Die Planung

Phase I

Planung

> *"Develops a project management plan and other planning documents. Provides the basis for acquiring the resources needed to achieve a solution."*

- Auch wenn Fokus hier mehr auf Ressourcen, Scope usw. liegt → es wird auch die **technical Feasibility** hinterfragt werden
  - In den Folgekapitel wird „**Risiko**" eine essentielle Rolle spielen
  - Eventuell muss man auch Subsysteme / andere Systeme ändern
- Nicht zu verwechseln mit Big Design Up Front (BDUF)

FH University of Applied Sciences TECHNIKUM WIEN

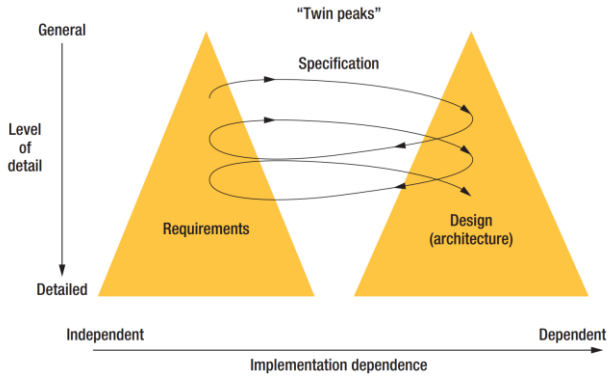
Was hilft uns dieses ganze Theoriewissen?

Fairbanks, G. (2010). Just enough software architecture: a risk-driven approach. Marshall & Brainerd.

*“Since developers cannot grow bigger brains, they have instead **improved their weapons**. An improved weapon gives developers two options: to more easily conquer yesterday’s problems, or to combat tomorrow’s. We are no smarter than developers of the previous generation, but our improved weapons allow us **to build software of greater size and complexity**.”*

Wieviel Software-Architektur Planung sollte man betreiben?

- Viele Ansätze …
- Ein zielführender Ansatz: Bewertung anhand des **Risikos** …
- Wie hoch ist das Risiko der beiden Applikationen?
  - Consolen-Applikation zum Konvertieren CSV → JSON
  - Web-Applikation zum Verwalten von Gesundheitsdaten von Patienten



Randbemerkung: leichte Überschneidung mit der Design Phase des SDLC. Aber wir wissen: Architektur wächst mit den Anforderungen – in der Planung sehr grobe Anforderungen. Ressource-Planung: Brauchen wir einen Security-Experten? Oder andere Experten? Grob-Architektur als Kommunikationshilfe.

Was sind Smells (hohe Risiken), um auf die Planung von Software-Architektur zu achten?

- Wenig Lösungsalternativen
  - Die Planung ist wichtig, wenn es wenig Lösungsalternativen gibt (Flexibilität und ggf. Einfachheit sinkt) bzw. eine akzeptable Lösung schwer zu entwerfen ist
- Hohes Risiko des Misserfolgs
  - Einem Safety-Critical Environment (e.g. medizinischer Bereich) können Menschen sterben, wenn es zu Fehlern kommt.
  - Eine Security-Lücke könnte das Vertrauen der Kunden stark beinträchtigen und große Einbußen mit sich bringen
  - uvm.

Cont.

- Fordernde Qualitätsattribute
  - Nicht funktionale Anforderungen
  - Performance, Dependability [1], Security, Usability, Modifiability, …
- Neue Domäne
  - Wenn die Firma bereits die 10. Applikation in einer Domäne baut ist mehr Erfahrung vorhanden, als bei der ersten – daher mehr Augenmerk / Risiko einplanen
- Produkt-Linien
  - Produkte in Firmen teilen oft die Architektur – das kann die Sache einfacher machen – aber auch wesentlich komplexer

[1] https://en.wikipedia.org/wiki/Dependability

Bau-Architektur vs. Software-Architektur

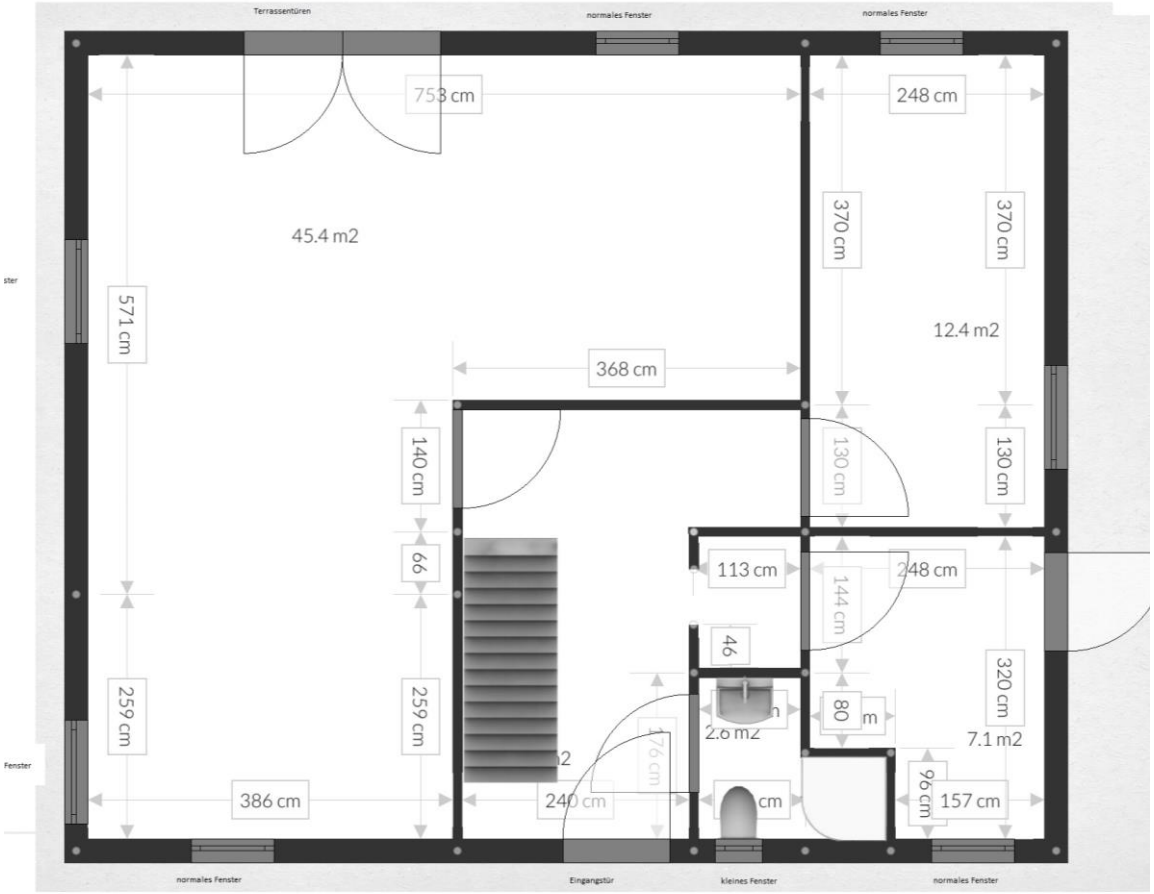
- Sehr viele Ähnlichkeiten – die Software-Architektur hat quasi den Ursprung in der Bau-Architektur
- Beispiel folgt:
  - Wie viel Architektur-Planungsaufwand würde man in die beiden folgenden Projekte investieren?
  - Wie hoch ist das Risiko bei beiden Projekten?

Projekt 1: Gartenhütte

Projekt 2: Wohngebäude

Projekt 1 vs. Projekt 2

- Würde man bei einem Wohngebäude Maurer (analog: Software-Entwickler) ohne Planung der Architektur zu mauern (analog: programmieren) beginnen lassen?
- Modelliert man in einem Bauplan jedes Detail? Wo ein Spiegel hängt oder die Wandfarbe? Nein! Nur jene Sachen, die zur Analyse, ob die Treiber der Architektur erreicht wurden, erforderlich sind (mehr dazu in Folgekapitel) *



* Alle architekturellen Entscheidungen, welche nichts zur Entscheidung betragen, ob die Treiber der Architektur erreicht wurden, heißen nicht-architekturell / Software-Design Entscheidungen

Technical Feasibility

- Presumptive architectures: *„A presumptive architecture is a family of architectures that is dominant in a particular domain.”*
  - Architektur für das Speichern von Personendaten
  - Architektur für IoT
  - Architektur für Trading
  - Architektur für selbstfahrende Autos
  - …
- Sollte man sich auf keine vorhanden Erfahrungen stützen können: je nach Risiko Platz für Prototypen einbauen, um Erfahrungen sammeln zu können (wichtig: Risiko erfassen, essentielle Anforderungen herausfinden)