# SWEN2 - Prüfungsstoff (ab FS25)

**Hinweis: Der Inhalt der Prüfung kann bis Ende Semester noch angepasst werden.**

## Hinweise zur Prüfung

- Es wird ein eAssessment (schriftlich) mit Moodle durchgeführt
- Die Prüfung darf nur im Prüfungszimmer geöffnet werden (nicht remote)
- Die Gewichtung der Vornote (Gruppenarbeit / Vortrag) beträgt 20 % und die des eAssessments 80 %
- Es wird eine Legi-Kontrolle durchgeführt
- Prüfungsdauer: 30 Minuten

### Hilfsmittel

- Ein eigener Laptop für die Durchführung der Prüfung ist erlaubt.
- Zusätzlich ist ein Tablet mit SWEN2-Folien und eigenen Notizen erlaubt.
- Andere elektronische Geräte sind *nicht* erlaubt.
- Als Unterlagen sind nur die Slides der SWEN2-Vorlesung, eigene Lösungen der Aufgaben des Selbststudiums sowie eigene Notizen erlaubt.
- Keine Kommunikation erlaubt.
- Suchmaschinen und KI wie ChatGPT etc. dürfen *nicht* benutzt werden!

### Weitere Hinweise

- Überwiegend Multiple-Choice-Fragen (K-prim), wenige Fragen mit Kurzantworten.
- Fragen möglich, bei denen keine oder alle Antworten richtig sind
- Achtung, die Zeit ist knapp

## Prüfungsstoff

**Prüfungsstoff sind Vorlesungs- und Selbststudiumsinhalte sowie der Fachvortrag.**

*Es wird vorausgesetzt, dass sie den Stoff des Selbststudiums gelesen und die dazugehörigen Aufgaben gelöst haben. Sie sollten in der Lage sein, den Stoff des Selbststudiums in groben Zügen zu erklären und einzuordnen.*

### Buch von Kent Beck

Sie haben das Buch *Tidy First?* gelesen und können die wichtigsten Ideen und Konzepte erklären:

*Tidy First? A Personal Exercise in Empirical Software Design, Kent Beck, ISBN: 978-1-098-15124-9*

**Einführung Softwareentwicklung**

Sie kennen SWEBOK. Sie können Vor-/Nachteile der plangetriebenen Methoden aufzählen. Sie wissen, für welche Arten von Projekten plangetriebene Methoden verwendet werden sollten und für welche nicht.

**Gesetze der Softwareentwicklung**

Sie können die drei «Gesetze der Softwareentwicklung» erklären. Ebenfalls können Sie weitere "Gesetze" wie «Conway's Law», «Brooks's Law», «Parkinson's Law», «Pareto's Fallacy», «Sturgeon's Revelation», «The Iceberg Fallacy», «The Peter Principle», «Eagleson's Law», sowie «Greenspun's 10th Rule of Programming», erklären. (siehe Selbstudium/Unterlagen Woche 1)

**Agile Manifesto**

Sie können die «Values» des «Agile Manifesto» aufzählen und deren Bedeutung erklären. Sie kennen auch die «12 Principles» hinter dem «Agile Manifesto» und können diese erklären.

Sie können die Bedeutung des agilen Manifesto für die Softwareentwicklung einordnen.

**Agile**

Sie kennen die folgenden Begriffe und können sie erklären: Risk, Cost of change und die Variablen der Softwareentwicklung (Iron Triangle) und deren Kräfte.

**eXtreme Programming**

Sie können erklären, was eXtreme Programming ist. Sie kennen die folgenden Begriffe und können sie erklären: Core Values, Principles, Practices, Cost of Change, Business Value.

Sie kennen folgende XP Practices und können sie erklären: The Planning Game, Small Releases, Metaphor, Simple Design, Unit-Testing, Refactoring, Pair-Programming, Collective Code Ownership, Continuous Integration, 40 Hours Week, On-site Customer, Coding Standards, Test-Driven Programming (TDD), Slack, Spike, Incremental Design, Self-Organized Team.

**Software Craft**

Sie kennen die Gründe, warum das Software Craft Manifest notwendig wurde. Sie kennen die Formate der Software Craft Community, wie Austausch entsteht und welche Prinzipien dahinter stehen. Sie können erklären, wie man als Software Entwickler:in üben kann. Sie können Coding Dojo und Code Katas genauer erklären.

### Pyramid of Agile Competencies

Sie kennen die «Pyramid of Agile Competencies» und können die drei Ebenen erklären.

### User Stories

Sie beschreiben die Ziele, die Anwendung und wie das Format einer Userstory aufgebaut ist. Sie kennen den Zusammenhang zwischen Epics, Themes und Stories. Sie kennen die Merkmale einer guten Userstory.

### Estimation and Planning

Sie kennen die folgenden Begriffe und können sie erklären: User Stories, User Roles , Epics, Themes, Story Points, Velocity, Planning Poker, Conditions of Satisfaction, Levels of Planning, Product Backlog, Priorisierung (der User Stories), Techniques for Estimating, relative Schätzung.

Sie kennen die folgenden Begriffe und können diese erklären: Planning for value, Cost, Financial value, Risk, New Knowledge, Kano-Model of Customer Satisfaction.

Sie kennen die Planungsebenen im agilen Kontext und können die Zuständigkeiten beschreiben.

### DevOps (inkl. Build Automation, CI, CD)

Sie kennen die Geschichte von DevOps. Sie kennen mindestens eine Definition von DevOps. Sie können erklären, wie DevOps funktioniert (DevOps lifecycle). Sie können erklären, wie Agile und DevOps zusammenhängen resp. wie sie sich ergänzen.

Sie können die vier «DORA Metrics» erklären. Sie kennen den DORA Report und können die wichtigsten Aussagen nennen.

Sie können die Arten der Software Automation (on-demand, scheduled, triggered) sowie deren Ziele erklären. Sie kennen die Software Automation Pipeline mit den einzelnen Schritten und deren Unterschiede. Sie können die «Key Practices» von Continuous Integration (CI) erklären (inkl. allen Aufgaben des Selbststudiums!)

Sie können die Begriffe Virtualization, Containerization sowie Infrastructure as Code (IaC) erklären. Sie können erklären, wieso eine Mircroservice-Architektur gut zur Agilen Software-Entwicklung passt. Sie kennen GitHub Actions.

### Scrum

Sie kennen die Charakteristiken von SCRUM inkl. allen «Roles resp. Scrum team», «Ceremonies resp. Scrum Events», «Scrum Artifacts», sowie «Scrum

Values» und können diese erklären. Sie kennen die folgenden Begriffe und können diese erklären: Sprint, Sprint Goal, Retrospective, Task Board, Burndown Chart, Definition of Done, Definition of Ready, Daily Scrum, Increment.

Sie haben den SCRUM Guide gelesen: Scrum Guide

**Architektur Patterns (Gruppenpuzzle)**

Sie kennen die folgenden Patterns und können sie erklären: CQRS (Command Query Responsibility Segregation), Event Sourcing, Strangler Pattern, Online-Migration, Circuit Breaker, Bulkhead (Schottwand), Retry, Serverless, Microservices, Self-contained Systems, Monolith (Modulith).

**Software Architecture**

Sie kennen die folgenden Begriffe und können sie erklären: Software-Architektur, Software-Architekt, Unterschied zwischen Enterprise Architecture und Application Architecture, Schwierigkeiten des Software Designs (Complexity, Conformity, Changeability, Invisibility).

Sie können die beiden *Architectural Drivers* erklären.

Sie können die Begriffe *Standard, Style, Pattern* erklären. Sie können den Unterschied zwischen *Layer* und *Tier* erklären.

Architekturstile: Sie kennen die Vor- und Nachteile aller behandelten Architekturstile und können diese erklären. Sie kennen die drei großen Patterns (Transaction Script, Domain Model und Table Module) und deren Einsatzgebiete und verstehen die 4 Ebenen des C4-Modells.

Sie kennen die Vor- und Nachteile des *Architecture Canvas* und wofür es eingesetzt werden kann.

Hinweis: Sie müssen die Bücher «Software Architecture for Developers: Volume 1 & 2» nicht im Detail gelesen haben. Das C4-Modell wird vorausgesetzt.

**Cynefin Framework / Codefin**

Sie haben das Cynefin Video auf Moodle gesehen und können den Nutzen von Cynefin im Kontext der Softwareentwicklung erklären. Sie kennen die folgenden Begriffe und können sie erklären: Cynefin, Framework, Exaptation, 4 + 1 Domains (Simple, Complicated, Complex, Chaotic, Disorder), Causality, Correlation, Constraint.

Sie haben auch das Codefin Video auf Moodle gesehen und können die praktische Anwendung in der Softwareentwicklung erklären. (siehe Selbststudium)

**Kanban / Lean Software Development**

Sie wissen, wie Agile, Lean Software Development, Scrum, XP und Kanban zusammenhängen und sie kennen deren Herkunft. Sie kennen die Principles, Practices und Values von Kanban. Sie können erklären, wie ein Kanban Board und das CFD funktionieren. Sie kennen die folgenden Begriffe und können diese erklären: Pull System, Limited WIP, «Visualize the Workflow» (Kanban Board), Cumulative Flow Diagramm, Lead time resp. Cycle time, «Kaizen», Waste in Software Development. Sie kennen die «Principles of Lean» und können diese erklären. Sie kennen in Grundzügen die Unterschiede und Gemeinsamkeiten von Scrum und Kanban.