



iSAQB Foundation Level

Grundbegriffe von Softwarearchitekturen



Agenda

1. LZ 1-1: Definitionen von Softwarearchitektur diskutieren (R1)
2. LZ 1-2: Nutzen und Ziele von Softwarearchitektur verstehen und erläutern (R1)
3. LZ 1-3: Softwarearchitektur in Software-Lebenszyklus einordnen (R2)
4. LZ 1-4: Aufgaben und Verantwortung von Softwarearchitekt:innen verstehen (R1)
5. LZ 1-5: Rolle von Softwarearchitekt:innen in Beziehung zu anderen Stakeholdern setzen (R1)
6. LZ 1-6: Zusammenhang zwischen Entwicklungsvorgehen und Softwarearchitektur erläutern können (R1)
7. LZ 1-7: Kurz- und langfristige Ziele differenzieren (R1)
8. LZ 1-8: Explizite von impliziten Aussagen unterscheiden (R1)
9. LZ 1-9: Zuständigkeit von Softwarearchitekt:innen in organisatorischen Kontext einordnen (R3)
10. LZ 1-10: Typen von IT-Systemen unterscheiden (R3)
11. LZ 1-11: Herausforderungen verteilter Systeme (R3)

Grundbegriffe von Softwarearchitekturen

- Dauer: 120 Min. Übungszeit: keine
- Wesentliche Begriffe:
 - Softwarearchitektur; Architekturdomeänen; Struktur; Bausteine; Komponenten; Schnittstellen; Beziehungen; Querschnittskonzepte; Nutzen von Softwarearchitektur; Softwarearchitekten und deren Verantwortlichkeiten; Rolle; Aufgaben und benötigte Fähigkeiten; Stakeholder und deren Anliegen; funktionale Anforderungen; Qualitätsanforderungen; Randbedingungen; Einflussfaktoren; Typen von IT-Systemen

Software ist überall und der Bedarf wächst

"IT no longer just supports corporate operations as it traditionally has, but is fully participating in business value delivery," said [John-David Lovelock](#), distinguished research vice president at Gartner. "Not only does this shift IT from a back-office role to the front of business, but it also changes the source of funding from an overhead expense that is maintained, monitored and sometimes cut, to the thing that drives revenue."

All IT spending segments are forecast to have positive growth through 2022 (see Table 1). The highest growth will come from [devices](#) (14%) and enterprise software (10.8%) as organizations shift their focus to providing a more comfortable, innovative and productive environment for their workforce.

Table 1. Worldwide IT Spending Forecast (Millions of U.S. Dollars)

	2020 Spending	2020 Growth (%)	2021 Spending	2021 Growth (%)	2022 Spending	2022 Growth (%)
Data Center Systems	219,940	2.3	236,806	7.7	247,513	4.5
Enterprise Software	466,647	-2.1	516,872	10.8	571,725	10.6
Devices	663,223	-6.9	755,798	14.0	778,949	3.1
IT Services	1,021,187	-1.8	1,112,626	9.0	1,193,461	7.3
Communications Services	1,386,471	-0.7	1,450,444	4.6	1,504,743	3.7
Overall IT	3,757,468	-2.2	4,072,547	8.4	4,296,391	5.5

Source: Gartner (April 2021)

4 <https://www.gartner.com/en/newsroom/press-releases/2021-04-07-gartner-forecasts-worldwide-it-spending-to-reach-4-trillion-in-2021> 



Wie helfen uns Softwarearchitekturen?

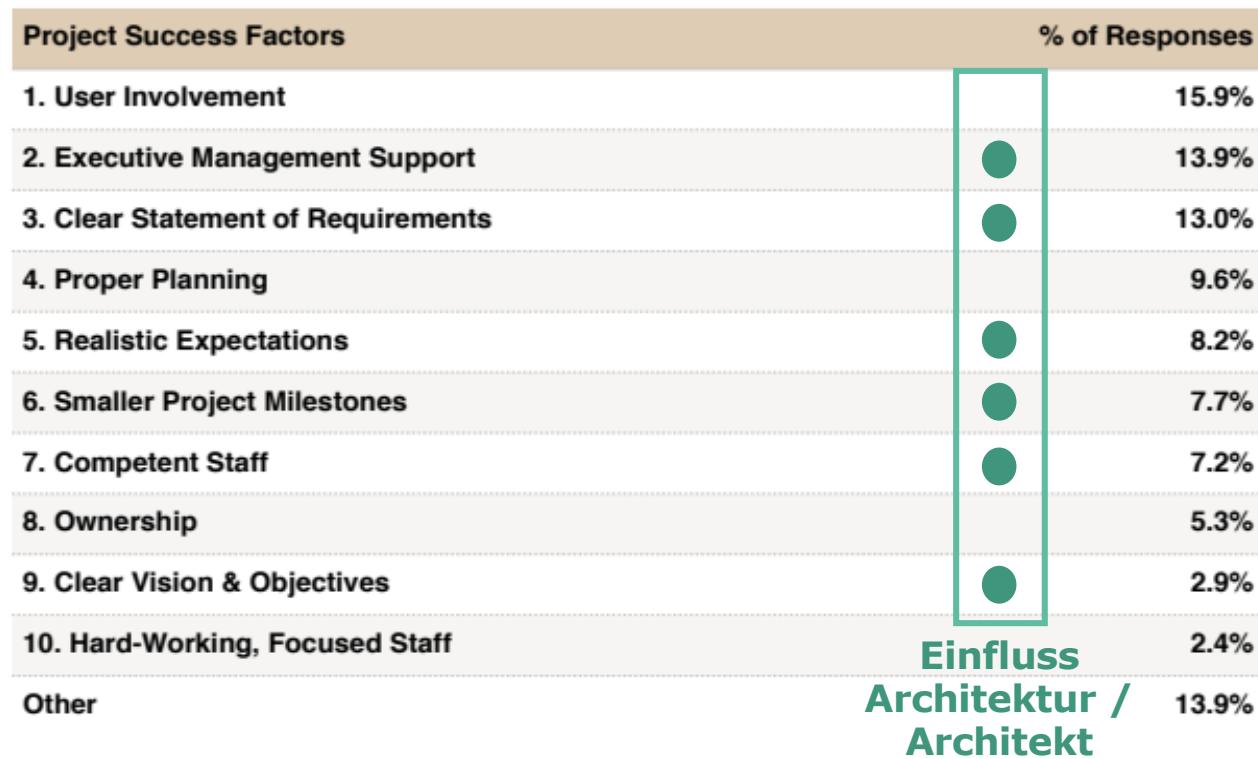
Der Chaos Report der Standish Group analysiert Software Projekte
(Drei Kategorien: Erfolg / Fertigstellung mit Einschränkungen / Abbruch)

Cost		Time		Features	
Cost Overruns	% of Responses	Time Overruns	% of Responses	% of Features/Functions	% of Responses
Under 20%	15.5%	Under 20%	13.9%	Less Than 25%	4.6%
21 - 50%	31.5%	21 - 50%	18.3%	25 - 49%	27.2%
51 - 100%	29.6%	51 - 100%	20.0%	50 - 74%	21.3%
101 - 200%	10.2%	101 - 200%	35.5%	75 - 99%	39.1%
201 - 400%	8.8%	201 - 400%	11.2%	100%	7.3%
Over 400%	4.4%	Over 400%	1.1%		

<https://www.projectsmart.co.uk/white-papers/chaos-report.pdf> aus dem Jahre 2014

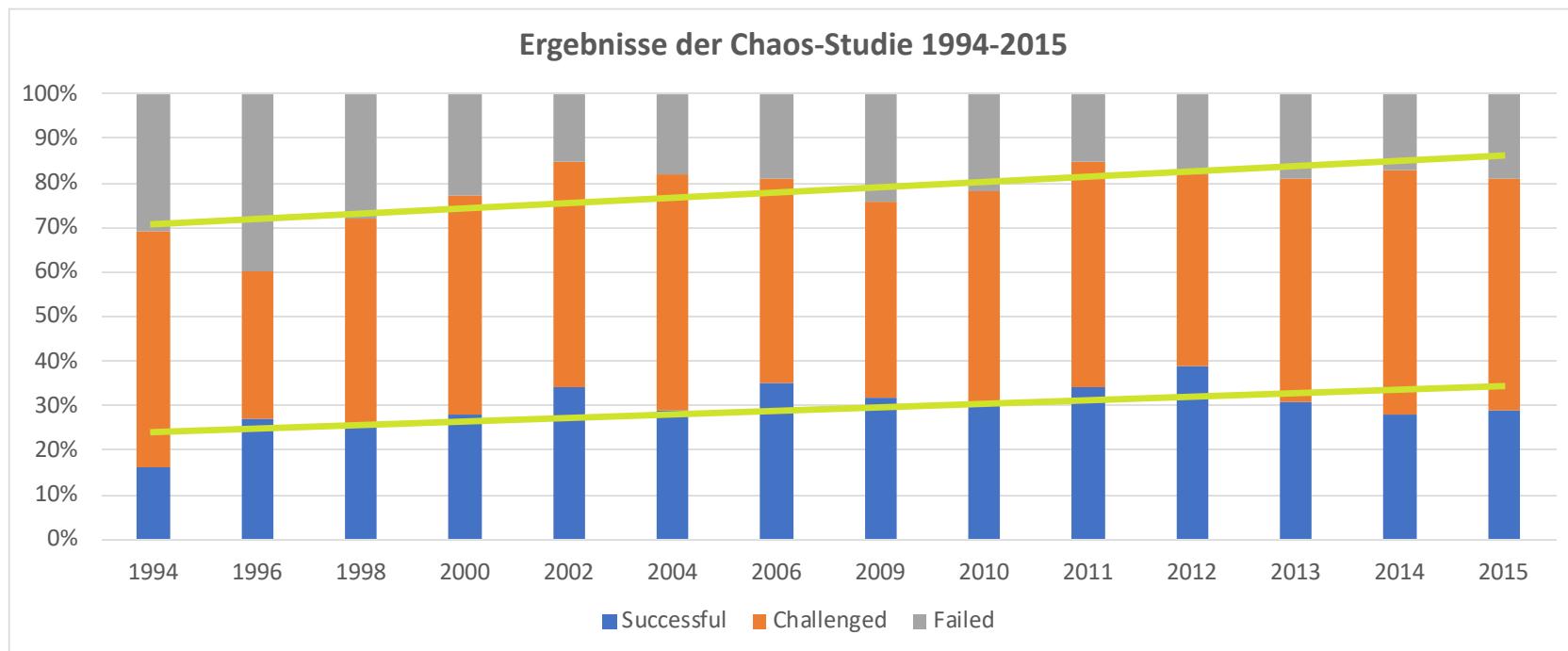
Wie helfen uns Softwarearchitekturen?

Der Chaos Report der Standish Group analysiert Software Projekte
(Drei Kategorien: Erfolg / Fertigstellung mit Einschränkungen / Abbruch)



Wie helfen uns Softwarearchitekturen?

Über einen Zeitraum von 22 Jahren betrachtet, hat sich eine stabile Verbesserung ergeben, die jedoch mit jeweils ca. 10% zwischen Challenged und Failed sowie zwischen Challenged und Success gering ist.



LZ 1-1: Definitionen von Softwarearchitektur diskutieren (R1)

- Softwarearchitekt:innen kennen mehrere Definitionen von Softwarearchitektur (u. a. ISO 42010/IEEE 1471, SEI, Booch etc.) und können deren Gemeinsamkeiten benennen:
 - Komponenten/Bausteine mit Schnittstellen und Beziehungen
 - Bausteine als allgemeiner Begriff, Komponenten als eine spezielle Ausprägung davon
 - Strukturen, Querschnittskonzepte, Prinzipien
 - Architekturentscheidungen mit systemweiten oder den gesamten Lebenszyklus betreffenden Konsequenzen

Der Begriff der Softwarearchitektur

*Architecture is the fundamental **organization** of a **system** embodied in its **components**, their **relationships** to each other, and to the **environment**, and the **principles** guiding its design and evolution.*

[IEEE 1471]

*An architecture is the set of significant decisions about the **organization** of a software **system**, the selection of the **structural elements** and their **interfaces** by which the system is composed, together with their behavior as specified in the **collaborations** among those elements, the **composition** of these structural and behavioral elements into progressively larger **subsystems**, and the architectural style that guides this organization --- these elements and their interfaces, their collaborations, and their composition*

[The Rational Unified Process]

*Software architecture is the **structure** or structures of the **system**, which comprise **software elements**, the externally visible **properties** of these elements, and the **relationships** among them*

[SEI].

Softwarearchitektur ist auch...

*...die **Summe der Entscheidungen**, die falsch getroffen ein **Risiko für Qualität, Budget und Timeline** darstellen!*

*...ein **Kommunikationsmittel!***

*...das Schaffen von **Ordnung** und **Überblick***

*...**strukturelle Qualität!***

*...die **Basis für Erweiterbarkeit** und **Wiederverwendung***

*...eng verzahnt mit der **Implementierung***

Bausteine sind die Grundbestandteile von Software

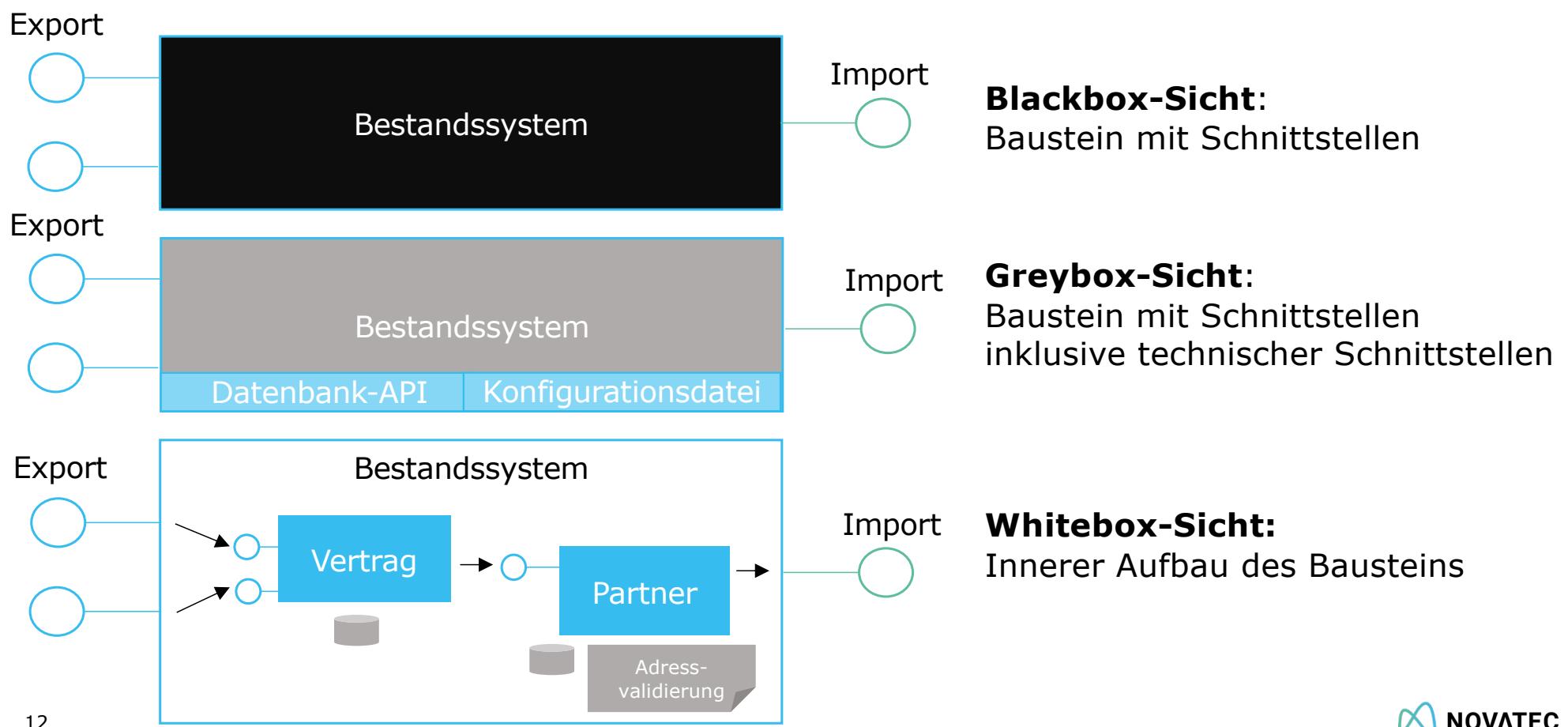
Bausteine...

- stellen Schnittstellen zur Verfügung
- nutzen Schnittstellen von anderen Bausteinen (**Beziehungen**)
- kapseln Funktionalität (**Verantwortungsbereich**)

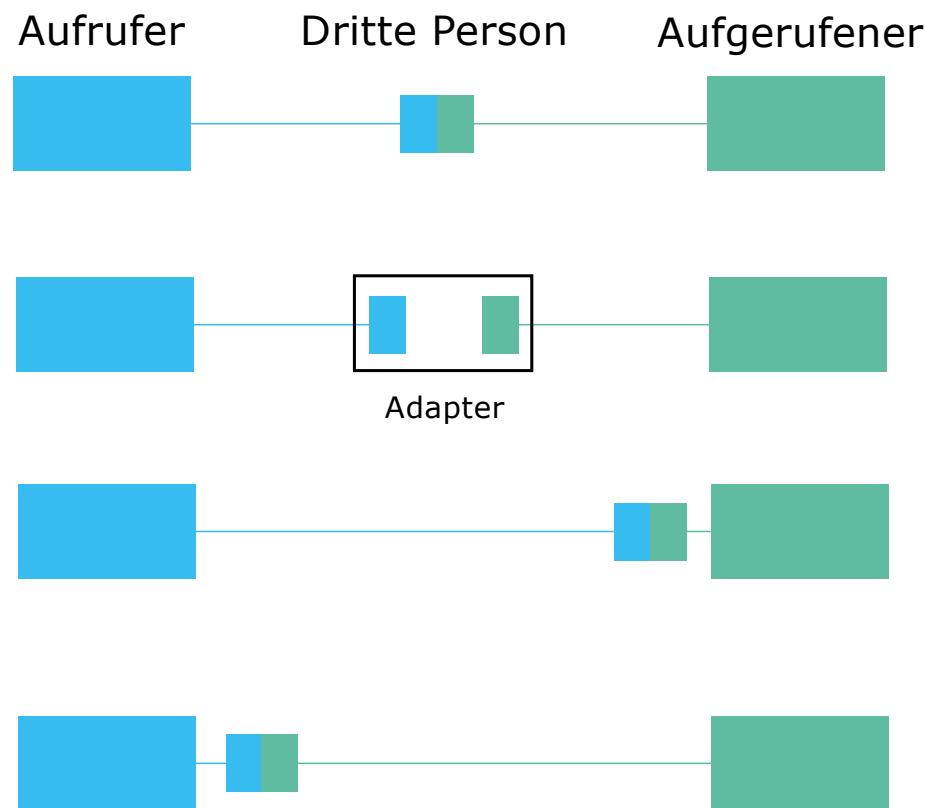
Komponenten sind konkrete Ausprägungen von Bausteinen (**Austauschbarkeit**)



Sichten auf Bausteine



Schnittstellen und ihr Vertrag



Standardschnittstelle

Aufrufer und Aufgerufener halten den Schnittstellenvertrag ein, der von außen festgelegt ist.

Unabhängige Schnittstellen

Aufrufer und Aufrufender haben eigene Schnittstellen, die mithilfe eines Adapters verbunden werden.

Angebote Schnittstelle

Der Aufgerufene definiert die Schnittstelle. Diese Art des Vertrags wird sehr häufig verwendet.

Angeforderte Schnittstelle

Der Aufrufer legt den Vertrag fest. Diese Art wird häufig bei Frameworks verwendet.

Bausteine werden entsprechend ihrer Funktionalität zu Schichten zugeordnet

User Interface

Darstellung der Nutzerschnittstelle, Eingabe- und Ausgabe

Service Layer

Beschreibt die Schnittstelle (Technologie zentriert)

Application Layer

Beschreibt die Fachlichkeit (Dienste oder Geschäftsprozesse)

Entity Layer

Fachliche Geschäftsobjekte mit persistentem Zustand

Infrastructure Layer

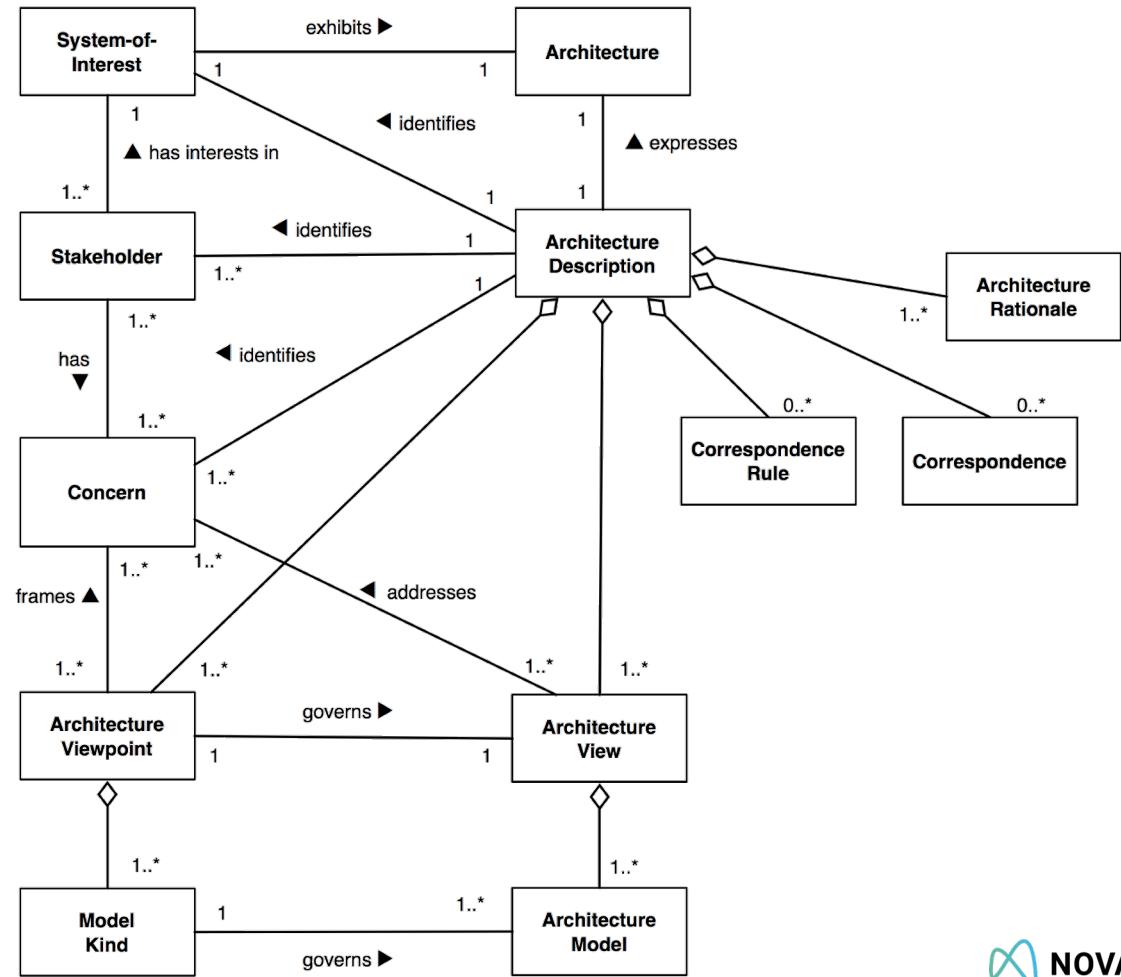
Allgemeine technische Services (z.B. Persistenz) und Kommunikation

- Schichten dienen der Strukturierung von Systemen
- Jede Schicht hat klar definierte Verantwortlichkeiten
- Bausteine werden einzelnen Schichten zugeordnet.

Architektur entsteht auf Basis bewusster Entscheidungen

- Architektur entsteht **nicht am Reißbrett** eines elitären Personenkreises (Elfenbeinarchitektur), sondern **begleitend zur Projektdurchführung**
- Immer dann wenn unterschiedliche Alternativen zur Umsetzung einer Problemstellung existieren, sollte eine **bewusste Architekturentscheidung** getroffenen und entsprechend **dokumentiert** werden. Das hilft diese Entscheidung zu einem späteren Zeitpunkt nachvollziehen und wenn nötig erneut validieren zu können, falls sich beispielsweise Randbedingungen verändert haben
- Architekturentscheidungen sollten sich stets an den **Anforderungen und Werten der Stakeholder** orientieren und nicht an den Vorlieben des Architekten
- Eine Architekturentscheidung kann entweder **taktisch** (fokussiert einen kürzeren Zeitraum) oder **strategisch** (fokussiert einen längeren Zeitraum) erfolgen.

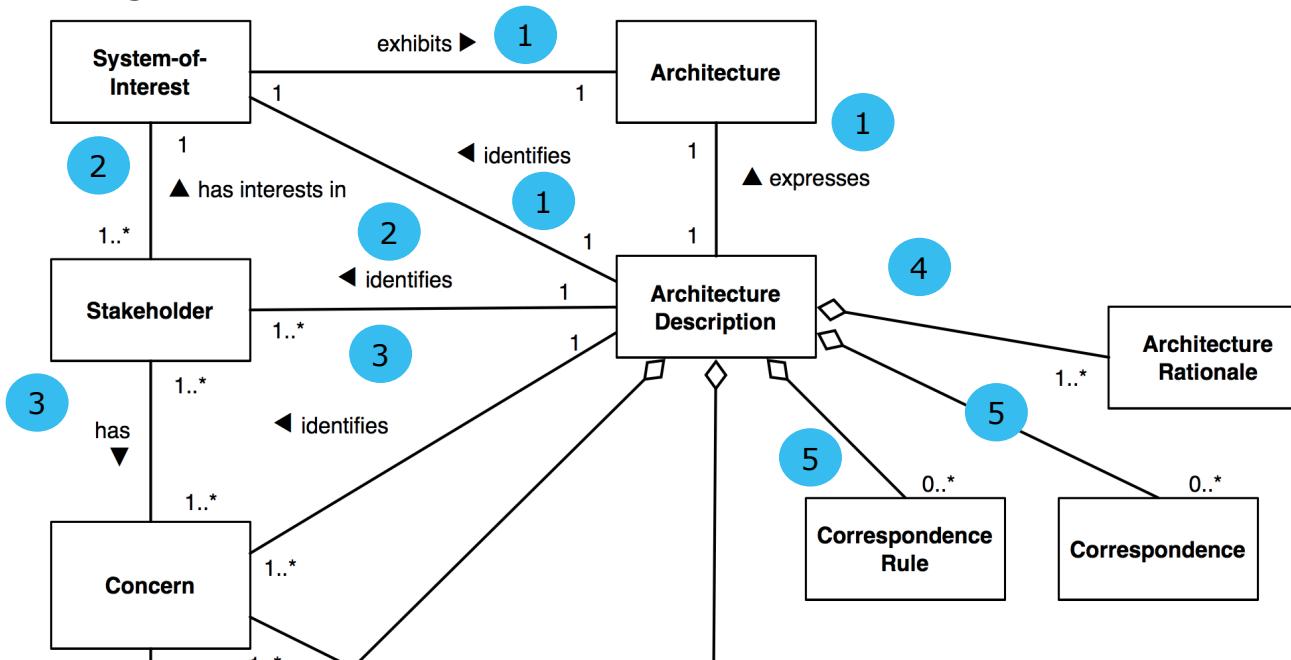
ISO/IEC/IEEE 42010:201



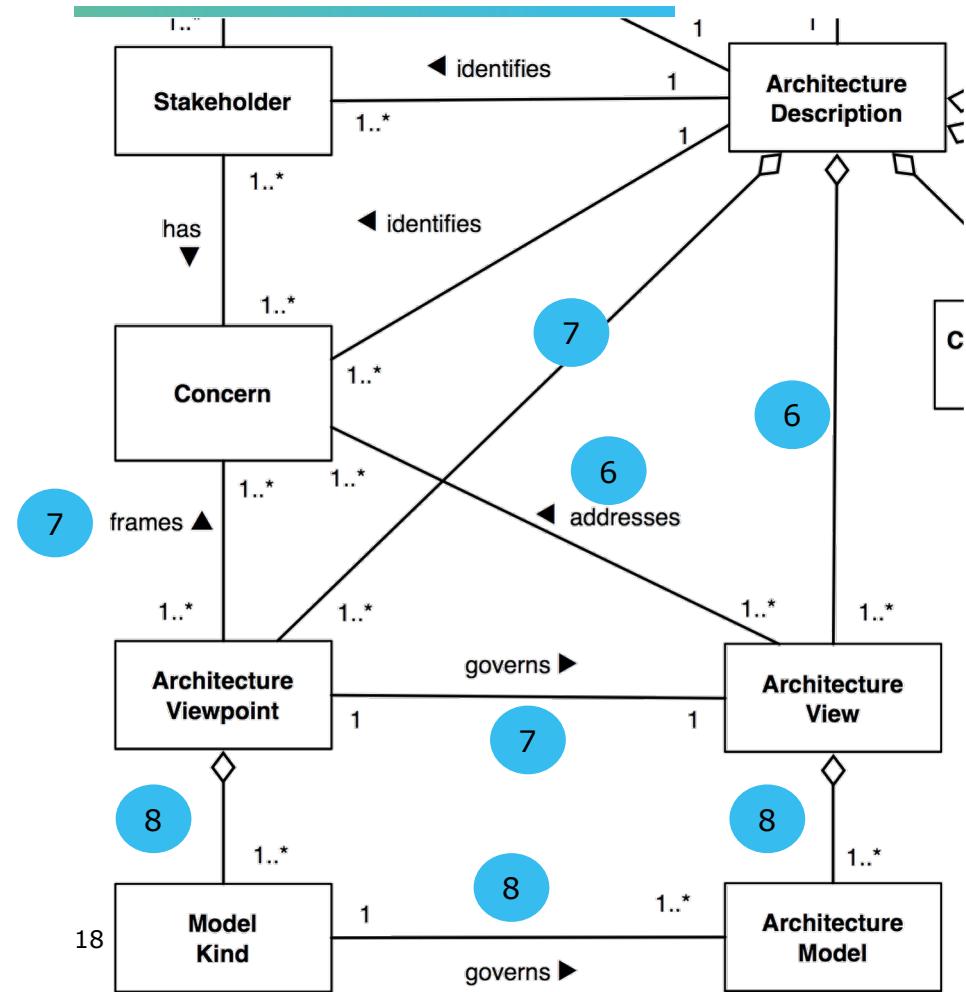
<http://www.iso-architecture.org/ieee-1471/cm/>

Modell zur Architekturbeschreibung

- 1 Ein System besitzt eine Architektur, beschrieben durch eine Architekturbeschreibung
- 2 Die Architekturbeschreibung nennt Stakeholder des Systems
- 3 Jeder Stakeholder hat Anliegen, die Teil der Architekturbeschreibung sind
- 4 Architektur wird anhand von begründete Architekturentscheidungen beschrieben
- 5 Teile der Architekturbeschreibung stehen in Zusammenhang mit Teilen der Architektur und identifizieren diese



Modell zur Architekturbeschreibung



- 6 Die Architekturbeschreibung umfasst mehrere Sichten (Architecture View) auf die Architektur, die Anliegen von Stakeholdern visualisieren
- 7 Abhängig vom Standpunkt (Architecture Viewpoint) existieren verschiedene Sichten auf die Architektur. Standpunkte bilden Anliegen ab
- 8 Jeder Standpunkt wird durch verschiedene Modellelemente beschrieben. Ein Modellelement beeinflusst Architekturmodelle, die eine Sicht beschreiben

Sichten (View) und Standpunkte (Viewpoint)

Beispiel: Luftfahrtsystem

- Die Sicht des Piloten umfasst Passagiere, Benzin, Reiseflughöhe, Geschwindigkeit und der Typ des Flugzeugs
- Die Sicht des Fluglotsen umfasst Flugzeuge, Reiseflughöhe, Routen und Wetter
- Ein Standpunkt (*Template = Viewpoint*) drückt die Art der Beschreibung für die Informationen innerhalb einer Sicht (*Instanz = View*) aus
- Piloten beschreiben ein Flugzeug aus ihrer Perspektive auf Basis der aktuellen Position und der Richtung von Startbahn bzw. Landebahn
- Fluglotsen beschreiben Flugzeuge auf eine andere Art. Sie nutzen ein Modell des Luftraums sowie die aktuellen Positionen und die Bewegungsvektoren der Flugzeuge innerhalb des Luftraums.

Sichten (View) und Standpunkte (Viewpoint)

Beispiel: Versicherungsanwendung

- Zwei verschiedene Standpunkte: Product Owner und Betrieb
- Jede der beiden Standpunkte (Viewpoint) besitzt eine spezielle Sicht (View) auf das System (Versicherungsanwendung)
- Für den Product Owner ist die fachliche Sicht (Bestandsystem, Partnersystem, Tarifierung, etc.) interessant
- Für den Betrieb ist die Verteilungssicht (Server, Datenbanken, IP-Adressen, Protokolle, Firewall) interessant.

LZ 1-2: Nutzen und Ziele von Softwarearchitektur verstehen und erläutern (R1)

- Softwarearchitekt:innen können folgenden Nutzen und wesentlichen Ziele von Softwarearchitektur begründen:
 - Entwurf, Implementierung, Pflege und Betrieb von Systemen zu unterstützen
 - Qualitätsanforderungen wie Zuverlässigkeit, Wartbarkeit, Änderbarkeit, Sicherheit zu erreichen
 - funktionale Anforderungen zu erreichen bzw. deren Erfüllbarkeit sicherzustellen
 - Verständnis für Strukturen und Konzepte des Systems zu vermitteln, bezogen auf sämtliche relevanten Stakeholder
 - systematisch Komplexität zu reduzieren
 - architekturelle Richtlinien für Implementierung und Betrieb zu spezifizieren

Qualität von Softwarearchitektur

Die Qualität der Softwarearchitektur ist der Grad, indem die Software die Bedürfnisse der Stakeholder erfüllt [ISO/IEC 25010].

Übergeordnete Qualitätsmerkmale

- Funktionale Eignung
- Zuverlässigkeit
- Bedienbarkeit
- Leistungseffizienz
- Skalierbarkeit
- Sicherheit
- Wartbarkeit
- Kompatibilität
- Übertragbarkeit

Produktspezifische Qualitätsmerkmale

- Time-to-market
- Nutzbarkeit als Mobile App / Mobile First Ansatz
- API-First Ansatz
- Einheitliche Codebasis
- Deployment in der Cloud (Plattform)
- Verständlichkeit des Codes
- Dokumentation
- DevOps des Produktes muss gewährleistet sein

Nutzen von Softwarearchitektur

Qualitative Softwarearchitektur ...

- ist änderbar und erweiterbar ohne bestehende Qualitätseigenschaften zu gefährden
- vermeidet Änderungen an tragenden Elementen und somit unangemessene Kosten-Nutzen-Verhältnisse
- ist einfacher zu verstehen und in Ausnahmensituationen zu analysieren
- erleichtert die Integration und Pflege von Schutzmaßnahmen
- ermöglicht es Teams in hoher Geschwindigkeit End-to-End Funktionalität zu implementieren und somit einen hohen Business Value zu generieren
- ermöglicht und vereinfacht Wiederverwendung
- stellt konzeptionelle Integrität sicher.

Reduktion von Komplexität

Ein Schlüssel ein erfolgreiches Softwaresystem zu realisieren ist liegt in Reduktion der Komplexität. Dies kann anhand folgender Prinzipien erfolgen:

- Eine Softwarearchitektur nutzt eine **Schichtenarchitektur** um die erstellten Artefakte gemäß ihren Aufgaben zuzuordnen
- Wohldefinierte **Schnittstellen** erhöhen die Verständlichkeit des Systems und klare Zuordnung von Verantwortlichkeiten
- Das Erstellen einer **Referenzarchitektur** macht die verschiedenen Typen von Softwareartefakten sichtbar und erläutert deren Zusammenspiel
- Die Anwendung des **KISS-Prinzips** (Keep it simple stupid) führt dazu, dass die einfachste Lösung umgesetzt wird
- Die Anwendung des **YAGNI-Prinzips** (you ain't goinna need it) führt dazu, dass auf Abstraktionen verzichtet werden, die im Moment nicht benötigt werden.

LZ 2-3: Einflussfaktoren auf Softwarearchitektur erheben und berücksichtigen können (1) (R1-R3)

Softwarearchitekt:innen können Einflussfaktoren (Randbedingungen) als Einschränkungen der Entwurfsfreiheit erarbeiten und berücksichtigen. Sie verstehen, dass ihre Entscheidungen weitere Anforderungen und Einschränkungen für das zu entwerfende System, seine Architektur oder den Entwicklungsprozess mit sich bringen können.

Sie erkennen und berücksichtigen den Einfluss von:

- produktbezogenen Faktoren wie (R1)
 - funktionale Anforderungen
 - Qualitätsanforderungen und Qualitätsziele
 - zusätzliche Faktoren wie Produktkosten, beabsichtigtes Lizenzmodell oder Geschäftsmodell des Systems

LZ 2-3: Einflussfaktoren auf Softwarearchitektur erheben und berücksichtigen können (2) (R1-R3)

- technischen Faktoren wie (R1-R3)
 - extern beauftragte technische Entscheidungen und Konzepte (R1)
 - bestehende oder geplante Hardware- und Software-Infrastruktur (R1)
 - technologische Beschränkungen für Datenstrukturen und Schnittstellen (R2)
 - Referenzarchitekturen, Bibliotheken, Komponenten und Frameworks (R1)
 - Programmiersprachen (R3)
 - organisatorischen Faktoren wie
 - Organisationsstruktur des Entwicklungsteams und des Kunden (R1)
 - Unternehmens- und Teamkultur (R3)
 - Partnerschaften und Kooperationen (R2)
 - Normen, Richtlinien und Prozessmodelle (z.B. Genehmigungs- und Freigabeprozesse) (R2)
 - Verfügbarkeit von Ressourcen wie Budget, Zeit und Personal (R1)
 - Verfügbarkeit, Qualifikation und Engagement von Mitarbeitenden (R1)

LZ 2-3: Einflussfaktoren auf Softwarearchitektur erheben und berücksichtigen können (3) (R1-R3)

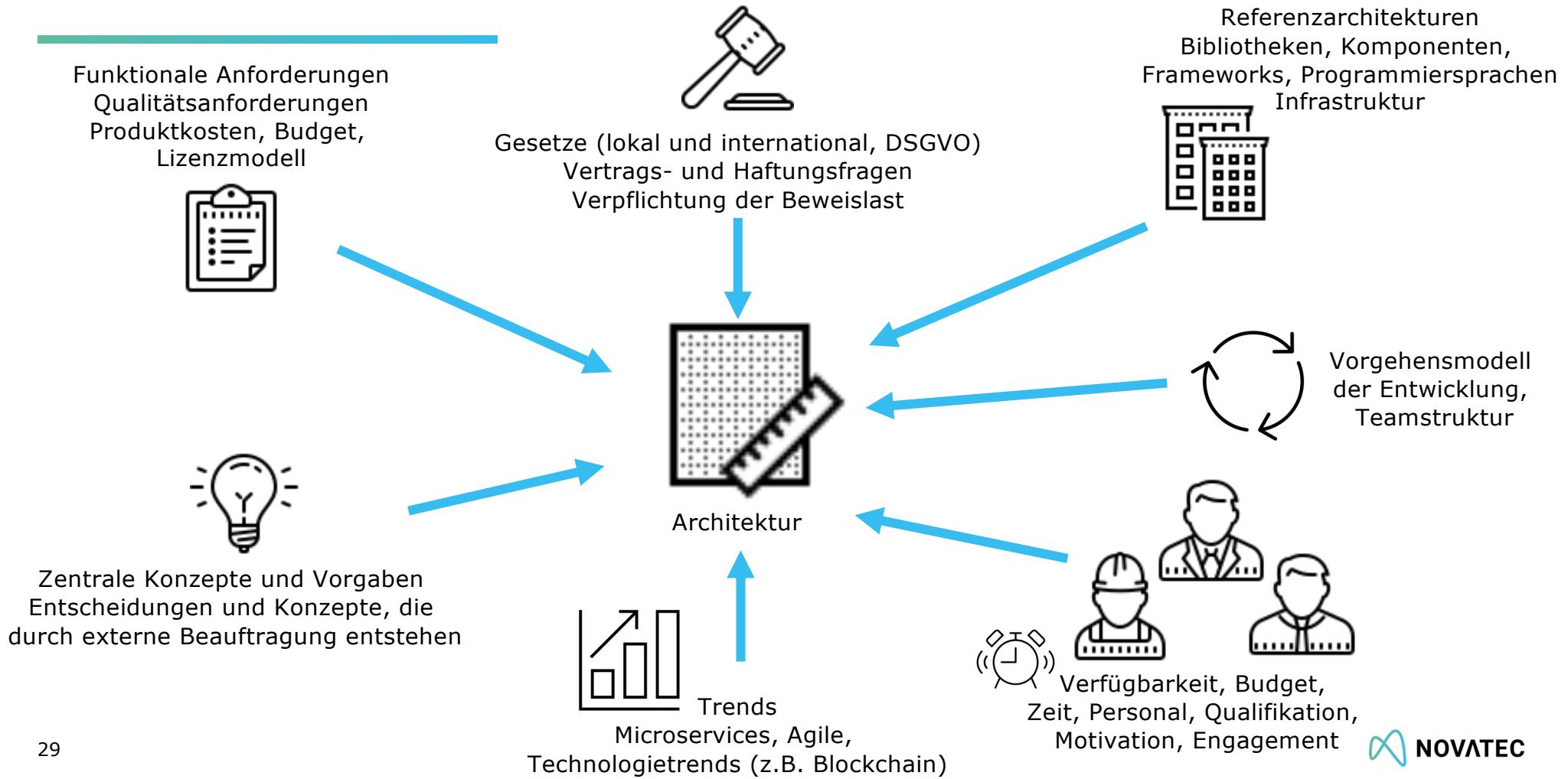
- regulatorischen Faktoren wie (R2)
 - lokale und internationale rechtliche Einschränkungen
 - Vertrags- und Haftungsfragen
 - Datenschutzgesetze und Gesetze zum Schutz der Privatsphäre
 - Fragen der Einhaltung oder Verpflichtungen zur Beweislast
- Trends wie (R3)
 - Markttrends
 - Technologietrends (z.B. Blockchain, Microservices)
 - Methodik-Trends (z.B. agil)
 - (potenzielle) Auswirkungen weiterer Stakeholderinteressen und vorgegebener oder extern festgelegter Designentscheidungen (R3)

Einflussfaktoren (Randbedingungen)

Einflussfaktoren können veränderbar oder nicht veränderbar sein und beeinflussen die Softwarearchitektur.

Diese **Faktoren** aus dem **Systemkontext**, schränken den architekturellen **Entscheidungsspielraum** ein.

Einflussfaktoren auf Architekturen



Einflussfaktoren auf Architekturen

Referenzarchitekturen und zentrale Konzepte sowie Vorgaben



Gesetze

Verschlüsselter Transport von personenbezogenen Daten



Konzepte
und
Vorgaben

Caching Verbot bei personenbezogenen Daten

RDBMS und REST Technologien sind gesetzt

Nicht veränderbar

Einflussfaktoren auf Architekturen

Qualitätsanforderung Verfügbarkeit



Qualitätsanforderungen

- Möglichkeiten der Plattform
(automatische Skalierung, Load Balancer)
- Architekturmuster
(Command Query Responsibility Segregation)
- Stabilitätsmuster
(Bulkhead, Circuit Breaker, Timeout)

Fehlendes Wissen

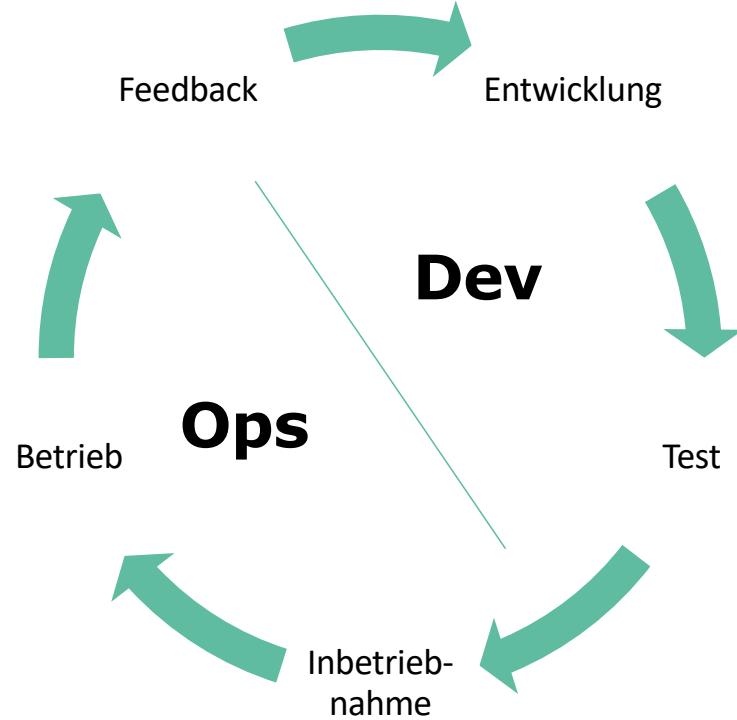


Dev Team

- Coaching
- Pair Programming
(Senior Junior Pair)
- Architektur Kata

Veränderbar

DevOps verzahnt die Entwicklung den Betrieb



- Einheitliche Entwicklungsvorgaben erleichtern den Betrieb
 - Programmiersprache
 - Frameworks
 - Schichtenarchitektur
 - Verwendung von Mustern

- Betriebsaspekte bereits bei der Entwicklung berücksichtigen (Logging und Monitoring)
 - Nachvollziehbarkeit von Fehlern
 - Fachliches und technisches Monitoring

LZ 1-3: Softwarearchitektur in Software-Lebenszyklus einordnen (R2)

- Softwarearchitekt:innen können Ihre Aufgaben und Ergebnisse in den gesamten Lebenszyklus von IT- Systemen einordnen. Sie können:
 - Konsequenzen von Änderungen bei funktionalen Anforderungen, Qualitätsanforderungen, Technologien oder der Systemumgebung im Bezug auf die Softwarearchitektur erkennen
 - inhaltliche Zusammenhänge zwischen IT-Systemen und den unterstützten Geschäfts- und Betriebsprozessen aufzeigen

Der Lebenszyklus von Software gemäß IEEE-Standard 12207:2017

Der IEEE-Standard 12207:2017 stellt ein gemeinsames Rahmenwerk für Software-Lebenszyklus-Prozesse mit definierten Begriffen zur Verfügung

Der Standard legt Prozesse fest, die wiederum aus Aktivitäten mit Aufgaben bestehen

Die Prozesse lassen sich grob in drei Kategorien unterteilen:

- Primäre Prozesse innerhalb des Lebenszyklus
- Unterstützende Prozesse innerhalb des Lebenszyklus
- Organisatorische Prozesse innerhalb des Lebenszyklus

Architekten nehmen in diesen Prozessen verschiedene Aufgaben wahr.

Quelle: <https://id3427.securedata.net/abelia/docs/12207cpt.pdf>

Die Aufgaben eines Architekten (Primäre Prozesse)

Primäre Prozesse	
Name	Aufgabe des Architekten (im Team) inkl. Umsetzung
Beschaffung	Produktauswahl
Entwurf	Anforderungsanalyse, Qualitätskriterien erarbeiten
Entwicklung	Anforderungsanalyse, Software-Design, Entwicklungsumgebung definieren, CI/CD-Pipeline definieren, Umsetzung, Architekturentscheidungen / Standards festlegen Test-Strategie (Testpyramide, Lasttest, Pentest) / Test
Betrieb	CI/CD-Pipeline definieren, Strategie für Monitoring, DevOps Mindset leben
Wartung und Pflege	Regelmäßige Softwareupdates sicherstellen, kontinuierliche Verbesserung vorantreiben, Behebung von Schwachstellen sicherstellen

Die Aufgaben eines Architekten (Unterstützende Prozesse)

Unterstützende Prozesse

Name	Aufgabe des Architekten (im Team) inkl. Umsetzung
Dokumentation	Pflege / Vervollständigung / Verbesserung der Dokumentation
Konfigurationsmanagement	Release Management Prozess mit Stakeholdern im Unternehmen definieren und abstimmen, Etablieren zentraler Repositories
Qualitätssicherung	Test-Driven Development / Codereview propagieren und durchführen, regelmäßige Last- und Pentests initiieren
Verifikation von Software	Automatisierte Ende-zu-Ende Tests, Codequalität in CI/CD Pipeline checken
Validierung der Software	Regelmäßige Nutzertests initiieren
Gemeinsame Prüfung (Auftraggeber / -nehmer)	Teilnahme / Präsentation und ggf. Moderation
Auditierung	Teilnahme
Problem-Management	Problemanalyse und -behebung, kontinuierliche Verbesserung initiieren

Die Aufgaben eines Architekten (Organisatorische Prozesse)

Organisatorische Prozesse

Name	Aufgabe des Architekten (im Team) inkl. Umsetzung
Management	Prozessverbesserung der Prozesse im Lebenszyklus in Boards / beim Management platzieren
Infrastruktur	Veränderungen / Verbesserung / Upgrades von Infrastrukturkomponenten propagieren
Verbesserung	Kontinuierliche Prozessverbesserung (beispielsweise gemäß agiler Werte - inspect & adapt) vorantreiben
Schulung	Eigene Aus- und Weiterbildung sowie im Team vorantreiben

LZ 1-4: Aufgaben und Verantwortung von Softwarearchitekten verstehen (R1) (1)

- Softwarearchitekt:innen tragen die Verantwortung für die Erreichung der geforderten oder notwendigen Qualität und die Erstellung des Architekturentwurfs der Lösung. Sie müssen diese Verantwortung, abhängig vom jeweiligen Prozess- oder Vorgehensmodell, mit der Gesamtverantwortung der Projektleitung oder anderen Rollen koordinieren.
- Aufgaben und Verantwortung von Softwarearchitekt:innen:
 - Anforderungen und Randbedingungen klären, hinterfragen und bei Bedarf verfeinern Hierzu zählen neben den funktionalen Anforderungen (Required Features) insbesondere die geforderten Qualitätsmerkmale (*Required Constraints*)

LZ 1-4: Aufgaben und Verantwortung von Softwarearchitekten verstehen (R1) (2)

- Strukturentscheidungen hinsichtlich Systemzerlegung und Bausteinstruktur treffen, dabei Abhängigkeiten und Schnittstellen zwischen den Bausteinen festlegen
- Querschnittskonzepte entscheiden (beispielsweise Persistenz, Kommunikation, GUI) und bei Bedarf umsetzen
- Softwarearchitektur auf Basis von Sichten, Architekturmustern sowie technischen und Querschnittskonzepten kommunizieren und dokumentieren
- Umsetzung und Implementierung der Architektur begleiten, Rückmeldungen der beteiligten Stakeholder bei Bedarf in die Architektur einarbeiten, Konsistenz von Quellcode und Softwarearchitektur prüfen und sicherstellen
- Softwarearchitektur analysieren und bewerten, insbesondere hinsichtlich Risiken bezüglich der geforderten Qualitätsmerkmale

LZ 1-4: Aufgaben und Verantwortung von Softwarearchitekten verstehen (R1) (3)

- Die Konsequenzen von Architekturentscheidungen erkennen, aufzeigen und gegenüber anderen Stakeholdern argumentieren
- Sie sollen selbständig die Notwendigkeit von Iterationen bei allen Aufgaben erkennen und Möglichkeiten für entsprechende Rückmeldung aufzeigen

LZ 1-6: Zusammenhang zwischen Entwicklungs- vorgehen und Softwarearchitektur erläutern (R1)

- Softwarearchitekt:innen können den Einfluss von iterativem Vorgehen auf Architekturentscheidungen erläutern (hinsichtlich Risiken und Prognostizierbarkeit).
- Sie müssen aufgrund inhärenter Unsicherheit oftmals iterativ arbeiten und entscheiden. Dabei müssen sie bei anderen Stakeholdern systematisch Rückmeldung einholen.

Anforderungen und Architektur – Das Twin Peaks Modell

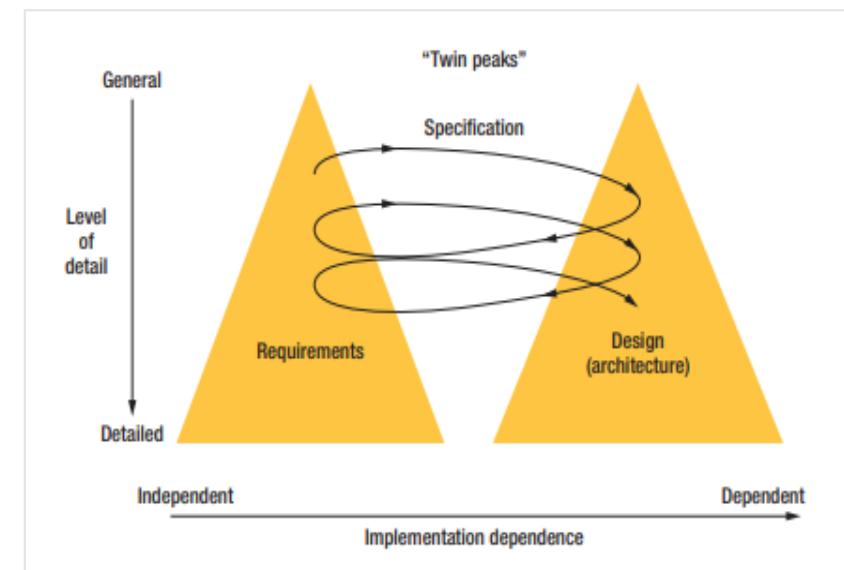
Sachverhalt:

- Anforderungen beeinflussen die Architektur des Systems
- Architektur beeinflusst den Aufwand zur Umsetzung von Anforderungen

Vorgehensweise:

- Die Ermittlung der Anforderung und der Architekturentwurf sollten verzahnt und parallel durchgeführt werden
- Iterative Detaillierung des Verständnisses über die Anforderungen und des Architekturentwurfs
- **Nur so viel Architektur wie notwendig!**

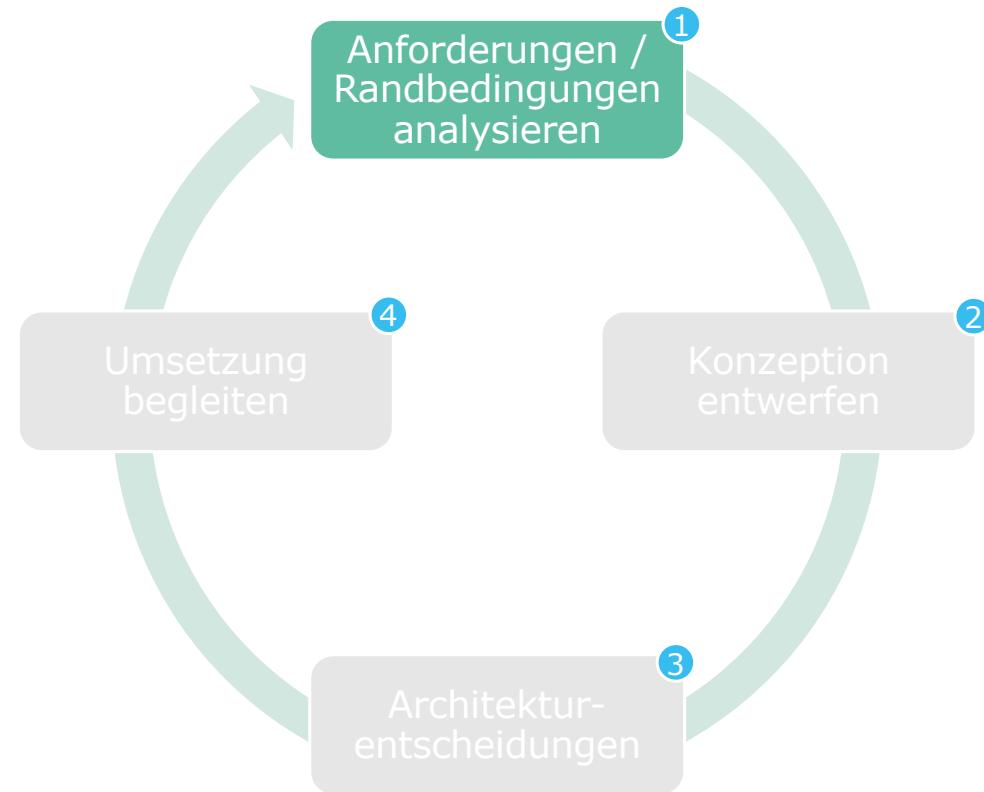
Twin-Peaks Modell



<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6470589>

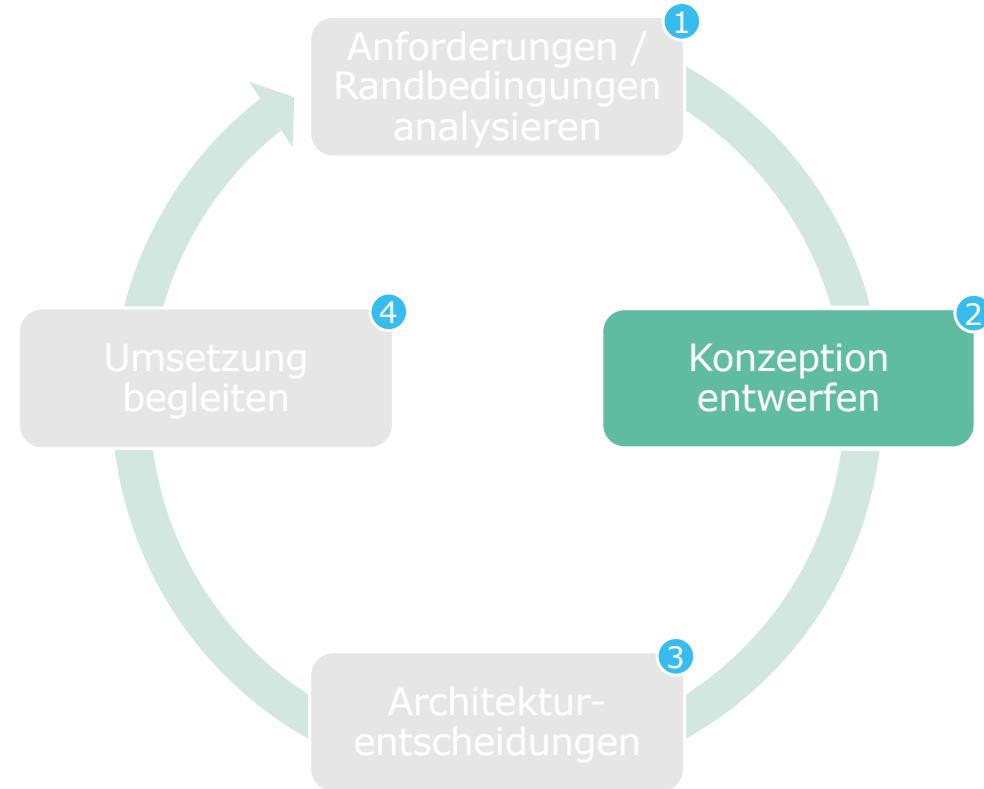
Aufgaben des Softwarearchitekten

- Funktionale und nicht funktionale Anforderungen / Randbedingungen verstehen und hinterfragen
- Qualität / Flexibilität / Änderbarkeit der Anforderungen prüfen
- Unklarheiten / Lücken identifizieren
- Gemeinsames erstes Verständnis für Architekturstil und technische Infrastruktur entwickeln
- Softwareentwicklungsprozess aufsetzen / erarbeiten
- *Tipp:* Verzicht Anforderungen umzusetzen (Anforderungen kritisch hinterfragen!)



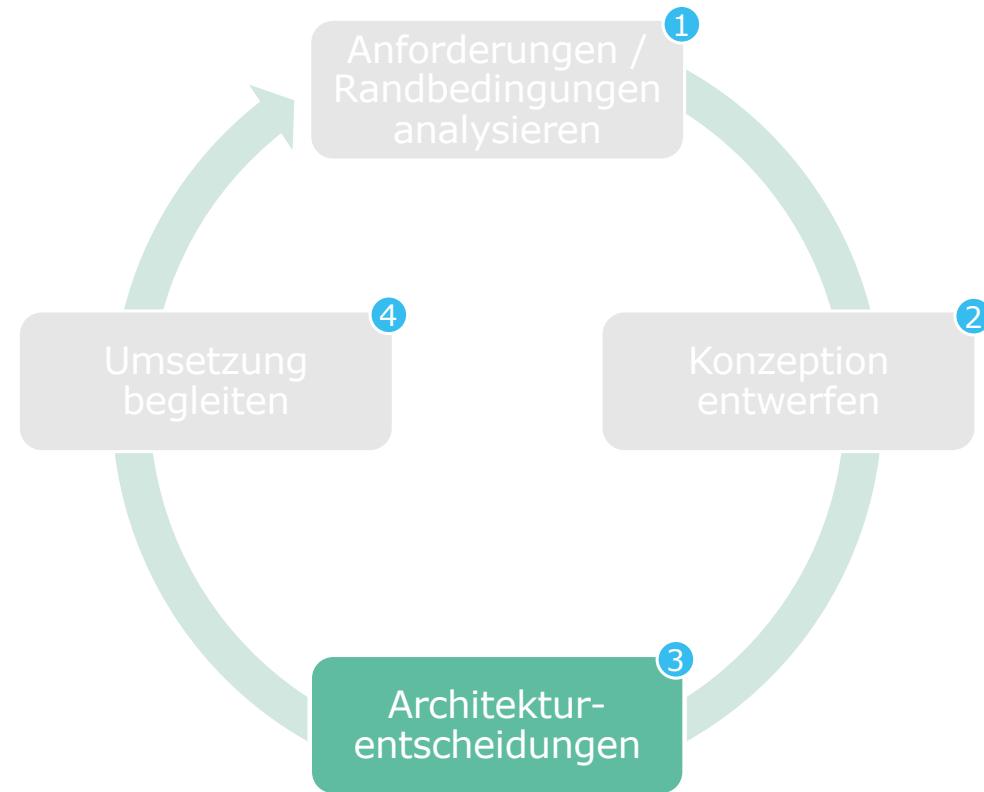
Aufgaben des Softwarearchitekten

- Detaillierung der Architektur
- Erstellung von Sichten zur Beschreibung der unterschiedlichen Architekturebenen
- Konzeption von Lösungsbausteinen der technischen Architekturebene
- Dokumentation der Architektur
- Hierbei muss der Lösungsrahmen / Architekturstil und die technische Infrastruktur beachtet werden



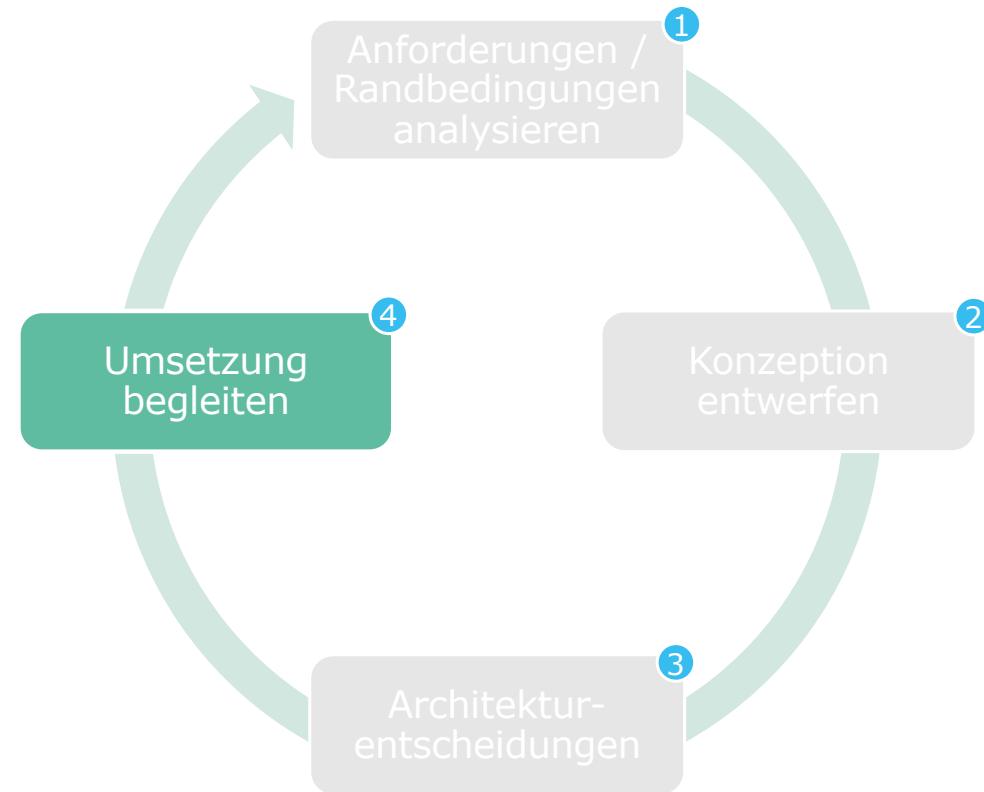
Aufgaben des Softwarearchitekten

- Qualitätssicherung des Entwurfs
(Review / Prototypen / Tests)
- Dokumentation der Architekturentscheidung
- Bewertung der Architektur
(siehe Kapitel 4)



Aufgaben des Softwarearchitekten

- Kommunizieren der Architektur an alle Projektbeteiligten, insbesondere der Stakeholder
- Jede Person muss die Architektur verstehen und akzeptieren, nur so gelingt die erfolgreiche Umsetzung
- Der Detailgrad der Kommunikation hängt vom Gegenüber ab
- Kommunikation ist keine Einbahnstraße, sondern ein gemeinsames Verstehen und Voneinander-Lernen
- Es werden nicht nur offene Fragen, sondern auch Verbesserungspotenziale, Fehler und Weiterentwicklungen identifiziert



Die Aufgaben des Architekten

Auf Basis der **Anforderungen** (funktional und nicht-funktional), unter Berücksichtigung der **Randbedingungen** und der **Systemumgebung** (Schnittstellen zu Umsystemen) wird der **Bauplan** und der **Konstruktionsweg** für das System entwickelt

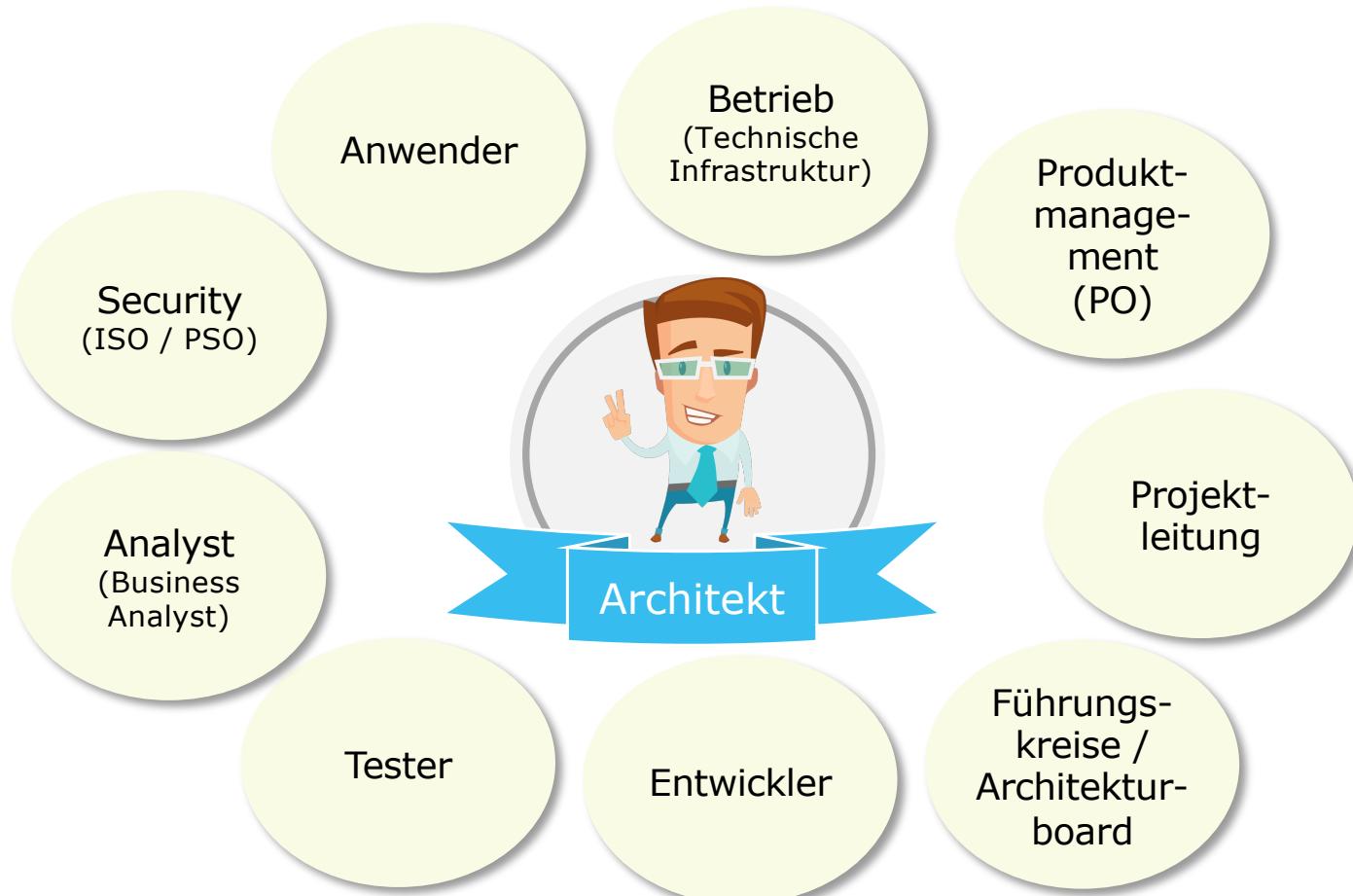
Dazu gehört auch ein Blick auf Zukunftsthemen hinsichtlich **Erweiterungen** und **Änderungen**

Die Architekturarbeit ist eine Kommunikations- und Diskussionsplattform mit dem Ergebnis des Design- und Implementierungsplans.

LZ 1-5: Rolle von Softwarearchitekt:innen in Beziehung zu anderen Stakeholdern setzen (R1)

- Softwarearchitekten können ihre Rolle erklären und ihren Beitrag zur Systementwicklung in Verbindung mit anderen Stakeholdern kontextspezifisch ausgestalten, insbesondere zu:
 - Produktmanagement, Product-Owner
 - Projektleitung und -management
 - Anforderungsanalytiker:innen (System-/Businessanalyse, Anforderungsmanagement, Fachbereich)
 - Entwicklung
 - Qualitätssicherung und Test
 - IT-Betrieb (Produktion, Rechenzentren), zutreffend primär für Informationssysteme
 - Hardwareentwicklung
 - Unternehmensarchitektur, Architekturboard

Der Architekt in Bezug zu anderen Rollen



Der Architekt in Bezug zu anderen Rollen

Architekt ↔ PL / Analyst / Anwender

- Machbarkeit
- Identifikation von Widersprüchen
- Aufzeigen von Möglichkeiten & Alternative sowie Evaluation von z.B. Integration bestehender Software
- Unterstützung bei Projekt- und Iterationsplanung
- Wirkt mit bei Risikoanalyse und -vermeidung

Architekt ↔ Tester

- Testrahmenbedingungen
- Testreihenfolge und Abhängigkeiten (Mocks)
- Testfälle (insbesondere zur Überprüfung von Architekturzielen)

Architekt ↔ Betrieb / Security

- Zentraler Ansprechpartner aus Sicht der Unternehmensorganisation
- Diskussion übergreifender Themen

Der Architekt in Bezug zu anderen Rollen

Architekt ↔ Produktmanagement (Product Owner)

- Technische Implikation von Anforderungen (Vor- / Nachteile) aufzeigen
- Verständnis für Spikes (Prototypen) vermitteln
- Fachliche versus technische Weiterentwicklung priorisieren
- Nicht-funktionale Anforderungen sicherstellen
- Enge Zusammenarbeit / Teaming

Architekt ↔ Führungskreis / Architekturboard

- Projektvorstellung
- Architekturvorschlag und Freigabe
- Schätzungen in Bezug auf das notwendige Budget vornehmen
- Unternehmensarchitektur, Architekturboard, Projektboard, etc.
- Evaluierung / Einführung neuer Technologien

Der Architekt in Bezug zu anderen Rollen

Architekt ↔ Entwickler

- Ist zentraler Ansprechpartner
- Erklärt die Architektur, die definierten Bausteine sowie deren Schnittstellen und Interaktionsmuster
- Leitung der Erarbeitung, Einführung, Schulung und Überprüfung von Programmierrichtlinien
- Prototyping und Entwicklung exemplarischer Teillösungen
- Erkennung und Realisierung von Wiederverwendungspotenzialen
- Erfahrungsweitergabe und Coaching
- Treiber innovativer Lösungen sowie Nutzung innovativer Technologien

Fähigkeiten eines Softwarearchitekten

- Teamarbeit, Teamführung (nicht hierarchisch) und Coaching
- Kommunikationsfähig und redegewandt
- Gut in der Argumentation und im Erklären
- Kann überzeugen und sich durchsetzen
- In der Lage unterschiedliche Expertisen und Potenziale im Entwicklungsteam zu nutzen
- Kann sich auf unterschiedliche Stakeholder und einzelnen Menschen einstellen und hineinversetzen
- Abstraktionsfähigkeit
- Technisches und fachliches Verständnis in seiner Domäne
- Ist ein sehr guter Entwickler
- Hat Erfahrungen mit unterschiedlichen Architekturen

Ein Softwarearchitekt ist auch...

...Anwalt der Kunden und Berater für Manager

*... **Diplomat** und **Akrobat***

*...ein **Vereinfacher***

*...derjenige, der den **Mut** hat eine **Entscheidung** zum richtigen Zeitpunkt **zu treffen***

*...jemand der **konstruiert, entwirft** und **implementiert***

*...ein **Kommunikator, Qualitätssicherer** und **Coach***

Lernziel 1 - Übung – 20 Minuten

- Sie müssen eine Aufzugsstory ("Elevator Pitch") für die Vorstellung der Architektur-Abteilung erstellen. Dafür haben Sie 5 – 10 Sätze zur Verfügung. Bitte erklären Sie dabei auch:
 - Was ist Softwarearchitektur und was ist wichtig bei der Erstellung?
 - Wieso braucht es überhaupt ein Architekturteam?

Ein **Elevator Pitch** ist eine Methode für eine kurze Zusammenfassung einer Idee. Der Fokus liegt auf positiven Aspekten wie zum Beispiel der Einzigartigkeit.

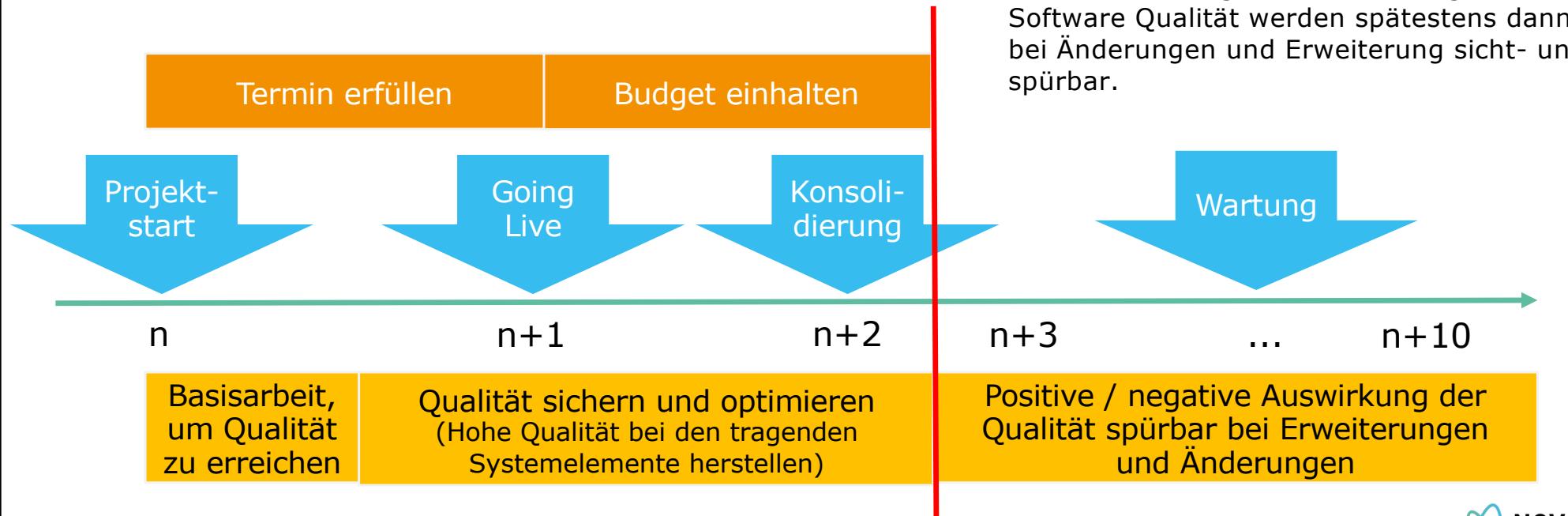
Der Kerngedanke eines „Elevator Pitches“ basiert auf dem Szenario, eine wichtige Person in einem Aufzug zu treffen und diese dann während der Dauer einer Aufzugsfahrt von einer Idee zu überzeugen. Ist die Idee überzeugend genug vorgestellt worden, wird das Gespräch weitergeführt oder man verabredet sich zu einem weiterführenden Meeting. Ziel ist es, positiv im Gedächtnis zu bleiben.

LZ 1-7: Kurz- und langfristige Ziele differenzieren (R1)

- Softwarearchitekt:innen können:
 - langfristige Qualitätsanforderungen sowie deren Abgrenzung gegen (kurzfristige) Projektziele erklären
 - Potentielle Zielkonflikte zwischen kurz- und langfristigen Zielen erklären, um eine für alle Beteiligten tragfähige Lösung zu erarbeiten
 - Qualitätsanforderungen identifizieren und präzisieren

Langfristige Architekturziele vs. kurzfristige Projektziele

Qualität vs. Budget / Timeline



Langfristige Architekturziele vs. kurzfristige Projektziele

Als Architekt möchte ich Entscheidung oft spät treffen um ausreichend wissen für eine zukunftsfähige Entscheidung zu sammeln (Entscheidung im letzten vernünftigen Moment)

Zeitpunkt der Entscheidung

Vollständigkeit der Informationen

Anforderungen sind nicht ausreichen beschrieben, Informationen über den Systemkontext nur geringfügig vorhanden. Basis für Aufwandsschätzung ist eigentlich nicht gegeben.

Zuverlässige Aufwandsschätzung

Als Projektleiter möchte ich in einer frühen Projektphase eine zuverlässige und belastbare Aufwandsschätzung.

Langfristige Architekturziele vs. kurzfristige Projektziele - Beispiele

Projektziele	Architekturziele
Schnelle Auslieferung von Features / Inkrementen	Investition in Schnittstellen-Homogenität des Systems
Time-to-market	Investition in funktionale Stabilität sichergestellt durch Tests
Time-to-market	Investition zur Ausschöpfung von Wiederverwendungspotenzial

LZ 1-8: Explizite von impliziten Aussagen unterscheiden (R1)

- Softwarearchitekt:innen:
 - können Annahmen oder Voraussetzungen explizit darstellen und vermeiden implizite Annahmen
 - wissen, dass implizite Annahmen potentielle Missverständnisse zwischen beteiligten Stakeholdern bewirken
 - können implizit formulieren, wenn es im gegebenen Kontext angemessen ist

Die Basis für Software Qualität legen...

Architekten müssen **implizite Annahmen**, die zwischen den Zeilen von Anforderungsbeschreibungen versteckt sind oder automatisch vorausgesetzt werden **explizit machen!**

- Ansonsten sind Anforderungen unbekannt / unklar und werden in der Architektur **nicht** oder **falsch** berücksichtigt. Dies wirkt sich langfristig **negativ** auf die Architektur und dessen **Qualität** aus
- Weiter sind damit **Missverständnisse** zwischen Projektakteuren und Stakeholdern verbunden

Explizit machen bedeutet das Thema **greifbar machen**, so dass dies umgesetzt werden kann und im Fall von Qualitätszielen / Architekturzielen das Ergebnis messbar ist

Randbedingungen sind oft implizit vorhanden und sollten möglichst explizit gemacht werden (z.B. vorgegebenes DBMS)

Implizite Formulierungen können u.U. für Aspekte wie Testbarkeit (Testabdeckung durch Unit-Tests), Sicherheit (Verschlüsselung) oder Überwachung (Logging) verwendet werden.

Softwarearchitektur als zentraler Wert

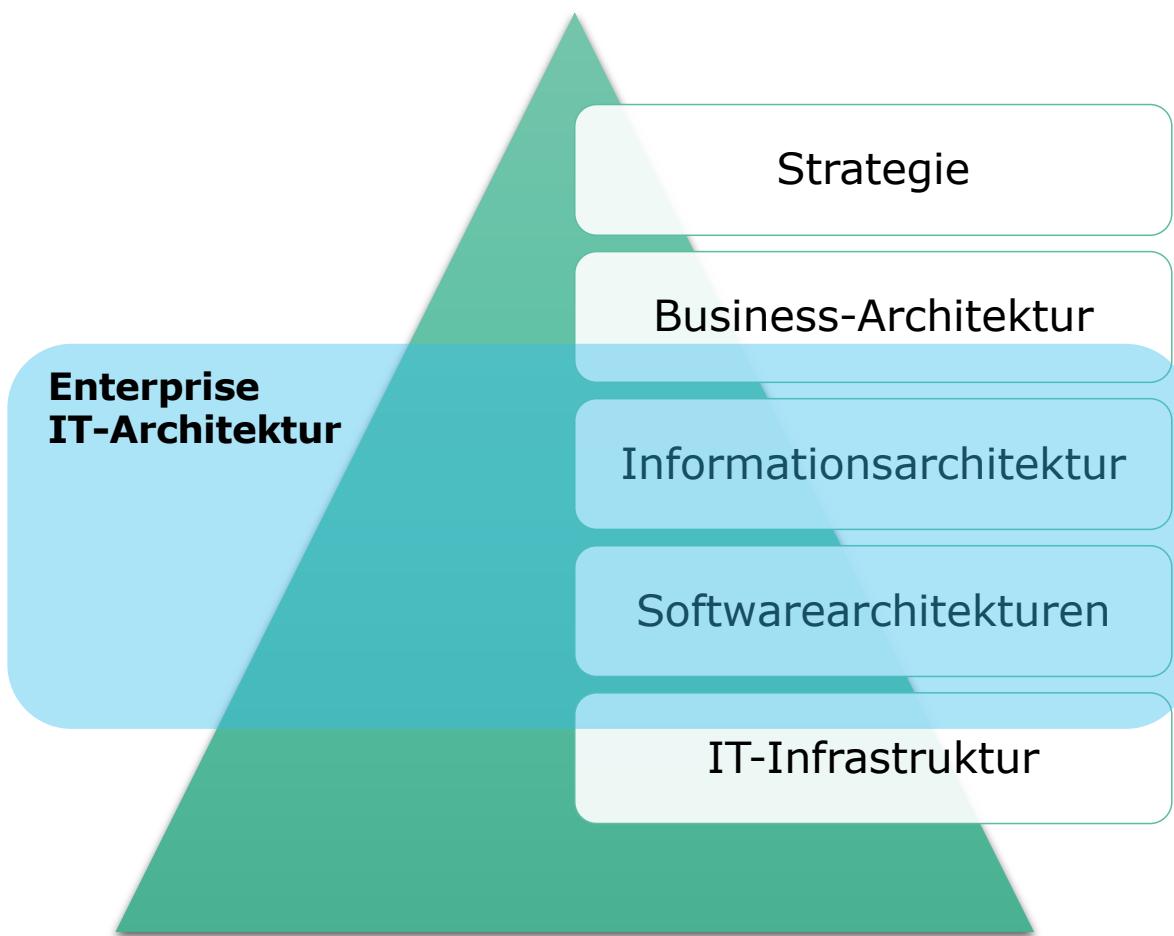
- Vertiefung erfolgt im Kapitel 4 „Softwarearchitekturen und Qualität“
- Architektur definiert Strukturen und in diesem Sinne auch strukturelle Qualität
- Ist ein System auf Langlebigkeit ausgelegt wird Wartbarkeit, Erweiterbarkeit und Verständlichkeit relevant
- Diese Qualitätsziele / Architekturziele werden selten von den Stakeholdern explizit formuliert
- Der Architekt muss Ziele explizit machen, z.B. durch Qualitätsszenarien

Hinweis: Es gibt auch Systeme, die für eine kurze Zeit laufen und sehr schnell sein müssen. In diesem Fall ist das Ziel sich auf Effizienz (Performance) zu fokussieren und dies explizit zu machen.

LZ 1-9: Zuständigkeit von Softwarearchitekten in organisatorischen Kontext einordnen (R3)

- Der Fokus des iSAQB CPSA-Foundation Level liegt auf Strukturen und Konzepten einzelner Softwaresysteme.
- Darüber hinaus kennen Softwarearchitekten weitere Architekturdomänen, z.B.:
 - Unternehmens-IT-Architektur (*Enterprise IT Architecture*): Struktur von Anwendungslandschaften
 - Geschäfts- bzw. Prozessarchitektur (*Business and Process Architecture*): Struktur von u.a. Geschäftsprozessen
 - Informationsarchitektur: systemübergreifende Struktur und Nutzung von Information und Daten
 - Infrastruktur- bzw. Technologiearchitektur: Struktur der technischen Infrastruktur, Hardware, Netze etc.
 - Hardware- oder Prozessorarchitektur (für hardwarenahe Systeme).
- Diese Architekturdomänen sind nicht inhaltlicher Fokus vom CPSA-F.

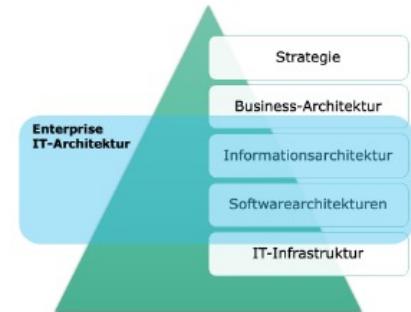
Architekturpyramide von Dern & Starke



Architekturpyramide von Dern & Starke

Strategie

- Ziele der Organisation und mögliche Wege diese zu erreichen (Unternehmensstrategie, strategisches Management)
- Alignment zwischen Business und IT auf Basis der Business-Architektur



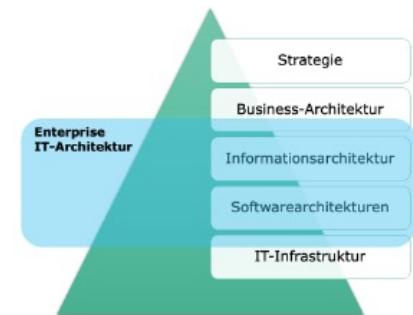
Business-Architektur

- Geschäftsprozesse (Geschäftsprozessarchitektur) und betriebliche Funktionen
- Begriff „Information“ wird im rein fachlichen Sinn (ohne Bezug auf IT-Systeme) verwendet
- Z.B. Prozess einer Schadensmeldung innerhalb eines Versicherers oder auch zwischen Versicherer, Gutachter und Werkstatt

Architekturpyramide von Dern & Starke

Informationsarchitektur

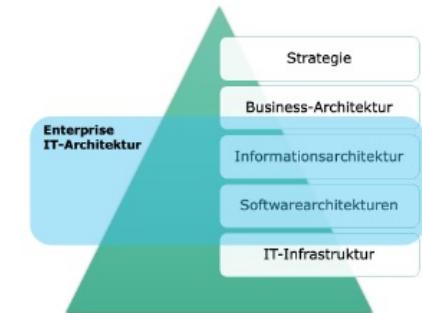
- Struktur und Zusammenarbeit aller IT-Systeme (auch IS-Portfolio / Anwendungslandschaft genannt)
- Betrachten Schnittstellen und Informationsflüsse zwischen den IT-Systemen
- Hauptaufgabengebiet Enterprise Architekt
- Vorgabe von Referenzarchitekturen (Blueprints)
- Architektonische Verwendung von Middleware (ESB, MOM, Gateway) und Datenbanken
- Open-Source-Strategie
- Inner-Sourcing-Strategie und Wiederverwendungsstrategie
- IT-Innovation



Architekturpyramide von Dern & Starke

Softwarearchitektur

- Entwurf und Implementierung von Softwaresystemen
- Ein System ist ein Baustein in der Informationsarchitektur
- **Alles was im CPSA-F Kurs behandelt wird**



IT-Infrastruktur

- Hardwarearchitektur
- Netztopologie
- Betriebssysteme
- Administration Datenbanksysteme (Backup & Recovery)
- Administration Middleware-Systeme

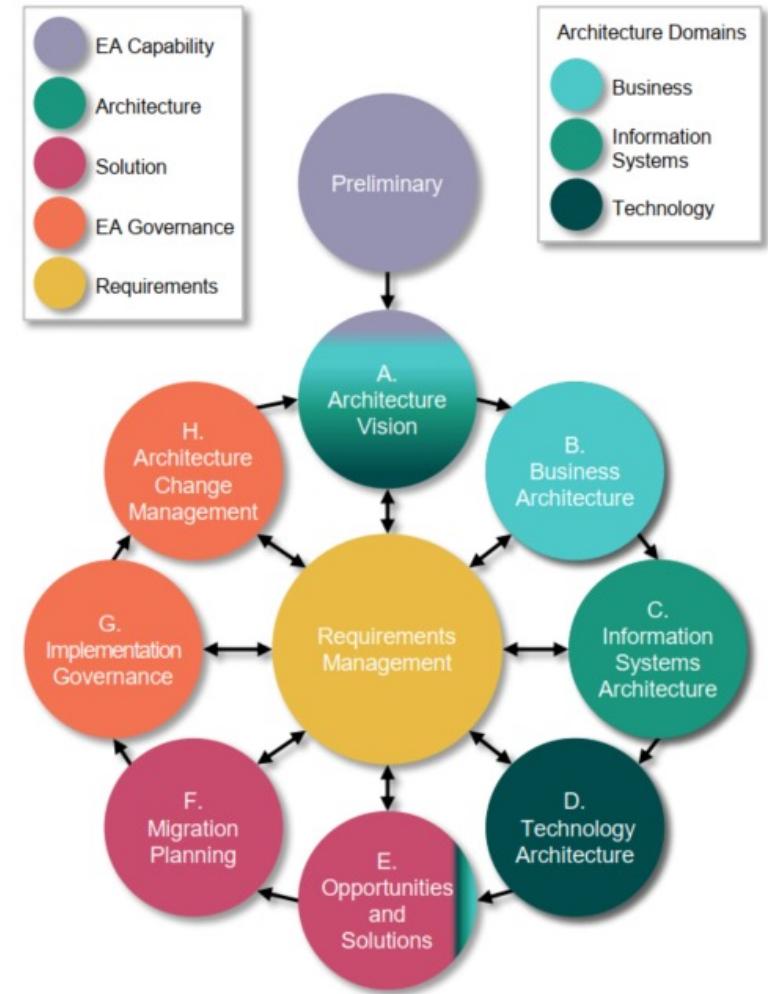
Wieso diese verschiedenen Architekturebenen?

Ziele dieser Architekturebenen im Zusammenhang mit einer Enterprise IT-Architektur:

- Business-IT-Alignment über Strategie, Geschäftsprozesse, Prozessunterstützung und –automatisierung durch IT
- Komplexitätsmanagement in einer Organisation
- Komplexitätsmanagement großer Anwendungslandschaften
Technische Standardisierung und Referenzarchitekturen
- Einführung von Ebenen, die die Rolle eines Vermittlers spielen
Informationsarchitektur vermittelt zwischen Business Architektur und IT-Sicht sowie Softwarearchitektur

Enterprise Architektur Beispiel

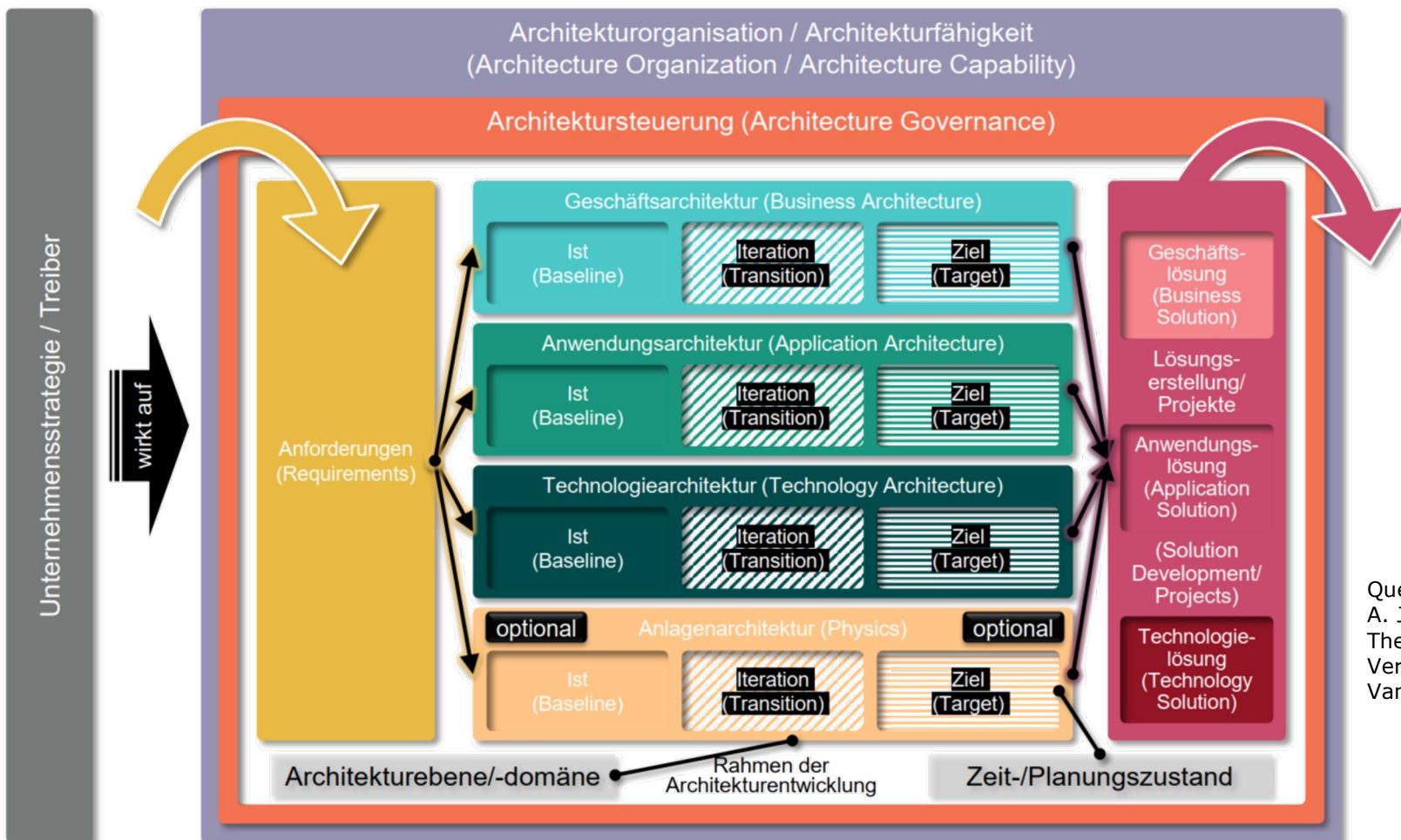
- The Open Group Architecture Framework (TOGAF)
- Umfassendes Vorgehensmodell



Quelle: A. Josey: The TOGAF ® Standard, Version 9.2
A Pocket Guide, Van Haren Publishing, April 2018

Architecture Development Method (ADM)

Enterprise Architektur Beispiel



Quelle:
A. Josey
The TOGAF ® Standard
Version 9.2 - A Pocket Guide
Van Haren Publishing, April 2018

LZ 1-10: Typen von IT-Systemen unterscheiden (R3)

- Softwarearchitekt:innen kennen unterschiedliche Typen von IT-Systemen, beispielsweise:
 - Informationssysteme
 - Decision-Support, Data-Warehouse oder Business-Intelligence Systeme
 - Mobile Systeme
 - Batchprozesse oder -systeme
 - hardwarenahe Systeme; hier verstehen sie die Notwendigkeit des Hardware-/Software-Codesigns (zeitliche und inhaltliche Abhangigkeiten von Hard- und Softwareentwurf).

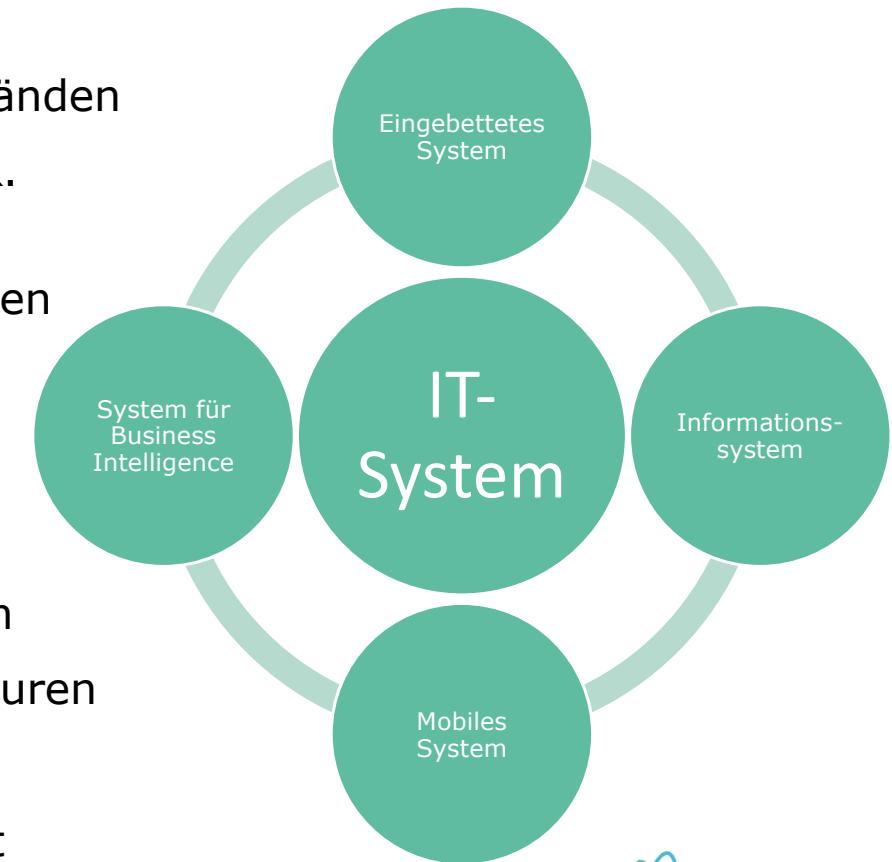
Typen von IT-Systemen

Eingebettetes System

- Software eingebettet in physikalischen Gegenständen
- Starke Abhängigkeit von der Hardware mit i.d.R. knappen Ressourcen
- Daten- und funktionssicherheitskritische Aufgaben
Regelungs-, Steuerungs- und Kommunikationsfunktionen
- Airbag-Steuerung, Wasch- oder Werkzeugmaschinen

Informationssystem

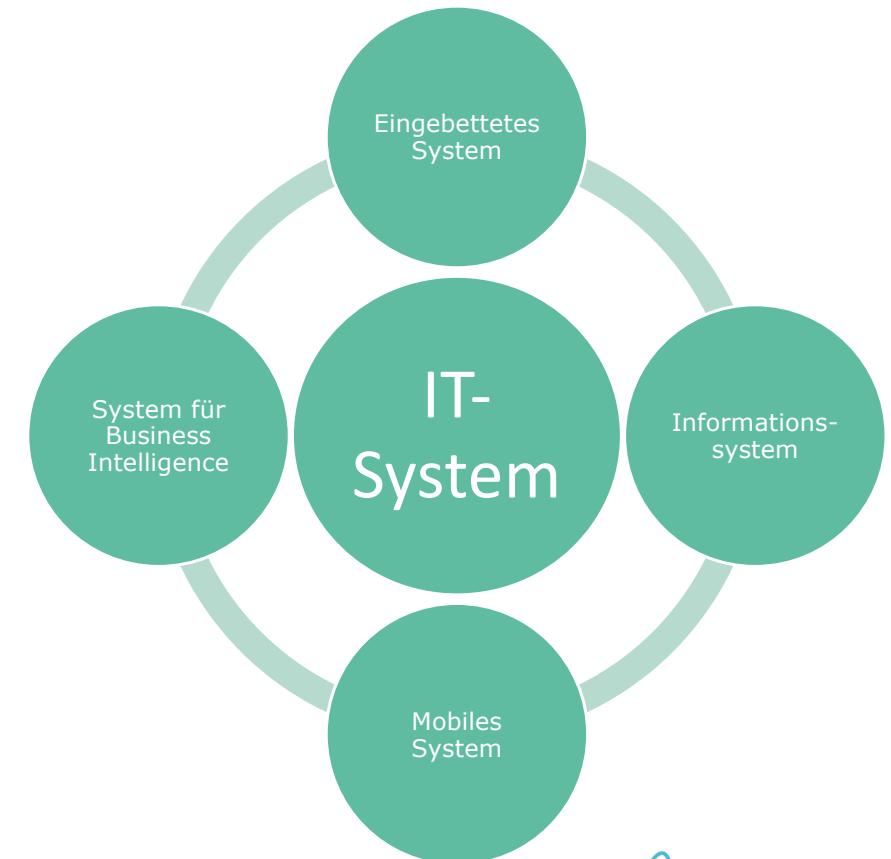
- Verwaltung und Verarbeitung von Informationen
- Große Datenmengen und komplexe Datenstrukturen
- Komplizierte Kalkulationen und Auswertungen
- ERP, CRM, Waren- oder Versicherungswirtschaft



Typen von IT-Systemen

Mobiles System

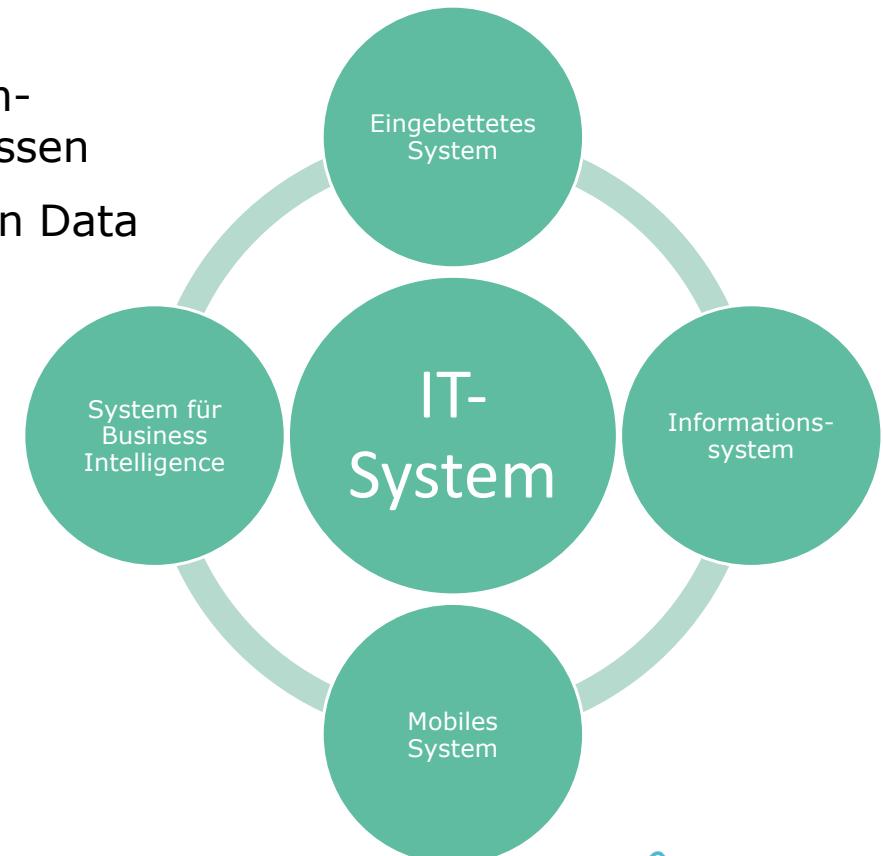
- (Semi-)autonome und personenspezifische Einheiten
- Bereitstellung lokaler und (semi-)autonomer Funktionen
- Software auf mobilem Gerät, i.d.R. im Austausch mit stationären Systemen
- Smartphones, (semi-) autonome Transportroboter, Sensor- und Aktuatorknoten von Ad-hoc-Netzwerken



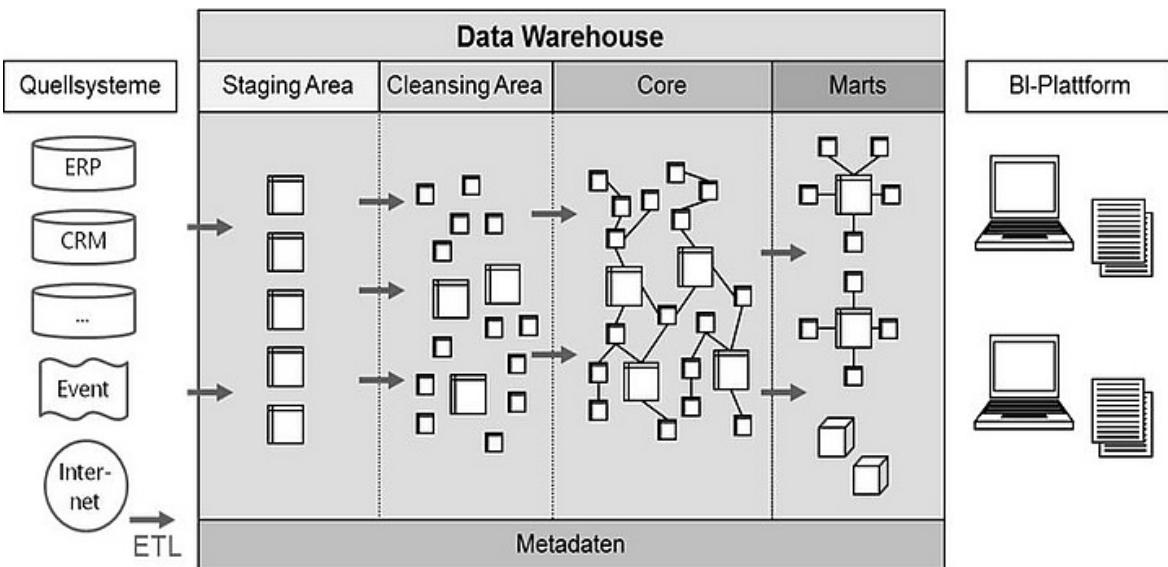
Typen von IT-Systemen

Business Intelligence

- Business Intelligence (BI) ist der Prozess der Umwandlung von Daten in Informationen und in Wissen
- Die Grundlage von Business Intelligence stellt ein Data Warehouse (DWH) dar, in dem operative Daten aggregiert werden.



Die Architektur eines Datawarehouse



Schritte:

1. Extraktion der operativen Daten (ETL) aus den Quellsystemen
2. Bereinigung der Daten und Überführung in einheitliche Zielstruktur
3. Integration der Daten in ein einheitliches Datawarehouse Modell
4. Teile der Daten mit fachlichem Fokus in Data Marts Antwortzeit-optimiert aufbereitet

Quelle: <https://www.informatik-aktuell.de/entwicklung/methoden/data-warehouse-im-wandel.html>

Grundmuster und Eigenschaften der IT-Systeme

Die Grenzen zwischen den Typen sind teilweise fließend. Gefördert durch die zunehmende Vernetzung sind Systeme von vielen Endgeräten und / oder von Fahrzeugen aus bedienbar.

So kann ein System ein **Informationssystem** und ein **Mobiles System** sein.

Systemtyp	Eigenschaften
Informationssystem	<ul style="list-style-type: none">▪ Verwaltung und die Verarbeitung von Informationen stehen im Vordergrund▪ Große Datenmengen▪ Hohe Parallelität
Eingebettetes System	<ul style="list-style-type: none">▪ Starke Ressourceneinschränkungen▪ Hohe funktionale und qualitative Anforderungen
Mobiles System	<ul style="list-style-type: none">▪ Online und Offline Funktionalität / Datenabgleich▪ Hohe Interaktionsanforderungen
Business Intelligence	<ul style="list-style-type: none">▪ Ableitung von Wissen / Trends aus Fakten▪ Verknüpfung der einzelnen Daten / zeitliche Abfolge

Softwarearchitektur und Hardwarearchitektur

Systemarchitektur = Softwarearchitektur + Hardwarearchitektur

Die **Softwarearchitektur** wird von der **Hardwarearchitektur beeinflusst**, insbesondere hinsichtlich der **Qualitätsanforderungen** Effizienz, Hochverfügbarkeit und Sicherheit.

Besonders kritisch ist dies bei eingebetteten Systemen, da hier nur sehr eingeschränkte Ressourcen zur Verfügung stehen.

LZ 1-11: Herausforderungen verteilter Systeme (R3)

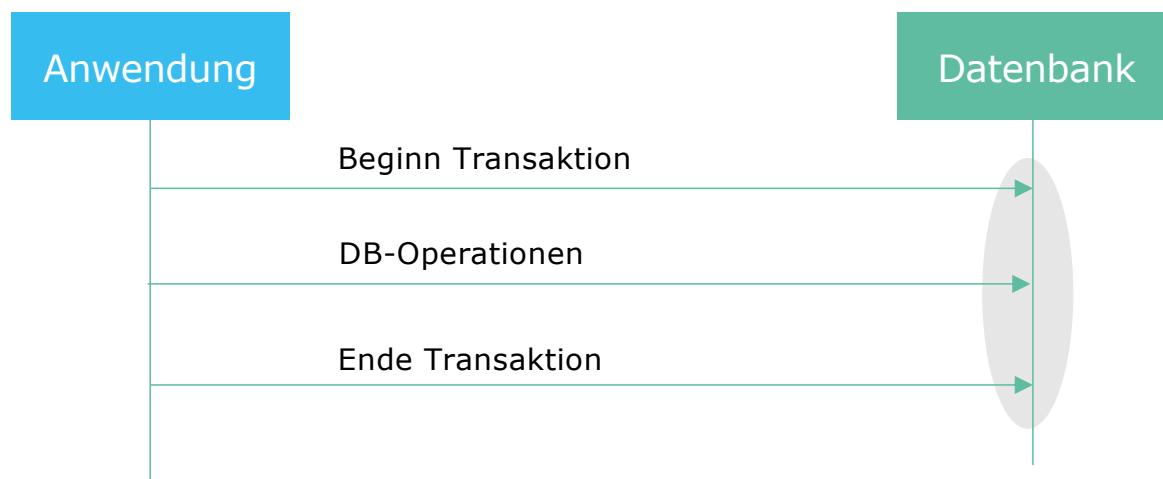
- Softwarearchitekt:innen können:
 - die Verteilung in einer gegebenen Software-Architektur identifizieren
 - Konsistenzkriterien für ein gegebenes fachliches Problem analysieren
 - Kausalität von Ereignissen in einem verteilten System erklären
- Softwarearchitekt:innen wissen:
 - dass Kommunikation in einem verteilten System fehlschlagen kann
 - dass es bei verteilten Systemen Einschränkungen hinsichtlich der Konsistenz in Datenbanken gibt
 - was das "Split-Brain"-Problem ist und warum es schwierig zu lösen ist
 - dass es unmöglich ist, die exakte zeitliche Reihenfolge der Ereignisse in einem verteilten System zu bestimmen

Was versteht man unter einem verteilten System?

- Andrew S. Tanenbaums Definition eines verteilten Systems: *Ein verteiltes System ist ein Zusammenschluss unabhängiger Computer, die sich für den Benutzer als ein einziges System präsentieren.*
- Gründe für den Einsatz verteilter Systeme:
 - Skalierbarkeit
 - Redundanz
 - Lastverteilung
- Probleme bei verteilten Systemen:
 - Konsistenz der Verarbeitung
 - (Teil-)Ausfälle
 - Timeouts / Erreichbarkeit (unklare Verhältnisse ob eine Verarbeitung stattgefunden hat).

Datenbankzugriffe werden typischerweise über ACID-Transaktionen durchgeführt

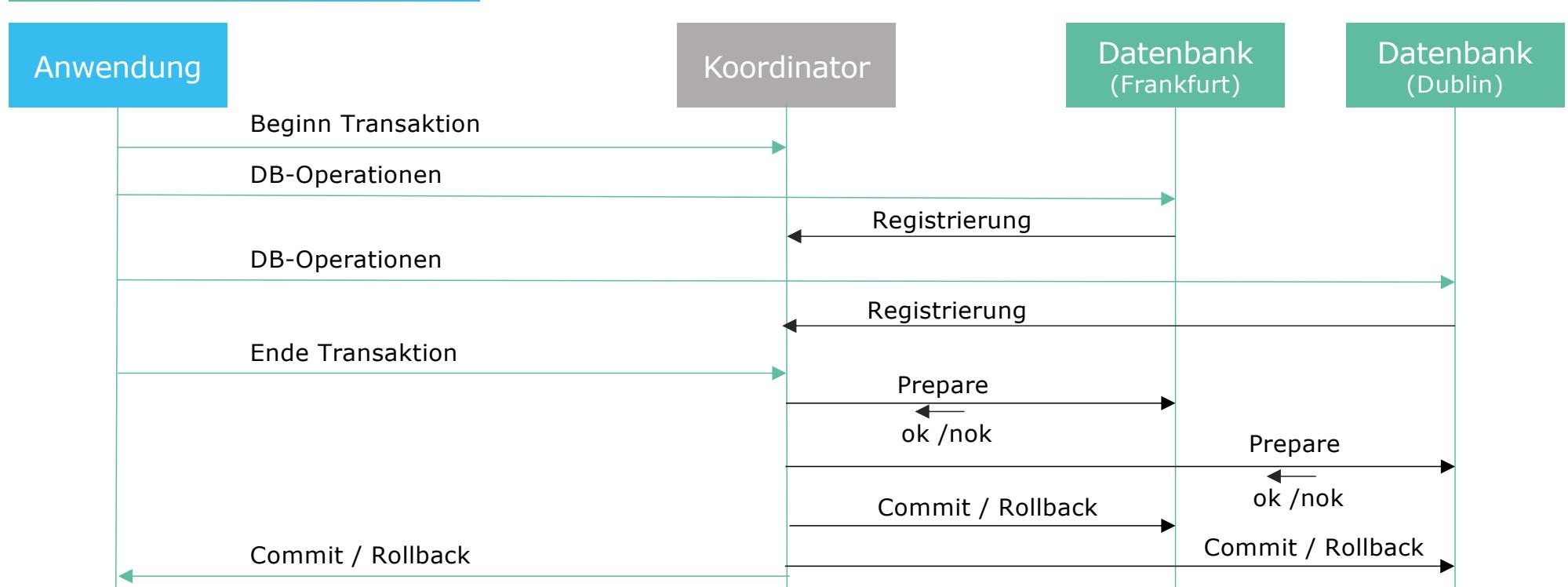
Konsistenz in Datenbanken durch ACID-Transaktionen



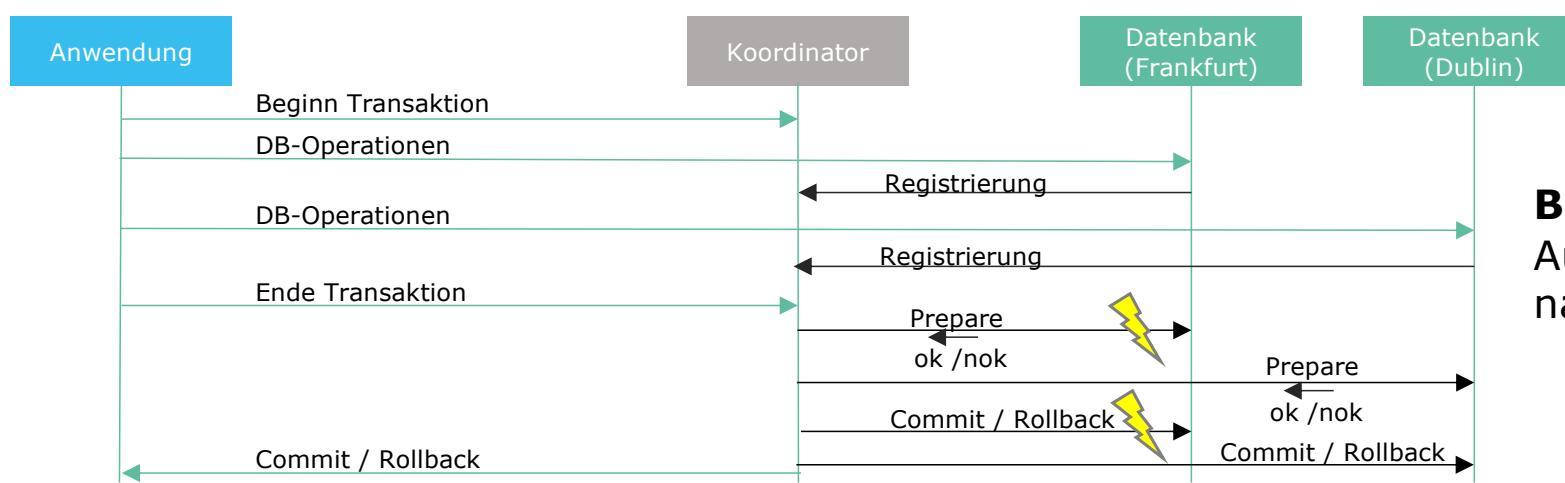
Single unit of work:

- **Atomic** (ganz oder gar nicht)
- **Consistent** (Datenkonsistenz wird gewährleistet)
- **Isolated** (Zwischenzustände sind für nebenläufige Aktivitäten nicht sichtbar)
- **Durable** (Änderungen überleben einen Systemabsturz)

Für verteilte Transaktion wird ein externer Koordinator (Transaktionsmonitor) benötigt



Das sogenannte Split Brain Problem tritt auf wenn eine Netzwerkverbindung ausfällt

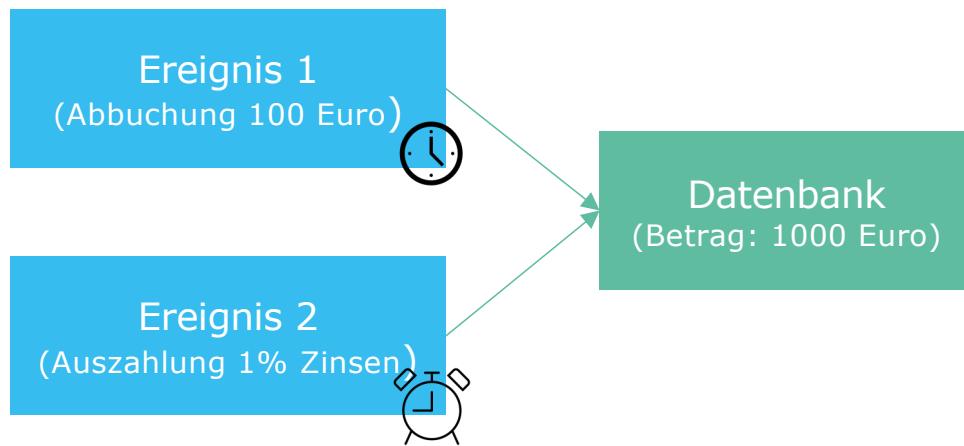


Beispielszenario:
Ausfall der Verbindung
nach Frankfurt

- Das Ergebnis der Operation Prepare ist unklar
- Das Ergebnis der Operation Commit / Rollback ist unklar

Fazit: Die Transaktion verbleibt im Status *in doubt* bis die Verbindung wieder verfügbar ist.

In verteilten Systemen ist es problematisch die Reihenfolge von Ereignissen zu bestimmen



Situation:

Mögliche Reihenfolge

- E1 vor E2: Betrag: 909 Euro
- E2 vor E1: Betrag: 910 Euro

Systemuhren besitzen leicht abweichende lokale Zeit

Es existiert kein globaler Zähler für Ereignisse

Fazit:

- Es ist generell unmöglich, physikalische Uhren in einem verteilten System absolut zu synchronisieren
- Damit ist es auch nicht möglich, basierend auf diesem System die Reihenfolge zweier beliebiger Ereignisse zu bestimmen.

Quellen

[Gharbi 2018] Gharbi, Mahbouba [u.a]: Basiswissen für Softwarearchitekten. Aus- und Weiterbildung nach iSAQB-Standard zum Certified Professional for Software Architecture – Foundation Level. 3. überarb. Aufl. Heidelberg

[Lilienthal 2016] Lilienthal, Carola: Langlebige Software-Architekturen. Technische Schulden analysieren, begrenzen und abbauen. 1. Aufl. Heidelberg

[Starke 2015] Starke, Gernot: Effektive Software-Architekturen: Ein praktischer Leitfaden. 7. überarb. Aufl. München

[Toth 2015] Stefan Toth: Vorgehensmuster für Software-Architektur: Kombinierbare Praktiken in Zeiten von Agile und Lean. 2. Aufl., Hanser Verlag, München, 2015.

[Zitzewitz 2010] Alexander von Zitzewitz: The Value of Architecture. [online]
<https://www.hello2morrow.com/whitepapers/29/download>, zuletzt abgerufen am 10.04.2019

Diskussion – 15 Minuten

Haben Sie Architekten im Projekt / im Unternehmen? Was sind deren Aufgaben und wie nehmen Sie die Architekten wahr?

iSAQB Foundation Level

Grundbegriffe von Softwarearchitekturen

Beispielfragen



Beispielfragen

Frage 1

A-Frage: Wählen Sie eine Option aus

1 Punkt

Was definiert der Architekturstil?

- (a) Das Datenformat der Schnittstellen aller Bausteine
- (b) Das Datenformat der Schnittstellen bestimmter Bausteintypen
- (c) Die Bausteine und ihre Beziehungen
- (d) Grundsätzliche Regeln für die Basisstrukturierung eines Systems
- (e) Strukturelle Eigenschaften eines Bausteins

Beispielfragen

Frage 1

A-Frage: Wählen Sie eine Option aus

1 Punkt

Was definiert der Architekturstil?

- (a) Das Datenformat der Schnittstellen aller Bausteine
- (b) Das Datenformat der Schnittstellen bestimmter Bausteintypen
- (c) Die Bausteine und ihre Beziehungen
- (d) Grundsätzliche Regeln für die Basisstrukturierung eines Systems
- (e) Strukturelle Eigenschaften eines Bausteins

Beispielfragen

Frage 2

P-Frage: Wählen Sie die **drei besten** Aspekte aus 1 Punkt

Welche Aussagen bzgl. des Zusammenhang von Projekt / Projektleitung und Softwarearchitektur / Softwarearchitekt sind richtig

- (a) Projektleitung und Architektur müssen stetig im Wechsel aufeinander zugehen
- (b) Die Softwarearchitektur kann vom Softwarearchitekten nach Vorliegen sämtlicher Anforderungen entworfen werden
- (c) Softwarearchitektur wirkt i.d.R. deutlich länger als die Projektlaufzeit
- (d) Im vgl. zu einem Softwarearchitekt im Architekturentwurf verfolgt die Projektleitung eher kurzfristige Ziele
- (e) Softwarearchitekten haben hinsichtlich Architekturarbeiten freie Hand im Projektplan

Beispielfragen

Frage 2

P-Frage: Wählen Sie die **drei besten** Aspekte aus 1 Punkt

Welche Aussagen bzgl. des Zusammenhang von Projekt / Projektleitung und Softwarearchitektur / Softwarearchitekt sind richtig

- (a) Projektleitung und Architektur müssen stetig im Wechsel aufeinander zugehen
- (b) Die Softwarearchitektur kann vom Softwarearchitekten nach Vorliegen sämtlicher Anforderungen entworfen werden
- (c) Softwarearchitektur wirkt i.d.R. deutlich länger als die Projektlaufzeit
- (d) Im vgl. zu einem Softwarearchitekt im Architekturentwurf verfolgt die Projektleitung eher kurzfristige Ziele
- (e) Softwarearchitekten haben hinsichtlich Architekturarbeiten freie Hand im Projektplan

Beispielfragen

Frage 3

K-Frage: Wählen Sie für jede Zeile „Geeignet“ oder „Nicht geeignet“ aus 2 Punkte

Welchen Nutzen verschafft eine Softwarearchitektur?

Geeignet Nicht geeignet

- | | | |
|--------------------------|--------------------------|--|
| <input type="checkbox"/> | <input type="checkbox"/> | (a) Vermeidet Änderungen an tragenden Elementen und somit unangemessene Kosten-Nutzen-Verhältnisse |
| <input type="checkbox"/> | <input type="checkbox"/> | (b) Ermöglicht und vereinfacht Wiederverwendung |
| <input type="checkbox"/> | <input type="checkbox"/> | (c) Stellt konzeptionelle Integrität sicher |
| <input type="checkbox"/> | <input type="checkbox"/> | (d) Stellt den Einsatz von Standardprodukten sicher |

Beispielfragen

Frage 3

K-Frage: Wählen Sie für jede Zeile „Geeignet“ oder „Nicht geeignet“ aus 2 Punkte

Welchen Nutzen verschafft eine Softwarearchitektur?

Geeignet Nicht geeignet

- | | | |
|-------------------------------------|-------------------------------------|--|
| <input checked="" type="checkbox"/> | <input type="checkbox"/> | (a) Vermeidet Änderungen an tragenden Elementen und somit unangemessene Kosten-Nutzen-Verhältnisse |
| <input checked="" type="checkbox"/> | <input type="checkbox"/> | (b) Ermöglicht und vereinfacht Wiederverwendung |
| <input checked="" type="checkbox"/> | <input type="checkbox"/> | (c) Stellt konzeptionelle Integrität sicher |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | (d) Stellt den Einsatz von Standardprodukten sicher |

iSAQB Foundation Level

Grundbegriffe von Softwarearchitekturen
Selbstlernkontrolle



Selbstlernkontrolle

In der Greyboxsicht eines Bausteins werden die folgenden Elemente gezeigt:

- Schnittstellen
- Schnittstellen und Attribute eines Bausteins
- Schnittstellen und technische Schnittstellen
- Schnittstellen und innerer Aufbau

Selbstlernkontrolle

In der Greyboxsicht eines Bausteins werden die folgenden Elemente gezeigt:

- Schnittstellen
- Schnittstellen und Attribute eines Bausteins
- **Schnittstellen und technische Schnittstellen**
- Schnittstellen und innerer Aufbau

Selbstlernkontrolle

Was sind die vier Aufgaben des Softwarearchitekten

- Analysieren / Prüfen / Testen / Beschreiben
- Planen / Designen / Review / Implementierung
- Anforderungen und Randbedingungen / Entwurf und Konzeption / Architekturentscheidung / Umsetzung
- Anforderungen und Randbedingungen / Entwurf und Konzeption / Umsetzung / Test

Selbstlernkontrolle

Was sind die vier Aufgaben des Softwarearchitekten

- Analysieren / Prüfen / Testen / Beschreiben
- Planen / Designen / Review / Implementierung
- **Anforderungen und Randbedingungen / Entwurf und Konzeption / Architekturentscheidung / Umsetzung**
- Anforderungen und Randbedingungen / Entwurf und Konzeption / Umsetzung / Test

Selbstlernkontrolle

Was sind die Architekturebenen des Architekturentwurfs

- Fachliche Architerebene, Technische Architekturebene, Technische Infrastruktur, Architekturstil
- IT-Infrastruktur, Business-Architektur, Informationsarchitektur, Softwarearchitektur
- Architekturmuster, Entwurfsmuster, Implementierung, Technische Infrastruktur
- Business-Architektur, Architekturstil, Informationsarchitektur, Technische Architekturebene

Selbstlernkontrolle

Was sind die Architekturebenen des Architekturentwurfs

- **Fachliche Architerebene, Technische Architekturebene, Technische Infrastruktur, Architekturstil**
- IT-Infrastruktur, Business-Architektur, Informationsarchitektur, Softwarearchitektur
- Architekturmuster, Entwurfsmuster, Implementierung, Technische Infrastruktur
- Business-Architektur, Architekturstil, Informationsarchitektur, Technische Architekturebene

Selbstlernkontrolle

Was sind die Architekturebenen einer Enterprise IT Architektur

- Fachliche Architerebene, Technische Architekturebene, Technische Infrastruktur, Architekturstil
- IT-Infrastruktur, Business-Architektur, Informationsarchitektur, Softwarearchitektur
- Architekturmuster, Entwurfsmuster, Implementierung, Technische Infrastruktur
- Business-Architektur, Architekturstil, Informationsarchitektur, Technische Architekturebene

Selbstlernkontrolle

Was sind die Architekturebenen einer Enterprise IT Architektur

- Fachliche Architerebene, Technische Architekturebene, Technische Infrastruktur, Architekturstil
- **IT-Infrastruktur, Business-Architektur, Informationsarchitektur, Softwarearchitektur**
- Architekturmuster, Entwurfsmuster, Implementierung, Technische Infrastruktur
- Business-Architektur, Architekturstil, Informationsarchitektur, Technische Architekturebene

Selbstlernkontrolle

Ein Architekt muss Bottom Up und Top Down vorgehen. Das bedeutet, ...

Freitext ...

Selbstlernkontrolle

Ein Architekt muss Bottom Up und Top Down vorgehen. Das bedeutet, ...

Freitext ...

Was sollte in der Antwort enthalten sein:

- **Iterativ und inkrementelles erarbeiten des Bauplans, Konstruktionswegs**
- **Wechselspiel zwischen den Architekturebenen**
- **Einflüsse der Ebene mit Projektbeteiligten / Stakeholder diskutieren**

Selbstlernkontrolle

Architekturziele sind

- Kurzfristig, weil die Architektur den Projektleiter unterstützt
- Sind nur zu Beginn des Projektes relevant, bis die Implementierung startet
- Wirken über den gesamten Lebenszyklus eines Systems und sind von langfristiger Natur
- ein Führungsinstrument für die Führungskräfte von Softwarearchitekten / Softwarearchitektur Team / Softwarearchitektur Abteilung

Selbstlernkontrolle

Architekturziele sind

- Kurzfristig, weil die Architektur den Projektleiter unterstützt
- Sind nur zu Beginn des Projektes relevant, bis die Implementierung startet
- **Wirken über den gesamten Lebenszyklus eines Systems und sind von langfristiger Natur**
- ein Führungsinstrument für die Führungskräfte von Softwarearchitekten / Softwarearchitektur Team / Softwarearchitektur Abteilung

Selbstlernkontrolle

Ein Architekt braucht sehr gute Fähigkeiten (5 Antworten)

- In Kryptografie und Verschlüsselungen
- Im Umgang mit Menschen
- Datenbankadministration und Performanceoptimierung
- Hacking von IT Systemen
- Soft Skills, Kommunikation und Teamführung
- Im Themengebiet IT Management
- In seiner fachlichen und technischen Domäne
- In der Analyse und Abstraktion
- Im Entwurf von Business-Architekturen
- Im Erstellen von Architektursichten

Selbstlernkontrolle

Ein Architekt braucht sehr gute Fähigkeiten

- In Kryptografie und Verschlüsselungen
- **Im Umgang mit Menschen**
- Datenbankadministration und Performanceoptimierung
- Hacking von IT Systemen
- **Soft Skills, Kommunikation und Teamführung**
- Im Themengebiet IT Management
- **In seiner fachlichen und technischen Domäne**
- **In der Analyse und Abstraktion**
- Im Entwurf von Business-Architekturen
- **Im Erstellen von Architektursichten**

Selbstlernkontrolle

Welche Schnittstellenarten gibt es? (2 Antworten)

- Nach Innen gerichtete Schnittstelle
- Standardsschnittstelle
- HTTP Schnittstelle
- Unabhängige Schnittstelle
- Angebotene Schnittstelle
- Übertragene Schnittstelle
- Technische Schnittstelle
- Angeforderte Schnittstelle
- Eingeschränkte Schnittstelle
- Port-basierte Schnittstelle

Selbstlernkontrolle

Welche Schnittstellenarten gibt es?

- Nach Innen gerichtete Schnittstelle
- Standardsschnittstelle
- HTTP Schnittstelle
- Unabhängige Schnittstelle
- **Angebotene Schnittstelle**
- Übertragene Schnittstelle
- Technische Schnittstelle
- **Angeförderte Schnittstelle**
- Eingeschränkte Schnittstelle
- Port-basierte Schnittstelle

Selbstlernkontrolle

Welche Aussagen bzgl. Bausteine und Schnittstelle sind wahr? (4 Antworten)

- Schnittstellen managen Bausteine
- Schnittstellen tragen die Verantwortung eines Baustein nach aussen
- Ein Baustein kann eine oder mehrere Schnittstellen haben
- Eine Schnittstelle definiert einen Vertrag
- Bausteine kommunizieren über Schnittstellen mit anderen Bausteinen
- Schnittstellen sind Subbausteine des Bausteins

Selbstlernkontrolle

Welche Aussagen bzgl. Bausteine und Schnittstelle sind wahr?

- Schnittstellen managen Bausteine
- **Schnittstellen tragen die Verantwortung eines Baustein nach aussen**
- **Ein Baustein kann eine oder mehrere Schnittstellen haben**
- **Eine Schnittstelle definiert einen Vertrag**
- **Bausteine kommunizieren über Schnittstellen mit anderen Bausteinen**
- Schnittstellen sind Subbausteine des Bausteins

Selbstlernkontrolle

Ein Softwarearchitekt (4 antworten)

- Ist hauptverantwortlich für den Projektplan
- Muss mit vielen Stakeholder effektiv interagieren können
- Trifft alle Entscheidung im Bereich der Architektur als Experte alleine
- Ist Ansprechpartner für das Entwicklerteam
- Implementiert niemals im Entwicklungsprojekt, da diese Rollen strikt getrennt werden sollten
- Führt das Entwicklungsteam und fördert effektive Architekturarbeit im Team
- Berät die Projektleitung bei der Erstellung des Projektplan

Selbstlernkontrolle

Ein Softwarearchitekt

- Ist hauptverantwortlich für den Projektplan
- **Muss mit vielen Stakeholder effektiv interagieren können**
- Trifft alle Entscheidung im Bereich der Architektur als Experte alleine
- **Ist Ansprechpartner für das Entwicklerteam**
- Implementiert niemals im Entwicklungsprojekt, da diese Rollen strikt getrennt werden sollten
- **Führt das Entwicklungsteam und fördert effektive Architekturarbeit im Team**
- **Berät die Projektleitung bei der Erstellung des Projektplan**

Selbstlernkontrolle

Welche Aussage bezüglich Qualität trifft zu (3 Antworten)

- Fördert die Erweiterbarkeit und Änderbarkeit eines Systems
- Ist einfach zu implementieren, da die meisten Themen durch Frameworks abgenommen wird
- Erleichtert das Verstehen der Software
- Ist i.d.R. jedem Stakeholder wichtig und elementarer Bestandteil eines Projektes
- Gewünschte Qualitätseigenschaften können durch Qualitätszenarien anschaulich gemacht werden
- Qualität muss von Softwarearchitektur oft explizit gemacht werden, da Qualitätsanforderungen selten formuliert werden aber indirekt erwartet werden.

Selbstlernkontrolle

Welche Aussage bezüglich Qualität trifft zu

- **Fördert die Erweiterbarkeit und Änderbarkeit eines Systems**
- Ist einfach zu implementieren, da die meisten Themen durch Frameworks abgenommen wird
- Erleichtert das Verstehen der Software
- Ist i.d.R. jedem Stakeholder wichtig und elementarer Bestandteil eines Projektes
- **Gewünschte Qualitätseigenschaften können durch Qualitätszenarien anschaulich gemacht werden**
- **Qualität muss von Softwarearchitektur oft explizit gemacht werden, da Qualitätsanforderungen selten formuliert werden aber indirekt erwartet werden.**

Selbstlernkontrolle

Ein Baustein mit Schnittstellen wird bezeichnet als

- Blackbox
- Invisiblebox
- Greybox
- Darkbox

Selbstlernkontrolle

Ein Baustein mit Schnittstellen wird bezeichnet als

- **Blackbox**
- Invisiblebox
- Greybox
- Darkbox

Selbstlernkontrolle

Beschreibe in wenigen Worten den hierarchischen Aufbau von Systemen anhand der Begriffe Blackbox, Greybox und Whitebox.

Freitext

Antwort: siehe Folie „Sichten auf Bausteine“

Selbstlernkontrolle

Zu welchen anderen Rollen steht der Architekt in Interaktion während des Projektgeschäfts?

- Security / Anwender / Projektleitung / Analyst / Personal / Technischer Mitarbeiter Support
- Tester / Security / Entwickler / Betrieb / Vertrieb
- Entwickler / Projektleitung / Betrieb / Analyst / Security / Anwender
- Betrieb / Security / Tester / Personal / Entwickler / Vertrieb

Selbstlernkontrolle

Zu welchen anderen Rollen steht der Architekt in Interaktion während des Projektgeschäfts?

- Security / Anwender / Projektleitung / Analyst / Personal / Technischer Mitarbeiter Support
- Tester / Security / Entwickler / Betrieb / Vertrieb
- **Entwickler / Projektleitung / Betrieb / Analyst / Security / Anwender**
- Betrieb / Security / Tester / Personal / Entwickler / Vertrieb

Selbstlernkontrolle

Nennen Sie Probleme implizierter Architekturziele.

Selbstlernkontrolle

Nennen Sie Probleme implizierter Architekturziele.

Annahmen von Stakeholdern treffen nicht ein, werden unterschiedlich interpretiert oder das System erfüllt nicht die gewünschten Eigenschaften.



Novatec Consulting GmbH
Berta-Benz-Platz 1
D-70771 Leinfelden-Echterdingen

T. +49 711 22040-700
info@novatec-gmbh.de
www.novatec-gmbh.de