Das magische Viereck erfolgreicher Softwareprojekte

A diagram of a project

Description automatically generated

Softwaresyteme sind ein zentrales Ruckgrat unserer Gesellschaft! Die Fahigkeit IT-Projekte durchzufuhren muss verbessert warden! => Software Engineering / Software Architektur.

Warumg sind Kenntnisse uber Softwarearchitektur wichtig?

* Komplexitat von Softwareprojekten
* Skalierbarkeit und Leistung
* Sicherheit und Datenschutz
* Technologische Vielfalt
* Agilitat und kontinuierliche Bereitstellung

**Big Ball of Mud**

= Begriff um eine unstrukturierte und chaotische Softwarearchitektur zu beschreiben

Merkmale:

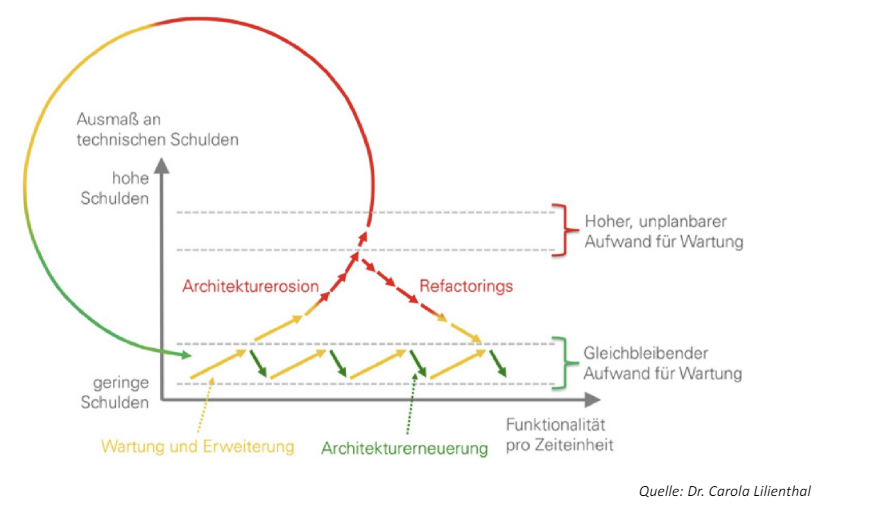
* Mangel an Struktur
* Schlechte Modularisierung
* Spaghetti-Code
* Technologische Vielfalt
* Wachsende Komplexitat

**Technische Schulden**

-> Enstehen, wenn Teams bewusst Entscheidungen treffen, die kurzfristig Vorteile bringen, aber langfristig Auswirkungen auf die Qualitat haben

Arten von technischen Schulden:

* Schnelle Losungen
* Veraltete Technologien
* Fehlende Tests
* Schlechte Codequalitat
* Nicht behandelte Probleme



**Arten von IT-Architekten und Aufgaben**

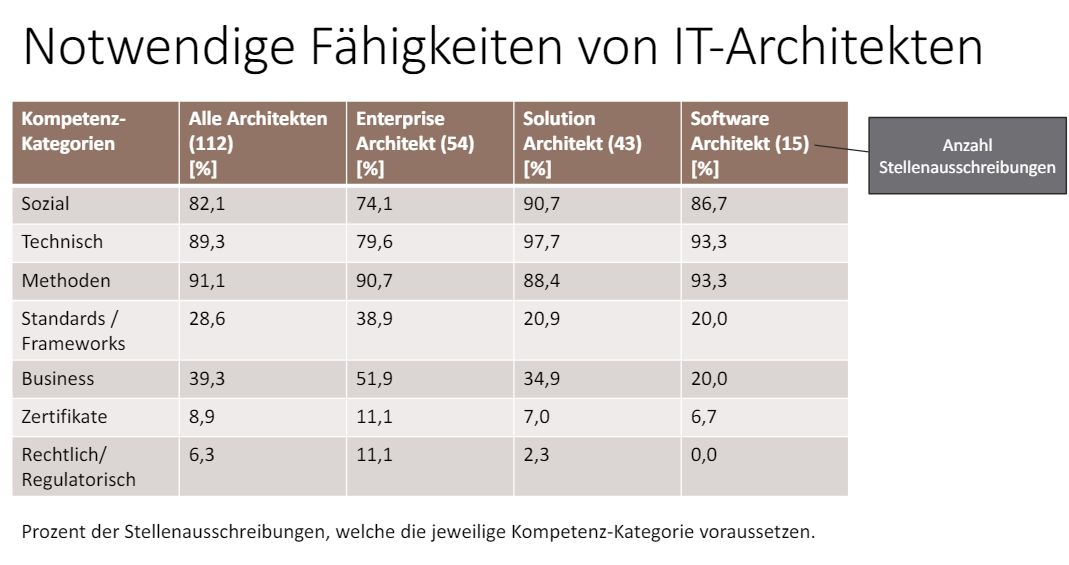
* Enterprise Architekt
  + -> fokusieren sowohl and den business Aspekt, nich fur einen einzigen Projekt, sonder fur das Unternehmen im ganzen
  + Starke strategische Ausrichtung
  + IT-Business-Alignment
  + Technologie-Roadmaps
* Solution / System Architekt
  + -> Projektfokus, leiten spezifische systeme
  + Leaderschip
  + Eingebunden in Anforderungsspezifikationen
  + Systemintegration
* Software Architekt
  + -> konzentrieren on technischen Aspekte innerhalb einer Software App
  + Erstellen und Managen von Softwarearchitekturen
  + Enge Zusammenarbeit mit Entwicklungsteam
  + Kooperation, Teamwork, Netzwerken

Typische Aufgaben von Softwarearchitekten

* Architektonische Plannung
* Technologieauswahl
* Code-Reviews und Best Practices
* Skalierung und Performance
* Risikomanagement
* Kommunikation und Zussamenarbeit
* Wartbarkeit und Erweitbarkeit
* Qualitatssicherung

**Wann ist eine Softwarearchitektur erfolgreich?**

Der Erfolg einer Softwarearchitektur bemist sich am Erfolg der Software



**Softwarearchitektur**

=Grundlegende Konzepte oder Eigenschaften einer Entitat ihere Umgebung und leitende Prinzipien fur die Realisierung und Entwicklung dieser Entitat und der damit verbundenen Lebenszyklusprozesse

**Grundlegende Konzepte** -> Sie sollen in der Regel durch Komponenten der Entitat, die Beziehungen zwischen den Komponenten und durch die Beziehungen zwischen der Entitat und ihrer Umgebung verkorpert warden.

**Architekrureinheit =** thing being considered, described, discussed, studied, or otherwise addressed during the architecting effort

Fundamentale Konzepte und Eigenschaften eines Systems

* Die Komponenten des Systems
* Die Beziehungen/ Schnittstellen zwischen Komponenten
* Die Beziehungen zwischen dem System und seiner Umgebung

**Schnittstelle**

Eine Schnittstelle reprasentiert einen wohldefinierten Zugangspunkt zum System oder zu dessen Komponente. Dabei beschreibt eine Schnittstelle die Eigenschaften dieses Zugangspunkts, wie z. B. Attribute, Daten und Funktionene.

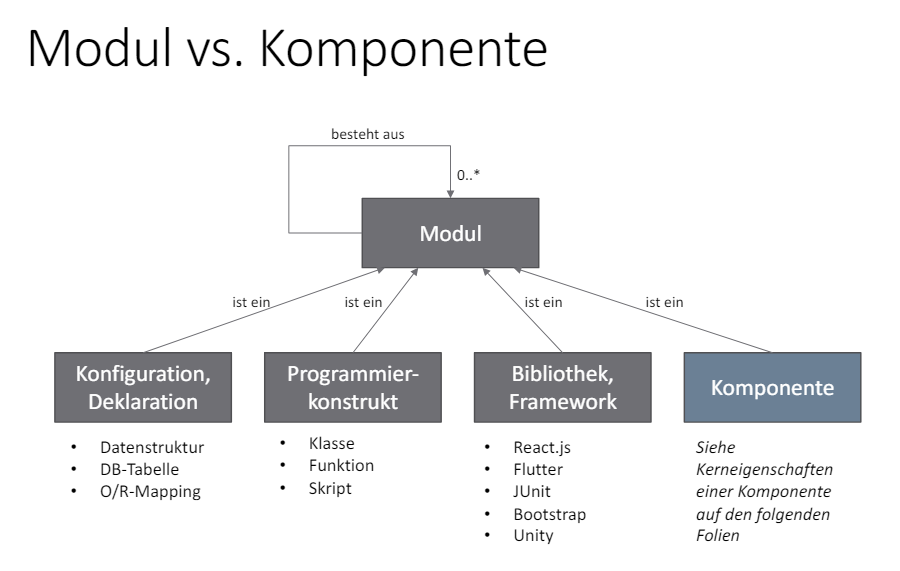
Arten von Schnittstellen:

* Angebotene Schnittstellen:
  + Wird von einer Komponente bereitgestellt
  + Definiert, welche Dienste oder Funktionen verfugbar sind
  + Andere Komponente konnen diese Schnittstelle nutzen, um Dienste in Anspruch zu nehmen
* Angeforderte Schnittstellen:
  + Wird von einer Komponente benotigt
  + Definiert Abhangigkeiten und Anforderungen and andere Komponenten
  + Komponente kann nur dann funktionieren, wenn alle angeforderten Schnittstellen von anderen Komponenten bereitgestellt warden

Verantwortung fur Schnittstellen:

* Normungsorganisationen
  + Fur allgemein akzeptierte Schnittstellen
  + Normungsorganisationen definieren Schnittstellen; Implementierung und Wartung meist durch Drittanbieter vorgenommen
* Schnittstellenanbieter
  + Schnittstellenanbieter kann verantwortlich fur die Definition, Entwicklung, Wartgun und Dokumentation der Schnittstelle sein (ist meistens der Fall)
* Schnittstellenverwender
  + Ein Verwender der Schnittstelle gleichzeitig die Gesamtverantwortung oder Teile der Verantwortung fur angebotene Schnittstellen ubernimmt

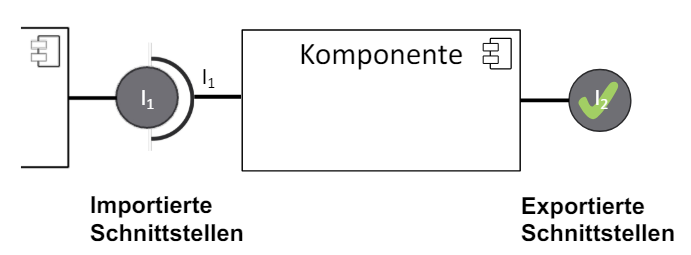
**Komponenten**



**Kerneigenschaften einer Komponente:**

* Export und Import von Schnittstellen
* Kapselung und Austauschbarkeit
* Konfigurierbarkeit

**Export und Import von Schnittstellen**

****

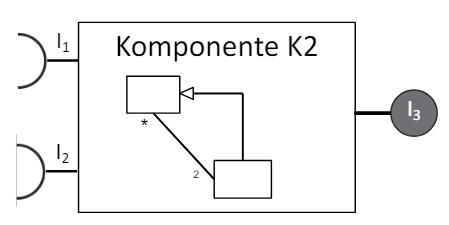
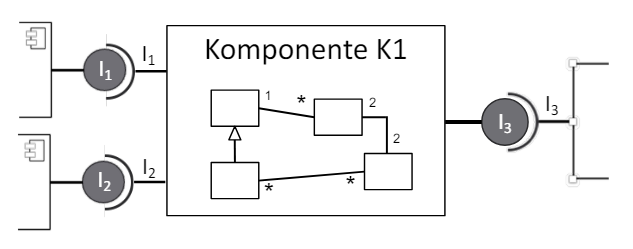
Eine Komponente bietet Schnittstellen an, die sie im Sinne eines Vertrags garantiert.

Diese Garantie gilt unter der Bedingung, dass die von ihr benotigten Schnittstellen im Rahmen einer entsprechenden Konfiguration bereitgestellt warden

**Austauschbarkeit und Kapselung**

Uber die Schnittstellen kapselt eine Komponente die Implementierung

Solange alle Schnittstellenvertrage erfullt warden, konnen Komponenten ersetzt warden



**Konfigurierbarkeit**

Arten von Konfigurierbarkeit:

* Konfiguration von Eigenschaften und Verhalten
* Komposition von Komponenten
* Dynamische vs. statische Konfiguration

Vorauseztung von Konfigurierbarkeit: Effektives Konfigurationsmanagement

* Konfigurationsdataien
* Versionierung
* Fehlermanagement