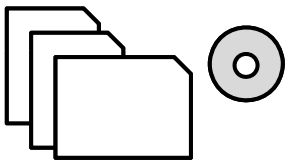
Standards

- Vorgehensmodelle
- Methoden, Modelle
- Verfahren, Notationen
- Projektmanagement



Best Practice

- Refactoring
- Pattern
- Technologien
- Werkzeuge



Reale Welt

- Soft Skills
- Beispiele
- Erfahrungen
- Teamplay

Vom Problem zur Lösung

oder: Domänen und ihre Sprachen

Stakeholder
Anwender
Auftraggeber

Projektleiter
Programmierer

Modell

Problem

Lösung

Programm

Die reale Welt mit der Domäne der Fachabteilung

Das Modell der Domäne

Die Problem-beschreibung Im Kontext der Domäne

Der Lösungsentwurf zur Abdeckung des Problems

Die Umsetzung des Lösungsentwurfs

Domänenspezifische Sprache (DSL)

Lastenheft

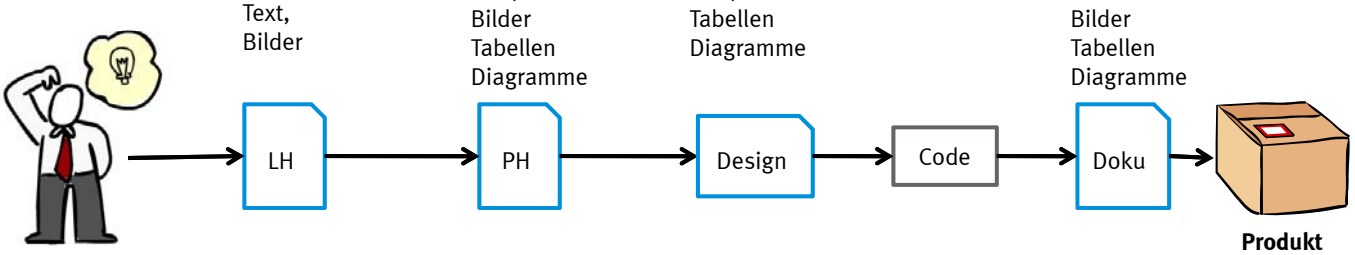
Pflichtenheft

Sourcecode

Problemdomäne

Lösungsdomäne

Artefakte und ihre Inhalte



Produktidee

Produktkarton
Mindmaps
Conceptmaps

Lastenheft (Was)

Übersicht
Zielgruppe
Anwendungsfälle
Anforderungen

Pflichtenheft (Wie)

Übersichten
Umfang
Abgrenzung
Aufgaben
Abläufe, Zustände
Daten,
Ein-/Ausgaben

Umsetzungsentwurf

Übersichten
Modularisierung
Objekte, Klassen,
Funktionen, Daten

Testplan
- Testfälle

Programmdokumentation
-Doku der IST-Umsetzung
(aus Umsetzungsentwurf)
- Inbetriebnahmedoku

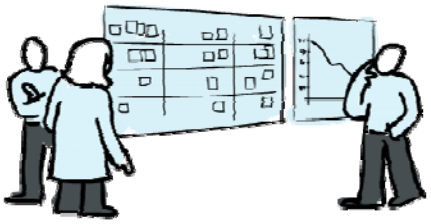
Anwenderdokumentation
-Bedienungsanleitung
- HowTo‘s

Inhaltsübersicht Software-Engineering I

1. Vorgehensmodelle

Phasen des SW-Engineerings
und deren Zusammenhänge

Überblick über die verschiedenen Prozessmodelle



Vergleich der Vorgehensmodelle:

- „Cowboy Coding“
- Wasserfallmodell
- Spiralmodell
- V-Modell
- Agile SW-Entw. (Scrum/XP/Kanban)

- Prototyping
- Modellgetriebene SW-Entw.
- Feature-Driven-Development (FDD)
- Test-Driven-Development

➢ Klausur (30 Min.)

Inhaltsübersicht Software-Engineering I

2. Analyse

Aufwandsschätzung, Lastenheft

Methoden der Aufwandsschätzung
Erstellen eines Lastenhefts.

- Produkt-Idee („Produktkarton“)
- Anforderungsanalyse (User Story‘s)
- Wichtigkeit und Umfang (Prio und Aufwand)
- Soft-Skills
- Anforderungs-Domaine analysieren (DSL, Glossar)
- Requirements-Engineering
- Projekt-Management (Releaseplan)

- Stakeholder, Projektbeteiligte
- Portfolio-Management (das große Ganze)

Labor: Rollenspiel „Anforderungserhebung“

Inhaltsübersicht Software-Engineering I

3. Spezifikation

Pflichtenheft, Geschäftsprozesse, SA, Methoden zur Repräsentation von Algorithmen, Datenmodellen, Funktionsweisen, Zustands- und Regelabhängigkeiten

Pflichtpunkte:

- Algorithmische Sicht
  (Kontrollstrukturen, Pseudocode, Struktogramme)
- Funktionale Sicht
- Entscheidungstabellen
- Zustandsorientierte Sicht (Zustandsgraph, evtl. Petrinetz)
- Evtl. Regelbasierte Sicht
- Entity-Relationship-Modell (wird in DB-Vorlesung behandelt)
- grundlegende Diagramme z.B. Klassendiagramme,
  falls noch nicht in anderen Vorlesungen behandelt (kein OO)
- Geschäftsprozesse
- Strukturierte-Analyse (SA)
- Pflichtenheft

- Problem-Domaine modellieren
- UseCases/UserStory‘s
- Abläufe/Workflows
- Funktionseinheiten und Zusammenhänge
- Datenstrukturen
- Regeln/Zustände

- System –› Subsystem -› Komponente

- Ist-Zustand/Soll-Zustand

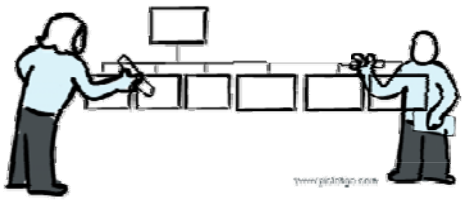
Labor: Übungen an konkreten Beispielen

Inhaltsübersicht Software-Engineering I

4. Entwurf

SW-Architekturen, Systementwurf, Schnittstellenentwurf, Klassendiagramme

- Modularer Entwurf,
- Top-Down, Bottom-Up Entwurf
- SW-Ergonomie im Hinblick auf UI-Entwurf,
- Komponentenbasierter Entwurf
- Software-Architekturen



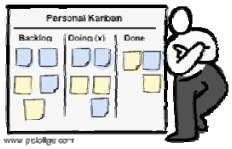
- Lösungs-Domaine modellieren
- Scoping (Abgrenzung zur Umwelt)
  -› Grenzen, Schnittstellen, Daten

- Benutzeroberfläche
- Funktionsebenen (horizontal)
- „Durchstich“ (vertikal)

- Aufgaben mit Prio und Aufwand

- Technologien, Frameworks, Pattern

Labor: Übungen an konkreten Beispielen


Inhaltsübersicht Software-Engineering I

4. Implementierung und Test

Codierrichtlinien und Codequalität,
Testarten und Testdurchführung,
Installation und Einführung

Schwerpunkt ist die Codequalität
⇒ keine Programmiervorlesung!

- Release-Planung
- Laufende Dokumentation
- Fortschritt und Reifegrad

- Codierrichtlinien, Frameworks, Pattern
- „Code Smells“, Refactoring

- Testgetriebene Entwicklung
- User-Tests, Testpläne, Testergebnisse

- Verteilung und Update

Labor: Testmanagement (z.B. UNIT-Test)

- Testarten:
  Blackbox/Whitebox,
  Paralleltest, Belastungstest
- Testabdeckung

- Versionsverwaltung,
- Konfigurationsmanagement

Inhaltsübersicht Software-Engineering I

6. Wartung und Pflege




- Dokumentation
- Schulung
- Bugfixes
- Feature-Requests
- Fehlermeldungen (Bugtracker)
- Releasenotes
- Support (1.Level, 2.Level)
- Refactoring
- Auslauf-Management (Ablösung, Redesign)

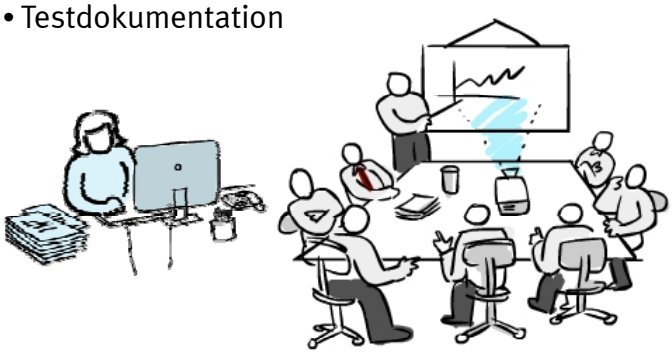
(... wird nicht im Detail behandelt)

Inhaltsübersicht Software-Engineering I

7. Dokumentation

Phasenspezifisch werden die verschiedenen Arten der Dokumentation behandelt:

- Benutzerhandbücher
- Programmdokumentation
- Testdokumentation



- Lastenheft, Pflichtenheft
- Architektur-Entwurf, -Dokumentation
- Testplan, Testergebnisse
- Releaseplan, Versionshistorie

- Printdokumente (Word, PDF)
- Präsentationen (PPT, PDF)
- ScreenCast (Video)
- Interaktive Schulung
- Bugtracker
- Wiki (strukturiert, Versionen)
- Blog (zeitliche Reihenfolge, Tags)
- FlipChart, Notizen (-› digitalisieren)

Inhaltsübersicht Software-Engineering I

Gruppenarbeit

Ein komplexes Problem wird als Projekt mit allen Phasen von den Studierenden erarbeitet und dokumentiert.

Rollenspiel:
„Anforderungserhebung“
„Domain-Modellierung“
„Architektur-Entwurf“
„Umsetzungs-Entwurf“
„Projekt-Management“
„Implementierung“
„Testmanagement“
„Dokumentation“

Wie sieht das bei den verschiedenen Vorgehensmodellen aus?
-› Wasserfall, Spiral, Prototyp, Agile

Inhaltsübersicht Software-Engineering I - Noten

9 ECTS Creditpoints

Die Vorlesung dauert **2 Semester** (3. und 4. Semester)
Es gibt nur **eine Note** über beide Semester

3. Semester:

Test (30 Min.)	1/3	Vorgehensmodelle
Projektarbeit	2/3	Anforderungsanalyse + Architekturentwurf

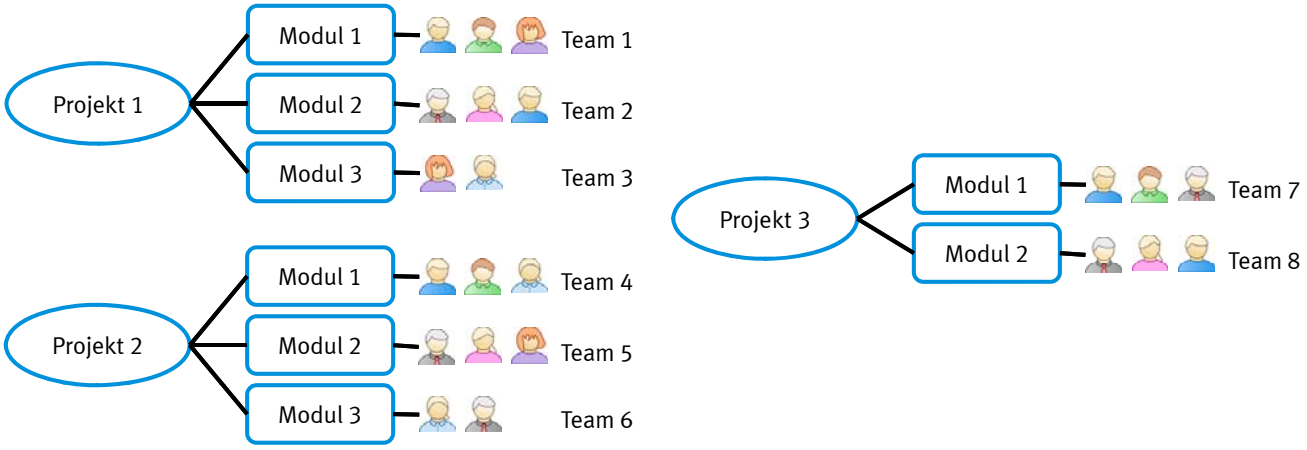
4. Semester:

Test (30 Min.)	1/3	Basistechniken (UML)
Projektarbeit	2/3	Architekturentwurf , Umsetzung +Testmanagement

Inhaltsübersicht Software-Engineering I - Gruppenarbeit

In jedem Semester werden **mehrere Projekte** bearbeitet und bewertet. Die Bearbeitung und Bewertung bricht sich dabei auf einzelne **Module** und **Teams** herunter. Es wird zwischen der **Gruppenleistung** und der **Teamleistung** bei der Bewertung unterschieden.

Die Bewertung erfolgt anhand den **Phasendokumente** wie Lasten-/Pflichtenheft, SW-Architektur unter Anwendung der entsprechenden **Methoden** und **Darstellungen**.



Exemplarische Ansicht!

Vorlesungsstruktur

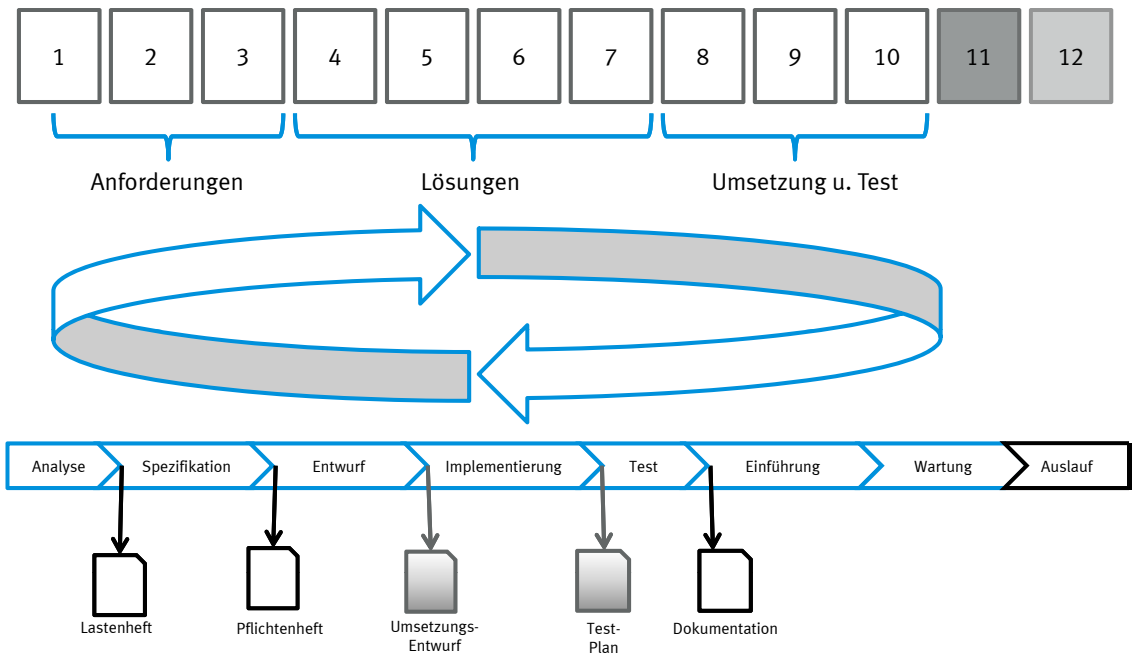
8:15 Uhr	**Vorlesung**	Wiederholung	15 Min.
9:45 Uhr		Neues Thema	60 Min.
10:00 Uhr	**Gruppenarbeit**	Gruppenarbeit	60 Min.
11:30 Uhr		Retrospektive	15 Min.

(30 Min. Puffer)

Script + Tafel + **eigener Aufschrieb**

+Vortrag

2x 11 Vorlesungen (3. und 4. Semester)

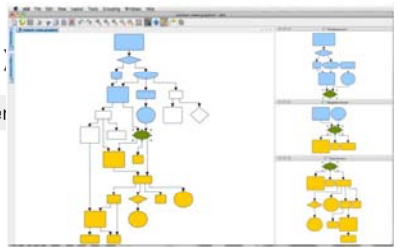


http://www.heise.de/software/download/yed_graph_editor/26948
http://www.yworks.com/en/products_yed_about.html
http://de.wikipedia.org/wiki/YEd

Softwaretools

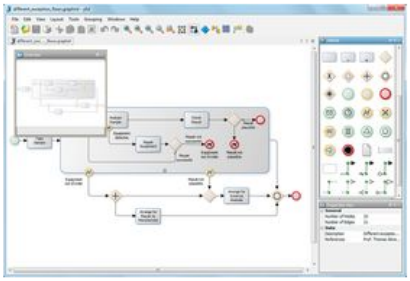
yEd Graph Editor

Bewertung: ★★★★★☆ (46 Bewertungen, gewichtet: 4,98 Punkte) ☐ vergleichen ☐ beobachten



Download

Der Java-Diagrammzeichner **yEd Graph Editor** erstellt Diagramme und Schemazeichnungen wie Ablauf- und Netzwerkpläne oder UML-Diagramme. Die fertigen Grafiken exportiert die Anwendung als Pixelbilder und in die gängigen Vektorformate EPS, PDF sowie den W3C-Standard SVG (Scalable Vector Graphics). Eine Stärke des Zeichenprogramms liegt in seinem Talent zum automatischen Layout etwa von Bäumen oder anderen hierarchischen Graphen; auch ungewöhnliche Anordnungen wie „organisch" oder „kreisförmig" gehören zum Repertoire. (pek)

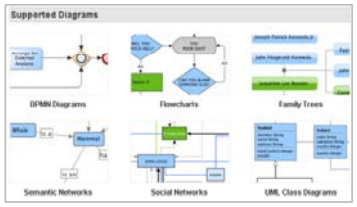


- Mind/Concept-Maps
- UML
- BPMN
- ERM
- ...

Name:
yEd Graph Editor

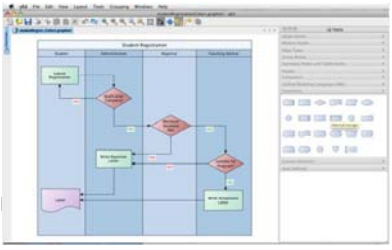
Sprache:
Deutsch/Englisch

Betriebssysteme:
Windows NT, 2000, XP, Server 2003, Server 2008, Vista, 7, Linux, Mac OS X, Mac OS X/Intel



Voraussetzungen:
2,0 GHz CPU, 1000 MByte RAM, 100 MByte HD

Benötigte Laufzeitumgebung:
Java



Download-Größe:
30 MByte

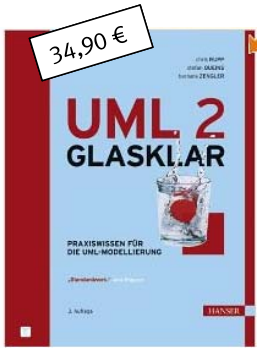
Preis:
kostenlos

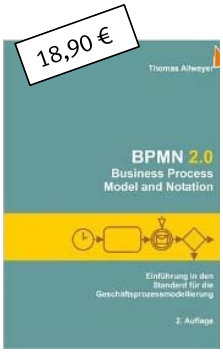
Einschränkungen:
keine

Hersteller/Autor:
Alle Programme von yWorks Gmbl

E-Mail-Adresse:
contact@yworks.com

Bücher





Links

- **Wikipedia** (http://de.wikipedia.org/, http://www.wikimindmap.org/) http://de.wikipedia.org/w/index.php?title=Software_Engineering
- **Heise-Developer** (http://www.heise.de/developer/)
- **Heise-Software** (http://www.heise.de/software/download/)

heise Developer

Podcast

SoftwareArchitekTOUR – Podcast für den professionellen Softwarearchitekten

Bekannte Softwareentwickler diskutieren über wichtige Softwarearchitekturthemen. Gehen Sie mit auf Tour und profitieren Sie von ihren Erfahrungen und Tipps. Wenn Sie keine Episode verpassen wollen, können Sie den Podcast abonnieren: RSS Feed


MichaelStal


Stefan Tilkov


Markus Völter

Episode 12: Systematischer Softwarearchitekturentwurf

Das in der Episode vorgestellte Modell für systematischen Architekturentwurf enthält einige Best Practices für das Aufsetzen einer Softwarearchitektur. Mehr...

Episode 24: Testing & Softwarearchitektur

Die Episode zum Thema "Testen" zeigt, dass der Softwarearchitekt sich dem Thema ausgiebig widmen sollte, da er als wichtiger Ansprechpartner zu Testentscheidungen fungiert. Mehr...

Fragen