

## Modulbeschreibung

# Software Engineering und Architekturen

**Allgemeine Informationen**

## Anzahl ECTS-Credits

3

## Modulkürzel

TSM\_SoftwEng

## Version

30. August 2009

## Modulverantwortliche/r

Martin Kropp, FHNW

## Sprache

	Lausanne	Bern	Zürich
Unterricht	<input type="checkbox"/> E <input checked="" type="checkbox"/> F	<input type="checkbox"/> D <input type="checkbox"/> E <input type="checkbox"/> F	<input checked="" type="checkbox"/> D <input type="checkbox"/> E
Unterlagen	<input checked="" type="checkbox"/> E <input type="checkbox"/> F	<input type="checkbox"/> D <input type="checkbox"/> E <input type="checkbox"/> F	<input type="checkbox"/> D <input checked="" type="checkbox"/> E
Prüfung	<input type="checkbox"/> E <input checked="" type="checkbox"/> F	<input type="checkbox"/> D <input type="checkbox"/> E <input type="checkbox"/> F	<input checked="" type="checkbox"/> D <input type="checkbox"/> E

## Modulkategorie

- Erweiterte theoretische Grundlagen
- Technisch-wissenschaftliche Vertiefung
- Kontextmodule

## Lektionen

- 2 Vorlesungslektionen und 1 Übungslektion pro Woche
- 2 Vorlesungslektionen pro Woche

## Kurzbeschreibung /Absicht und Inhalt des Moduls in einigen Sätzen erklären

Das Modul gibt einen vertieften Einblick in ausgewählte Themen des modernen Software Engineering. Die Themen stammen aus den folgenden Gebieten: Moderne Softwareentwicklungsprozesse, Softwarearchitektur und Prinzipien der Softwareevolution.

**Ziele, Inhalt und Methoden**

## Lernziele, zu erwerbende Kompetenzen

- Die Studierenden verstehen die Vor- und Nachteile der iterativen und inkrementellen Softwareentwicklung und wenden sie an.
- Die Studierenden können aus den zur Verfügung stehenden Methoden des modernen Software Engineering für ein Projekt eine geeignete auswählen, anpassen und anwenden.
- Die Studierenden kennen fortgeschrittene Architektur- und Designmuster und setzen diese ein, um Designentscheidungen zu fällen und über Softwaredesign zu reflektieren
- Die Studierenden lernen Software als ein sich kontinuierlich weiterentwickelndes und komplexes System kennen
- Die Studierenden wissen, wie sie bestehende Software verbessern, erweitern, integrieren und dabei das Qualitätsniveau hoch halten können

## Modulinhalt mit Gewichtung der Lehrinhalte

**Modernes Software Engineering**

- Agile Entwicklung
  - Wertschöpfung
  - Risikomanagement
  - Teamkultur
  - Kundenbeziehungen
- Mechanismen und Methoden
  - Effektive Kommunikation zwischen Stakeholdern
  - Projekt-Retrospektiven und Feedback Techniken
  - Qualitätsmanagement
  - Umgang mit Wandel
  - Requirements: laufende Erhebung und Management
  - Inkrementelle Planung

- Methoden der modernen Softwaretechnik
  - Übersicht und Vergleich verschiedener Ansätze
  - Z.B. XP; pragmatische Programmierung, Scrum
  - Implikationen für das Projektmanagement

### Software Architektur

- Die Rolle der Software Architektur und des Software Architekten
  - Was Software Architektur ist
  - Referenzmodelle und Referenzarchitekturen
  - Architekturstrukturen und -sichten
  - Dokumentation von Software Architekturen
  - Die Rolle des Software Architekten
- Fortgeschritten Design Konzepte
  - Interface-orientiertes Design
  - Kopplung und Kohäsion, command/query separation, don't talk to strangers
  - Abhängigkeitsmanagement / Package Design
  - Die SOLID Prinzipien
  - CRC --> IRI Cards
  - Design-By-Contract
- o Architektur Patterns
  - für Verteilte Architekturen
  - Architektur Patterns vs. Design Patterns
  - Pattern Stiele
- Auswahl, Erstellung und Bewertung von Softwarearchitekturen
  - Qualitätsmerkmale
  - Architekture Analyse

### Software Evolution

- Grundprinzipien der Software Evolution
  - Entwicklung, Wartung, Evolution
  - Software Alterung
  - Programmverstehen
  - Anwendungsbereich der Software Evolution
- Software Qualität & Analyse
  - Software Qualitätsmetriken
  - Software Visualisierung
  - Systematisches Debugging
  - Kontinuierliche Qualitätskontrolle
  - Konzepte der Architekturekontrolle
- Evolution von Legacy Code
  - „Re“-Technologien: Reverse Engineering, Re-Engineering, Re-Factoring
  - Objekt-orientiertes Re-Engineering
  - Effektiver Umgang mit Legacy Code
  - Testen von Legacy Systemen

### Lehr- und Lernmethoden

- 2 Vorlesungslektionen pro Woche plus 1 Übungslektion
- Selbststudium

### Voraussetzungen, Vorkenntnisse, Eingangskompetenzen

- Objektorientiertes Programmieren und Design in mehr als einer Programmiersprache
- Unified Process (UP)
- Unified Modeling Language (UML)
- *Design Patterns: Elements of Reusable Object-Oriented Software* (Gamma, Helm, Johnson, Vlissides; ISBN 0-201-63361-2)
- Grundlegende Architekturprinzipien/ -stile/-muster, Schichten, serviceorientiert, MVC, Pipes-and-Filters, z.B. aus *Pattern-Oriented Software Architecture Volume 1* (Buschmann, Meunier, Rohnert, Sommerlad, Stal; ISBN 0-471-95869-7)

- Konzepte für Versions- und Konfigurationsmanagement
- Konzepte für automatische Builds (z.B. ant, make, cruisecontrol)
- Konzepte für Unit Testing (z.B. junit oder nunit)

#### Bibliografie

1	Jim Highsmith: Agile Software Development Ecosystems Mary & Tom Poppendiek: Lean Software Development Kent Beck: eXtreme Programming 2nd Ed.
2	Ken Schwaber et al, Agile Software Development with Scrum, Prentice Hall, 2002
3	Alistair Cockburn: Agile Software Development Alistair Cockburn: Crystal Clear
4	Stephen Palmer, John Felsing: A Practical Guide to the Feature-Driven Development Kent Beck: eXtreme Programming explained, 2nd Ed.
5	Gernot Starke: Effektive Software Architekturen 2. Auflage
6	Ken Pugh, Interface Oriented Design
7	Doug Schmidt et.al.: Pattern-oriented Software Architecture, Vol. 2 Frank Buschmann et al: Pattern-oriented Software Architecture, Vol. 4
8	Robert C. Martin: Agile Software Development
9	Lehmann "Laws of Software Evolution Revisited"
10	Martin Fowler et al, Refactoring Joshua Kerievsky, Refactoring to Patterns
11	Gerard Meszaros: xUnit Test Patterns
12	Michael Feathers, Working Effectively with Legacy Code
13	Andreas Zeller: Why Programs Fail ISBN 1558608664
14	Len Bass, Paul Clements, Rick Kazman: Software Architecture in Practice 2 <sup>nd</sup> Ed.

#### Leistungsbewertung

##### Zulassungsbedingungen für die Modulschlussprüfung (Testatbedingungen)

keine

##### Schriftliche Modulschlussprüfung

Prüfungsdauer : 120 Minuten  
Erlaubte Hilfsmittel: Papierunterlagen nach Ansage