Module Description, available in: EN, FR

Software Engineering and Architectures

General Information

Number of ECTS Credits

3

Module code

TSM_SoftwEng

Valid for academic year

2021-22

Last modification

2019-11-24

Coordinator of the module

Martin Kropp (FHNW, martin.kropp@fhnw.ch)

Explanations regarding the language definitions for each location:

- Instruction is given in the language defined below for each location/each time the module is held.
- Documentation is available in the languages defined below. Where documents are in several languages, the percentage distribution is shown (100% = all the documentation).
- The examination is available 100% in the languages shown for each location/each time it is held.

	Lausanne			Lugano	Zurich		
Instruction		**X** F 100%			**X** E 100%		
Documentation			**X** E 100%		**X** E 100%		
Examination		**X** F 100%			**X** E 100%		

Module Category

TSM Technical scientific module

Profile Recommendations

This module is recommended for the following profiles:

	Lausanne	Zurich
Profile	CS	CS

Lessons

2 lecture periods and 1 tutorial period per week

Entry level competences

Prerequisites, previous knowledge

- Object-oriented programming and design in more than one programming language
- Unified Modeling Language (UML)
- Design Patterns: Elements of Reusable Object-Oriented Software (Gamma, Helm, Johnson, Vlissides; ISBN 0-201-63361-2)
- Version and configuration management concepts
- Unit testing concepts and practice
- Basic knowledge of Scrum

Brief course description of module objectives and content

The module provides an in-depth view of selected topics of modern software engineering. These stem from the fields: modern software development processes, software architecture, and the principles of evolution of software systems.

Aims, content, methods

Learning objectives and competencies to be acquired

- The student applies and understands benefits and liabilities of agile and lean development
- The student knows about advanced architectural and design patterns and uses them to drive and reflect on design decisions
- The student has the awareness of software as a continuously evolving and complex system
- The student knows and can select maintenance and evolution techniques for continuous development of evolving and extension of legacy software systems while maintaining its quality

Module content with weighting of different components

Modern Software Engineering

- Agile Development
  - value creation and innovation
  - risk and complexity in software development
  - agile competence pyramid
- Principles and Practices
  - effective communication among stakeholders
  - project retrospectives, feedback techniques
  - ongoing requirements, solicitation and management
- Modern Software Engineering Methodologies
  - benefits/difficulties and comparison of different approaches (i.e. plan-driven, hybrid and agile)
  - implications for project management
  - incremental planning
- Research in Agile Development

Software Architectures

- Role of software architecture and software architects
  - reference models, reference architectures
  - architectural structures and views
  - software architecture documentation
- Advanced design concepts
  - the SOLID principles
  - Attribute Driven Design
  - design by contract
- Architectural patterns
  - for distributed architectures
  - architecture patterns vs design patterns
  - pattern styles
- Selection, creation, and evaluation of software architectures
  - quality attributes
  - architecture analysis methods
- Research in Software Architecture

Software Evolution

- Principles of Software Evolution
  - development, maintenance, evolution
  - software aging
  - Program comprehension
- Software Quality & Analysis
  - software quality metrics
  - software visualization
  - continuous quality control
- Evolution of Legacy Code
  - “Re”-Techniques: Reverse Engineering, Re-Engineering, Re-Factoring
  - object-oriented re-engineering
  - Testing legacy systems
- Research in Software Evolution

Teaching and learning methods

Self-study, homework assignments

Literature

1. Mary & Tom Poppendiek: Lean Software Development

2. Kent Beck: eXtreme Programming Explained 2nd Ed.
3. Ken Schwaber et al, Agile Software Development with Scrum, Prentice Hall, 2002
4. Alistair Cockburn: Agile Software Development
5. Robert C. Martin: Agile Software Development
6. Tom Mens: Software Evolution
7. Doug Schmidt et.al.: Pattern-oriented Software Architecture, Vol. 2
   Frank Buschmann et al: Pattern-oriented Software Architecture, Vol. 4
8. Len Bass, Paul Clements, Rick Kazman: Software Architecture in Practice 2nd Ed.
9. Gernot Starke: Effektive Software Architekturen 2. Auflage
10. Lehmann "Laws of Software Evolution Revisited"
11. Martin Fowler et al, Refactoring
    Joshua Kerievsky, Refactoring to Patterns
12. Michael Feathers, Working Effectively with Legacy Code
13. Andreas Zeller: Why Programs Fail ISBN 1558608664

Assessment

Certification requirements

Module uses certification requirements

Certification requirements for final examinations (conditions for attestation)

Assignments are completed in time and passed. Assignments can be oral presentations and written.

Basic principle for exams

As a rule, all standard final exams are conducted in written form. For resit exams, lecturers will communicate the exam format (written/oral) together with the exam schedule.

Standard final exam for a module and written resit exam

Kind of exam

Written exam

Duration of exam

120 minutes

Permissible aids

Aids permitted as specified below:

Permissible electronic aids

No electronic aids permitted

Other permissible aids

Any paper documentation

Special case: Resit exam as oral exam

Kind of exam

Oral exam

Duration of exam

30 minutes

Permissible aids

No aids permitted

Description du module, disponible en: EN, FR

Génie logiciel: méthodes de développement et architecture

Informations générales

Nombre de crédits ECTS

3

Code du module

TSM_SoftwEng

Valable pour l'année académique

2021-22

Dernière modification

2019-11-24

Coordinateur/coordinatrice du module

Martin Kropp (FHNW, martin.kropp@fhnw.ch)

Explications concernant les langues d'enseignement par site :

- Les cours se dérouleront dans la langue définie ci-dessous par lieu/exécution.
- Les documents sont disponibles dans les langues définies ci-dessous. Pour le multilinguisme, voir la répartition en pourcentage (100% = documents complets)
- L'examen est disponible à 100% dans chaque langue sélectionnée pour chaque lieu/exécution.

	Lausanne			Lugano	Zurich		
Leçons		**X** F 100%			**X** E 100%		
Documentation			**X** E 100%		**X** E 100%		
Examen		**X** F 100%			**X** E 100%		

Catégorie de module

TSM approfondissement technico-scientifique

Recommandations de profil

Ce module est recommandé pour les profils suivants :

	Lausanne	Zurich
Profil	CS	CS

Leçons

2 leçons et 1 leçon de pratique par semaine

Compétences préalables

Connaissances préalables, compétences initiales

- Conception et programmation orientées selon l'objet dans plus d'un langage de programmation
- Unified Modeling (UML)
- Design Patterns: Elements of Reusable Object-Oriented Software (Gamma, Helm, Johnson, Vlissides; ISBN 0-201-63361-2)
- Notions de gestion des versions et des configurations
- Notions de tests unitaires (par exemple junit ou nunit)
- Connaissance de base de Scrum

Brève description du contenu et des objectifs

Le module fournit une étude approfondie des sujets sélectionnés dans le domaine du génie logiciel. Ces sujets sont issus des domaines suivants: méthodes de spécification, méthodes de développement agiles, architecture logicielle, évolution du logiciel.

Objectifs, contenus, méthodes

Objectifs d'apprentissage, compétences à acquérir

- L'étudiant comprend et peut expliquer les avantages d'une méthodologie de développement itérative et incrémentale.
- En fonction des caractéristiques d'un projet donné, l'étudiant est en mesure de sélectionner, d'adapter et d'appliquer la méthodologie de développement appropriée.
- L'étudiant connaît des principes de conception et d'architecture avancés et peut les utiliser pour diriger la phase de conception d'un système.
- L'étudiant est conscient qu'un logiciel est un système complexe en évolution constante.
- L'étudiant sait comment améliorer, étendre et intégrer des logiciels existants, tout en maintenant leur qualité.

Contenu des modules avec pondération du contenu des cours

Génie Logiciel

- Principes et valeurs des méthodes de développement agiles
  - Création de valeur
  - Gestion du risque
  - Culture d'équipe
  - Relations clients
- Bonnes pratiques et techniques de gestion
  - Gestion de la communication entre parties prenantes
  - Revues de projets
  - Assurance qualité
  - Gestion des changements
  - Planification itérative et incrémentale
- Méthodologies de développement modernes
  - Vue d'ensemble et comparaison de différentes approches
  - Unified Process (UP), eXtreme Programming (XP), Scrum
  - Implications sur la gestion de projet

Architecture logicielle

- Le role de l'architecture est des architectes logiciels
  - notion d'architecture
  - modèles de référence, architectures de référence
  - structures et vues architecturales
  - architecture et documentation
  - les roles de l'architecte logiciel
- Principes de conception avancés
  - interface-oriented design
  - couplage et cohésion
  - gestion des dépendances
  - les principes SOLID
  - conception par contrat
- Modèles de conception et d'architecture
  - styles d'architectures
  - modèles pour architectures distribuées
- Sélection, création et évaluation d'architectures logicielles
  - qualités systémiques
  - méthodes d'analyse

Evolution du logiciel

- Principes d'évolution
  - Développement, maintenance, évolution
  - "Software aging"
  - Compréhension du code
- Analyse et qualité du logiciel
  - métriques
  - techniques d'analyse et de visualisation
  - contrôle de qualité continu
- Evolution de code hérité
  - Re-Technologies: Reverse Engineering, Re-Engineering, Re-Factoring
  - Re-engineering orienté objet
  - Travailler efficacement avec du code hérité
  - Tester des systèmes hérités

Méthodes d'enseignement et d'apprentissage

travail personnel, lectures et exercices

Bibliographie

1. Mary & Tom Poppendiek: Lean Software Development
   Kent Beck: eXtreme Programming Explained 2nd Ed.
2. Ken Schwaber et al, Agile Software Development with Scrum, Prentice Hall, 2002
3. Alistair Cockburn: Agile Software Development
4. Robert C. Martin: Agile Software Development
5. Tom Mens: Software Evolution
6. Doug Schmidt et.al.: Pattern-oriented Software Architecture, Vol. 2
   Frank Buschmann et al: Pattern-oriented Software Architecture, Vol. 4
7. Len Bass, Paul Clements, Rick Kazman: Software Architecture in Practice 2nd Ed.
8. Gernot Starke: Effektive Software Architekturen 2. Auflage
9. Lehmann "Laws of Software Evolution Revisited"
10. Martin Fowler et al, Refactoring
    Joshua Kerievsky, Refactoring to Patterns
11. Michael Feathers, Working Effectively with Legacy Code
12. Andreas Zeller: Why Programs Fail ISBN 1558608664

Evaluation

Conditions d'admission

Le module utilise les conditions d'admission

Conditions d'admission à l'examen de fin de module (exigences du certificat)

Les devoirs ont été réalisés dans les délais et jugés satisfaisants. Les devoirs peuvent être évalués sous forme orale et/ou écrite.

Principe pour les examens

En règle générale, tous les examens réguliers de fin de module se déroulent sous forme écrite. Concernant les examens de répétition, leur format (écrit ou oral) sera communiqué par l'enseignant-e en même temps que le calendrier des examens.

Examen de fin de module régulier et examen écrit de répétition

Type de l'examen

Examen écrit

Durée de l'examen

120 minutes

Aides autorisées

Les aides suivantes sont autorisées:

Aides électroniques autorisées

Aucune aide électronique autorisée

Autres aides autorisées

Documentation papier

Cas spécial: examen de répétition oral

Type de l'examen

Examen oral

Durée de l'examen

30 minutes

Aides autorisées

Sans aides