Aufgabe 4

In welchen Phasen eines Entwicklungsprozesses ist ein Software-Architekt tätig?

im V-Modell: Technischer Systementwurf (Architektur) und Komponenten Spezifikation (Detailliertes Design erstellen)

Welche Aufgaben hat ein Software-Architekt?

- Analyse
- Entwurf
- Implementierung
- Test (einschließlich Integration, synonym: Validierung)
- Deployment (Installation, Schulung)
- Evolution (vor allem Wartung)
- weitere (Versionsmanagement, Reviews, ...)

Was ist der Input, was der Output seiner Tätigkeit in Bezug auf Dokumente

Architektur entwerfen:

Input:

- Organisatorische Einflussfaktoren
- Technische Einflussfaktoren
- Software Anforderungen

Output:

- Architektur-Dokument

Architektur Prüfen:

Input

- Metriken/Review/Prototyp / Heuristiken

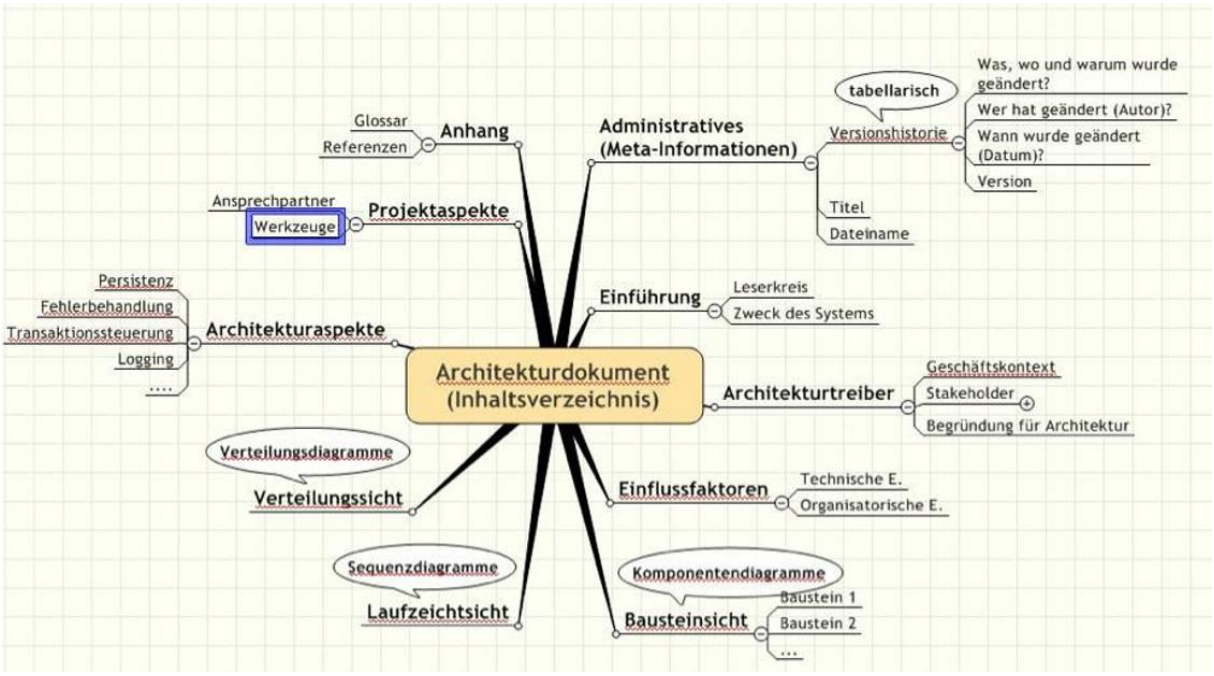
Was ist das Ziel der Software-Architektur?

- Das System in Komponenten zerlegen
- Bauplan / Arbeitsanweisung für Entwickler erstellen
- Eine Lösung konzipieren, die die Software-Anforderungen und damit auch die Kriterien der ISO 25010 erfüllt
- Eine Lösung in „Time and Budget“ konzipieren

Was sollte eine Software-Architektur beschreiben?

- Beschreibung der Grobarchitektur
    - Komponentendiagramm (z.B. UML) logische Sicht
    - Aktivitätsdiagramm / Sequenzdiagramm
    - Verteilungsdiagramm
- Beschreibung der Feinarchitektur
    - Komponenten- oder Klassendiagramm
- Beschreibung der Technologien
    - Programmiersprache
    - Frameworks
- Cross Cutting Concerns (Querschnittsaspekte)
    - I18N Internationalisierung (Coding, Ressource Dateien, Datum, Währung)
    - Realisierung der Barrierefreiheit
    - Fehlerbehandlung
    - Transaktion
- Werkzeuge
- Integrationsstrategie

Welche Inhalte sollte ein Architekturdokument enthalten?



Woraufhin sollte man dieses Dokument prüfen? Wie kann man das machen?

Prüfen durch Dritte:

- Vollständigkeit
- Verständlichkeit (Sprache, Diagramme)
- Traceability Matrix (meist Tabelle, ob alle Anforderungen vom SRS erfüllt sind)

Heuristiken

- Abstraktion (Finden von Gemeinsamkeiten Vereinfachung durch Generalisierung)
- Kapselung (Wenige Abhängigkeiten, Infos und Funktionalitäten sind „versteckt“)
- Hierarchie (Vererbungshierarchie, Zerlegung des Systems vom Groben ins Feine)
- Konzeptuelle (Integrität durchgängige Anwendung von Entwurfsentscheidungen)
- Modularität (Zerlegung Komponentenbildung)

Heuristiken aus dem Projektumfeld	■ Hinterfragen Sie Anforderungen! ■ Betrachten Sie Use Cases! ■ Nutzen Sie andere Systeme als Inspiration! ■ Suchen Sie Feedback!
Heuristiken für den Entwurf	■ So einfach wie möglich! ■ Bleiben Sie auf der richtigen Detailebene! ■ Modellieren Sie angemessen komplex! ■ Denken Sie in Verantwortlichkeiten! ■ Nicht übergeneralisieren! ■ Konzentrieren Sie sich auf die Schnittstellen! ■ Entwerfen Sie so lokal wie möglich!
Heuristiken zur Beherrschung von Komplexität	■ Teile und herrsche! ■ Verstecken Sie Details! ■ Kapseln Sie Risiken! ■ Trennen Sie Funktionalität und Kontrolle! ■ Trennen Sie Fachlogik und Infrastruktur!
Heuristiken zur Arbeitsmethodik	■ Beginnen Sie mit den schwierigsten Teilen! ■ Verwenden Sie Prototypen! ■ Werfen Sie Prototypen weg! ■ Dokumentieren Sie Entscheidungen!

Was zeichnet ein gutes Dokument aus?

- Vollständig
- Verständlich
- Widerspruchsfrei
- gesetzeskonform

Was eine gute Architektur?

- Ist modular / komponentenorientiert

  (modular = Testbar, Änderbar, Austauschbar, Wiederverwendbar, Planbar)
- Weak coupling -> Lose Kopplung der Komponenten
- Strong cohesion -> Starker Zusammenhang innerhalb der Komponenten
- Erfüllt die Anforderungen -> SRS / ISO 25001
- Lässt sich in Time & Budget realisieren

Welche Einflussfaktoren sollte ein Architekt beachten?

- Stakeholder-Anforderungen
- System-Anforderungen
- Finanzelle Aspekte
- Erfahrung
- Verfügbare Technik

- Bestehende Umsetzung
- Soup -> Verfügbare Bibliotheken, Komponenten und Frameworks
- Technische Einflussfaktoren
    - Beschränkungen, die zusätzlich zu denen in der SRS zu beachten sind
    - Hardware
    - Entwicklungswerkzeuge
    - Bestehende Systemlandschaft
    - Bestehende Architektur (Architekturstile)
- Organisatorische Einflussfaktoren
    - Budget
    - Zeit
    - Anzahl und Kenntnisstand der Entwickler
    - Make or Buy
    - Unterbeauftragung möglich
    - Risikobereitschaft des Auftraggebers
    - Haftung, gesetzliche Anforderungen

Wie können sich diese und die Systemanforderungen beeinflussen? Nennen Sie Beispiele

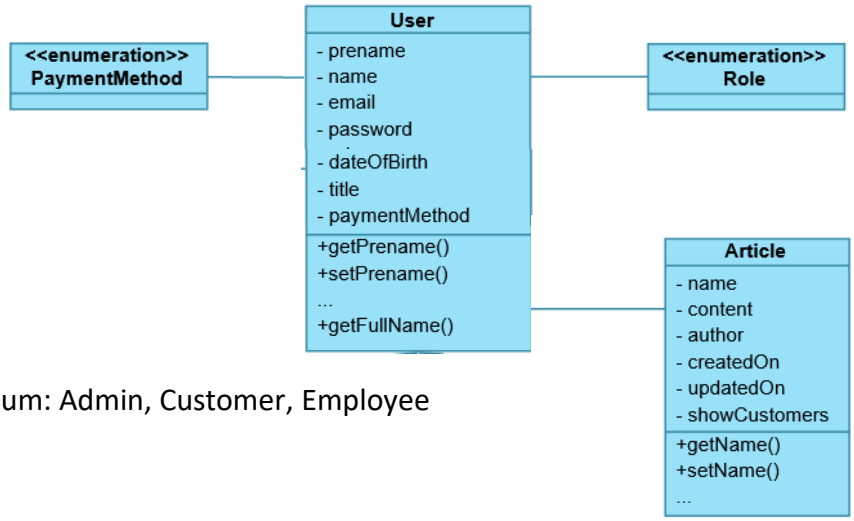
- wenn wenig Geld zur Verfügung steht -> hat nur wenig Entwickler -> dauert lange -> kann mit Zeit in Konflikt stehen, wenn Abgabetermin besteht und man merkt es reicht nicht
- neue Technologien -> größeres Risiko -> braucht neue Lizenzen & Ausbildung Mitarbeiter -> kann in Konflikt mit Geld stehen & Zeit (dauert länger)
- DSGVO Vorgaben können z.B. die Löschung von Userdaten beeinflussen

Aufgabe 5

Schreiben Sie ein komplettes Software Architektur Dokument inklusive aller in der Vorlesung kennengelernten UML-Diagrammen.

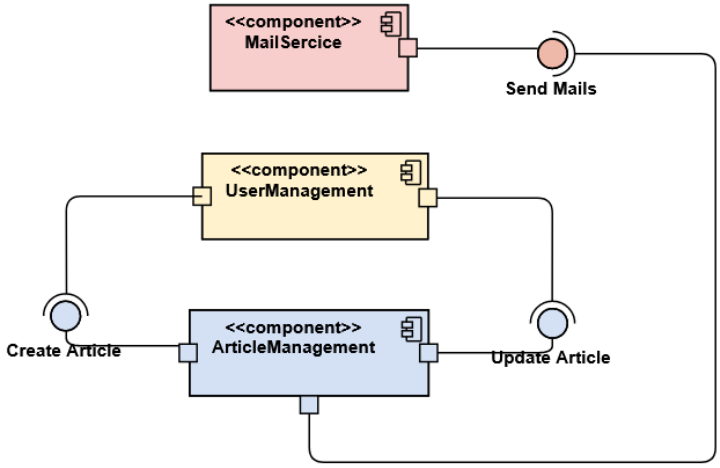
- Administratives (Meta-Information):
    - Titel: LaurasNewsletter
    - Dateiname: ArchitekturDokument_LaurasNewsletter
    - Versionshistorie:
    - Version: 1.0
    - Autor: Laura Tzigiannis
    - Datum zuletzt geändert: 07.05.2020
- Einführung:
    - Leserkreis: Projektleitung, Entwickler, Key User, Projektpate („Woller“)
    - Zweck: Das System soll sowohl Intern für die Mitarbeiter dieser Firma als auch für die Kunden zugänglich sein.
      Es soll die Möglichkeit bieten sich als neuer Abonnent eines bezahlten Newsletters ähnlich einer Tageszeitung zu registrieren. Die Kunden sollen Artikel online abrufen können und über E-Mail über neue Artikel informiert werden.
      Die Mitarbeiter der Firma sollen Ihre Kunden verwalten können und diesen Nachrichten schicken können. Außerdem sollen Sie Artikel erstellen und als Newsletter verschicken können.
- Architekturtreiber:
    - Geschäftskontext (Wo DB ist): https://www.postgresql.org/about/servers/
    - Stakeholder: bei AWS gehostet
    - Begründung für Architektur: entschieden für Spring und Java und ReactJS, da Mehrzahl der Programmierer damit arbeiten will
- Einflussfaktoren:
    - Technische Einflussfaktoren
        - Beschränkungen, die zusätzlich zu denen in der SRS zu beachten sind
        - Hardware: Lenovo ThinkPad, Prozessor: intel Core i5, 8 GB RAM, Festplatte 475 GB
        - Entwicklungswerkzeuge: IntelliJ, Git
        - Bestehende Systemlandschaft: DEV System

        - Bestehende Architektur (Architekturstile): keins (Plan mit Micro Services zu arbeiten)
    - Organisatorische Einflussfaktoren
        - Budget: keins
        - Zeit: bis 17.07.2020
        - Anzahl und Kenntnisstand der Entwickler: 1, WIN Student, Kenntnisse in Java Programmierung mit Spring
        - Make or Buy: Make
        - Unterbeauftragung möglich: Nein
        - Risikobereitschaft des Auftraggebers:
        - Haftung, gesetzliche Anforderungen: DSGVO, wer haftet bei nichteinhalten/ was kann passieren
- Bausteinsicht:
    - Klassendiagramm



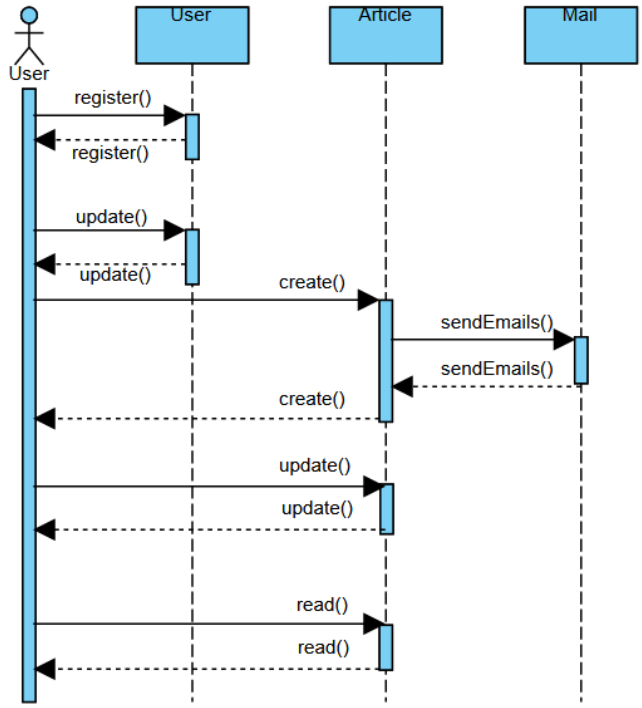
Role Enum: Admin, Customer, Employee

    - Komponentendiagramm

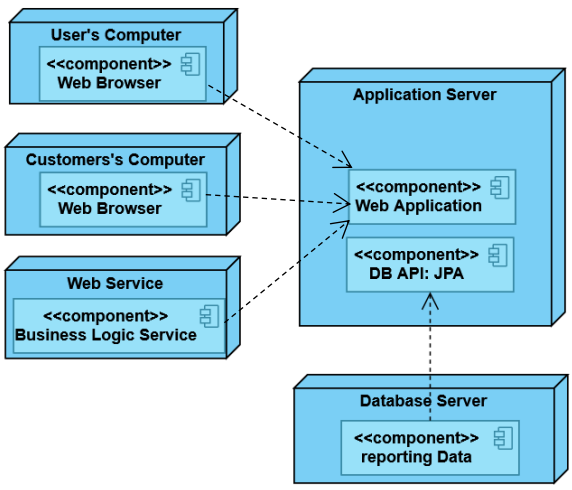


eine Komponente, ist eine Gruppe/Package (mit Entity, Mapper, DTO, Controller, Service, Repository) -> erstellt Komponenten, damit man leicht austauschen kann oder auslagern z.B. zu Micro Services

- Laufzeitsicht
  - Sequenzdiagramme



- Verteilungssicht
  - Verteilungsdiagramme



- Architekturaspekte
  - Persistenz: nach dem ACID Prinzip (Atomicity, Consistency, Isolation, Durability)
  - Fehlerbehandlung: individuelle Exceptions werden erstellt, falls notwendig; werden direkt gehandelt (kein ewiges throw); Info über Exception wir per Email gesendet
  - Transaktionssteuerung (was tun wenn Datensätze nicht komplett sind): roll back und Exception geworfen/ Meldung ausgegeben
  - Logging: evtl jog4j

    TODO: Log4 J nachschauen
- Projektaspekte
  - Werkzeuge: IntelliJ, Git
  - Ansprechpartner: Laura Tzigiannis
- Anhang
  - Glossar: bisher keins
  - Referenzen: keine

Aufgabe 6

Nach der Anforderungsanalyse und dem Architekturdesign geht es nun ans Entwickeln. Implementieren Sie Ihre Mockups prototypische in ReactJS.