Zurich University of Applied Sciences

School of Engineering

Bachelor of Science (BSc) in Informatik

Modul Advanced Software Engineering 1 (ASE1)

Software Architektur

Einführung

Institut für Angewandte Informationstechnologie (InIT)

Walter Eich (eicw) / Matthias Bachmann (bacn)

https://www.zhaw.ch/de/engineering/institute-zentren/init/

Agenda

- Software Architektur als Teil von Software Engineering
- ISAQB – Intl SW Architecture Qualification Board
- CPSA-F Certified Professional for Software Architecure
- Voraussetzung

Teilgebiete des Software Engineering (SWEBOK V3) – Stand 2013 Review

Zurich University of Applied Sciences
School of Engineering

KA01 - Requirements	KA02 - Design	KA03 - Construction	KA04 - Testing	KA05 Maintenance
KA06 – Software Configuration Management				
KA07 – Software Engineering Management				
KA08 – Software Engineering Process				
KA09 – **Software Engineering Model and Methods**				
KA10 – Software Quality				
KA11 – **Software Engineering Professional Practice**				
Educational Requirements for Software Engineering				
KA 12 – Software Engineering Economics Foundations				
KA 13 – Computing Foundations				
KA 14 – Mathematical Foundations				
KA 15 - Engineering Foundations				

2. Key Issues in Software Design

2.1. Concurrency
2.2. Control and Handling of Events
2.3. Data Persistence
2.4. Distribution of Components
2.5. Error and Exception Handling and Fault Tolerance
2.6. Interaction and Presentation
2.7. Security

3. Software Structure and Architecture

3.1. Architectural Structures and Viewpoints
3.2. Architectural Styles
3.3. Design Patterns
3.4. Architecture Design Decisions
3.5. Families of Programs and Frameworks

4. User interface design

5. Software Design Quality Analysis and Evaluation

5.1. Quality Attributes

5.2. Quality Analysis and Evaluation Techniques

5.3. Measures

6. Software Design Notations

6.1. Structural Descriptions (Static View)

6.2. Behavioral Descriptions (Dynamic View)

7. Software Design Strategies and Methods

7.1. General Strategies

7.2. Function-Oriented (Structured) Design

7.3. Object-Oriented Design

7.4. Data-Structure-Centered Design

7.5. Component-Based Design (CBD)

7.6. Other Methods

8. Software Design Tools

Parallelen zur klassischen Architektur

- Gute Architektur:
  - **utilitas (Nützlichkeit):** Die Software erfüllt die funktionalen und nichtfunktionalen Anforderungen der Nutzer und Kunden.
  - **firmitas (Festigkeit):** Die Software ist stabil im Hinblick auf die geforderten Qualitätseigenschaften z.B. langlebig, da zukünftige Weiterentwicklungen möglich sind, ohne das System komplett neu bauen zu müssen.
  - **venustas (Schönheit):** Die Software ist sowohl aussen (gegenüber dem Nutzer) wohl strukturiert, sodass sie intuitiv nutzbar ist, als auch innen (gegenüber demjenigen, der die Software pflegen und weiter-entwickeln soll) wohl strukturiert, sodass dieser die internen Strukturen der Software leicht verstehen und damit gut seinen Aufgaben nachkommen kann.

ISAQB – Intl SW Architecture Qualification Board

Zurich University of Applied Sciences
zh aw School of Engineering

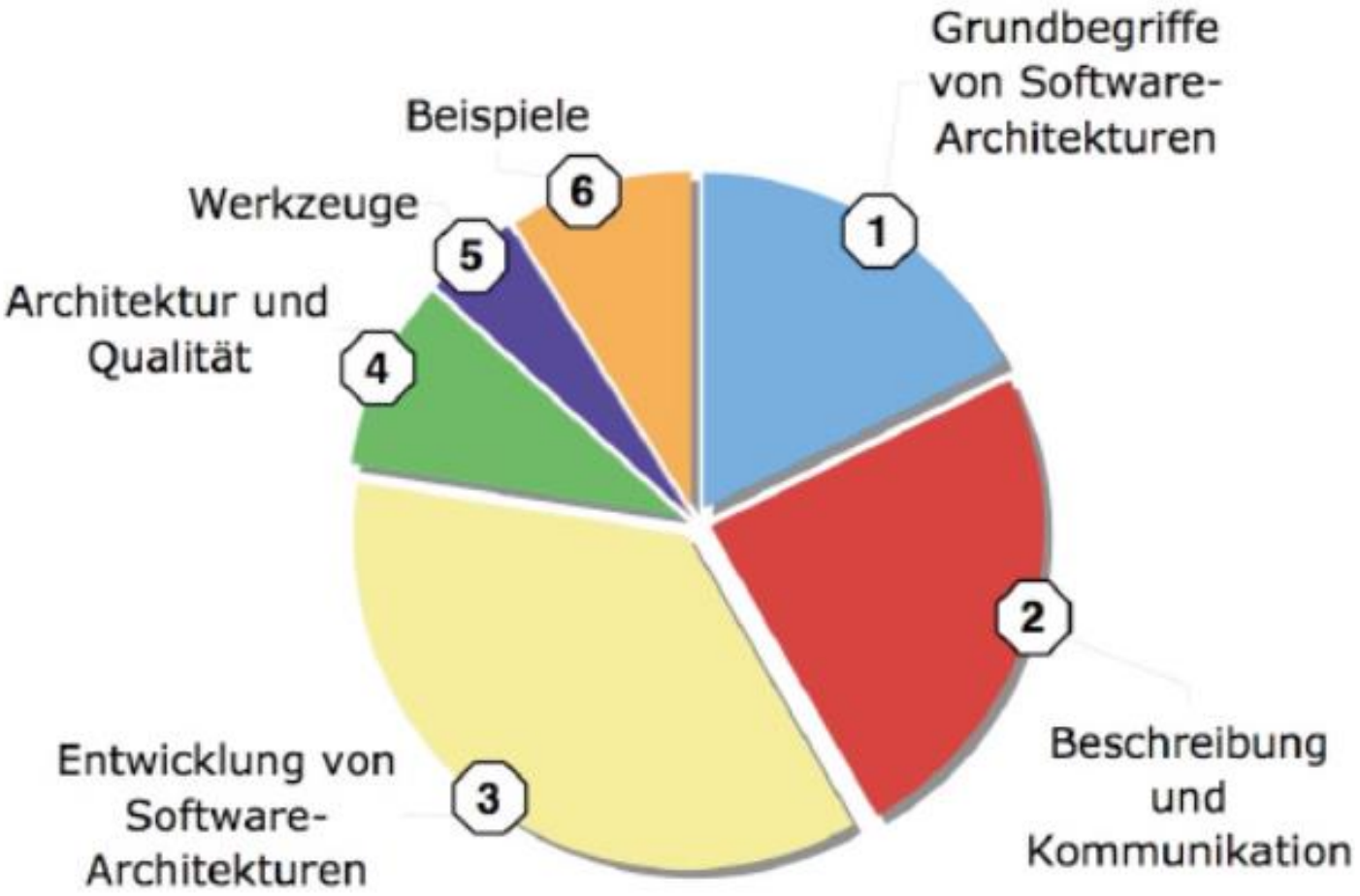
- **Ziele des Vereins** http://www.isaqb.org/
  - Das International Software Architecture Qualification Board ist ein Zusammenschluss von Fachexperten zu Softwarearchitektur aus Industrie, Beratungs- und Trainingsunternehmen, Wissenschaft und anderen Organisationen
  - Der iSAQB e.V. ist ein internationales Gremium (organisiert gemäß deutschem Vereinsrecht), mit den folgenden Zielen:
  - Erstellung und Pflege einheitlicher Lehr- und Ausbildungspläne für Softwarearchitekten (Certified Professional for Software Architecture)
  - Definition von Zertifizierungsprüfungen auf Basis der CPSA-Lehrpläne
  - Sicherstellung der fachlich-inhaltlichen Qualität von Lehre, Aus- und Weiterbildung für Softwarearchitektur

CPSA: Certified Professional für Software Architecture

Zurich University of Applied Sciences
zh aw School of Engineering

- ISAQB: International Software Architecture Qualification Board
- Dokumente CPSA-F:
  https://www.isaqb.org/documents/
- Kompetenzen:
  - mit anderen Projektbeteiligten aus den Bereichen Anforderungsmanagement, Projektmanagement, Test und Entwicklung wesentliche Softwarearchitekturentscheidungen abzustimmen
  - Softwarearchitekturen auf Basis von Sichten, Architekturmustern und technischen Konzepten zu dokumentieren und kommunizieren
  - die wesentlichen Schritte beim Entwurf von Softwarearchitekturen zu verstehen und für kleine und mittlere Systeme selbstständig durchzuführen
  - Bestehende Softwarearchitekturen zu analysieren und zu bewerten

Gliederung des Lehrplaninhalts



Voraussetzungen

- typischerweise mehrjährige praktische Erfahrung in der Softwareentwicklung, erworben anhand unterschiedlicher Projekte oder Systeme,
- vertiefte Kenntnisse und praktische Erfahrung mit mindestens einer höheren Programmiersprache,
- Grundlagen der Modellierung sowie von UML, insbesondere der Klassen-, Paket-, Komponenten- und Sequenzdiagramme sowie deren Abbildung auf Quellcode,
- praktische Erfahrung mit verteilten Systemen, d.h. mit der Entwicklung eines größeren Systems, das nicht nur auf einem Rechner ausgeführt wird.
- Erfahrung mit der technischen Dokumentation von Programmen, Entwürfen und technischen Konzepten.

Strukturierung

- Gemäss Buch «Basiswissen für Software Architekten»
  - (1) Einleitung
  - (2) Grundlagen von Architekturen
  - (3) Kommunikation von Architekturen
  - (4) Entwurf von Architekturen (Prinzipien, Architekturen (Rep Arch-Pattern, Ref Architekturen)
  - (5) Softwarearchitekturen und Qualität
  - (6) Werkzeuge
- Technikteil
  - Spring Framework Dependency Injection
  - Aspektorientierte Programmierung
  - Spring Data (JDBC, Hibernate, Repository)
  - Ev. Teil von Spring Boot

Links

- http://www.arc42.de/
- http://www.iso-architecture.org/ieee-1471/ (**ISO/IEC/IEEE 42010:2011**)
- http://msdn.microsoft.com/en-us/library/bb466232.aspx
- http://iacis.org/iis/2006/Urbaczewki_Mrdalj.pdf
- Beispiel DokChess: http://dokchess.de/
- Beispiel aus dem Buch von Gernot Starke: https://dl.dropbox.com/u/45486/esabuch-4-website/ESA4-Auszug-Kapitel12-Beispiele.pdf