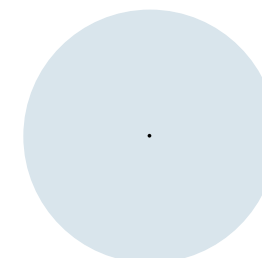


Architekturtreiber



Wiederholung

- **Was ist Software-Architektur?**

- Die Software-Architektur ist die Gesamtheit der **wichtigsten Entwurfsentscheidungen**, die für ein System getroffen werden. (dies ist *eine* mögliche Definition)

- **Welche Entscheidungen sind *wichtig*? D.h. welche Entscheidungen sind *architekturelevant*?**

- Entscheidungen, die das **System als Ganzes** betreffen
- Entscheidungen, die schwer und daher **kostspielig zu ändern sind**
- Entscheidungen, bei denen ein **hohes Risiko besteht, dass sie falsch sind** (z. B. bei neuen oder unbekannten Technologien)
- *Praxis-Tipp*: Fragen Sie sich, was passiert, wenn Sie die Entscheidung bei der Implementierung einem beliebigen Entwickler überlassen.

Frage: Architektur-Treiber

- Was sind Beispiele für architekturelevante Entscheidungen?
- Welche Entscheidungen sind nicht architekturelevant?
- Was "treibt" architekturelevante Entscheidungen an?
 - Woher stammen sie?
 - In welcher Hinsicht beeinflussen diese Entscheidungen das System?



Architektur-Treiber

- Geschäftsziele
 - vom Unternehmen des Kunden/Auftraggebers
 - vom Entwicklungsunternehmen
- Qualitätsattribute
 - Laufendes System (Laufzeitqualitätsattribute)
 - System in der Entwicklung (Qualitätsattribute der Entwicklungszeit)
- die wichtigsten funktionale Anforderungen
 - hervorstechende Eigenschaften
 - solche, die das System "lebensfähig" machen
- Randbedingungen ("*Constraints*")
 - Anforderungen an das System aus der Fachbereich (Domänenanforderungen)
 - Organisatorische, rechtliche und technische Randbedingungen
 - Kosten und Zeit

hauptsächlich
nicht-funktionale
Anforderungen!

Nicht-funktionale Anforderungen

- **Geschäftsziele**, z. B. eine weltweite Einführung der Software
- **Qualitätsattribute zur Laufzeit**, z.B. Zuverlässigkeit, Reaktionszeit und Speicherbedarf
- **Qualitätsattribute zur Entwicklungszeit**, z.B. Testbarkeit der Benutzeroberfläche mit automatisierten Tests
- **Technische Randbedingungen**, z. B. eine bestimmte verfügbare Hardware mit bestimmter I/O-Performance oder vorhandene Middleware, auf der das System laufen soll (z. B. WebLogic Application Server, MySQL-Datenbank)
- **Rechtliche Randbedingungen**, z. B. die Auswirkungen der Datenschutz-Grundverordnung (DSGVO/GSPR), wonach ein Nutzer das Recht hat, alle seine Daten aus dem System zu löschen
- **Andere Randbedingungen** (Bereich, Organisation, Kosten und Zeit)
- **Nicht-funktionale Anforderungen können kritischer sein als funktionale Anforderungen**. Wenn diese nicht erfüllt werden, kann das System unbrauchbar sein.
 - An dieser Stelle kommt die Software-Architektur ins Spiel!
 - Mehr dazu später (**Architektur-Treiber**)

HTWIN Software-Architektur (WIN) Prof. Dr. Johannes C. Schneider

75

Architektur-Treiber @ Digitale Dörfer

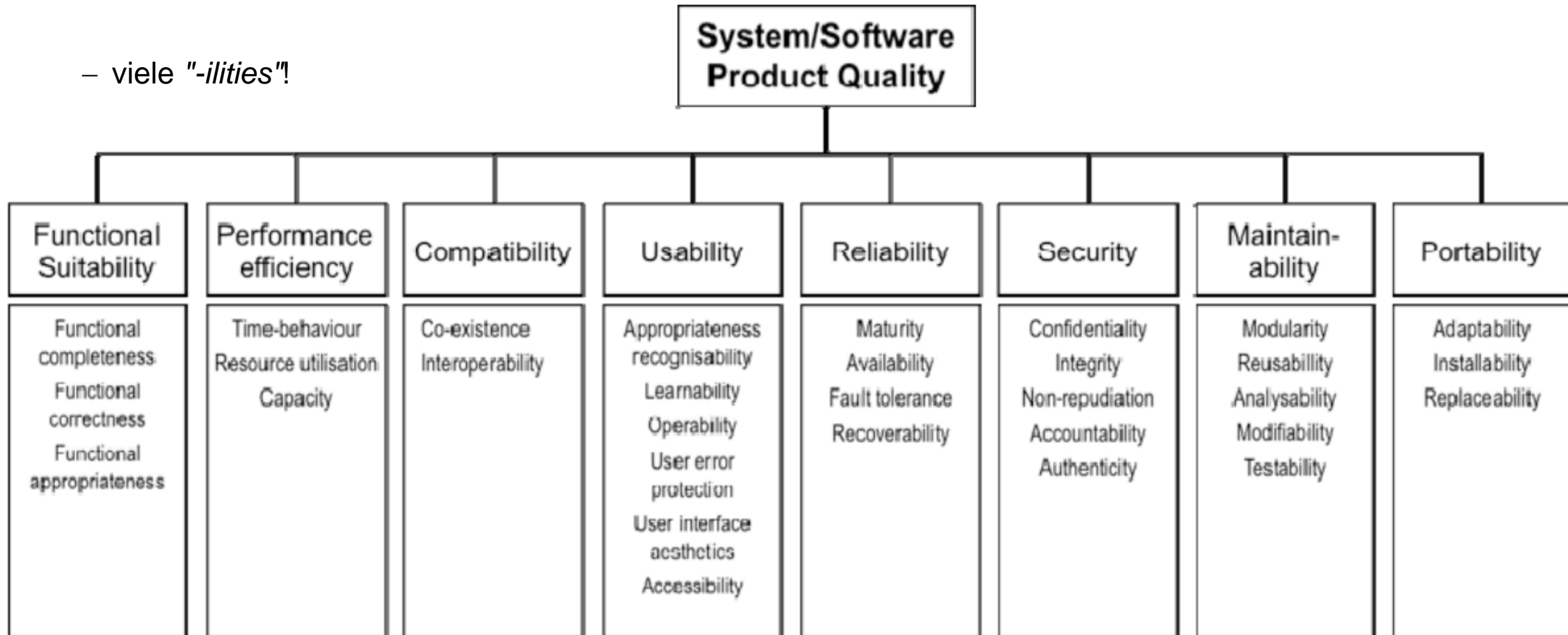
- Geschäftsziele
 - Kunde (Land Rheinland-Pfalz): Verbesserung der *Nahversorgung* im ländlichen Raum
 - Entwicklungsunternehmen: Eine Plattform für weitere Dienstleistungen schaffen
- Qualitätsattribute
 - Laufendes System (Laufzeitqualitätsattribute)
 - Auch für Nicht-Digital-Natives nutzbar
 - Zuverlässige Zustellung von Chat-Nachrichten, ...
 - System in der Entwicklung (Qualitätsattribute der Entwicklungszeit)
 - Leichte Testbarkeit von Backend-REST-Endpunkten
 - Erweiterbarkeit für weitere Dienste
- Wichtigste funktionale Anforderungen
 - Anmeldung bei Drittanbietern (Apple, Facebook, Google)
 - Bestellungen in der BestellBar sollten als Lieferungen in der LieferBar sichtbar sein
- Randbedingungen ("*Constraints*")
 - Übereinstimmung mit der Allgemeinen Datenschutzverordnung (GDPR, deutsch "DSGVO")

ISO 25010

- **Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models**
- Norm, die (unter anderem) ein Produktqualitätsmodell beschreibt, das aus **acht Merkmalen** und weiteren Untermerkmalen besteht
- Vollständige Norm nur nach Bezahlung verfügbar. Zusammenfassung in verschiedener Sekundärliteratur und Blogbeiträgen, z.B. <https://www.perforce.com/blog/qac/what-is-iso-25010>

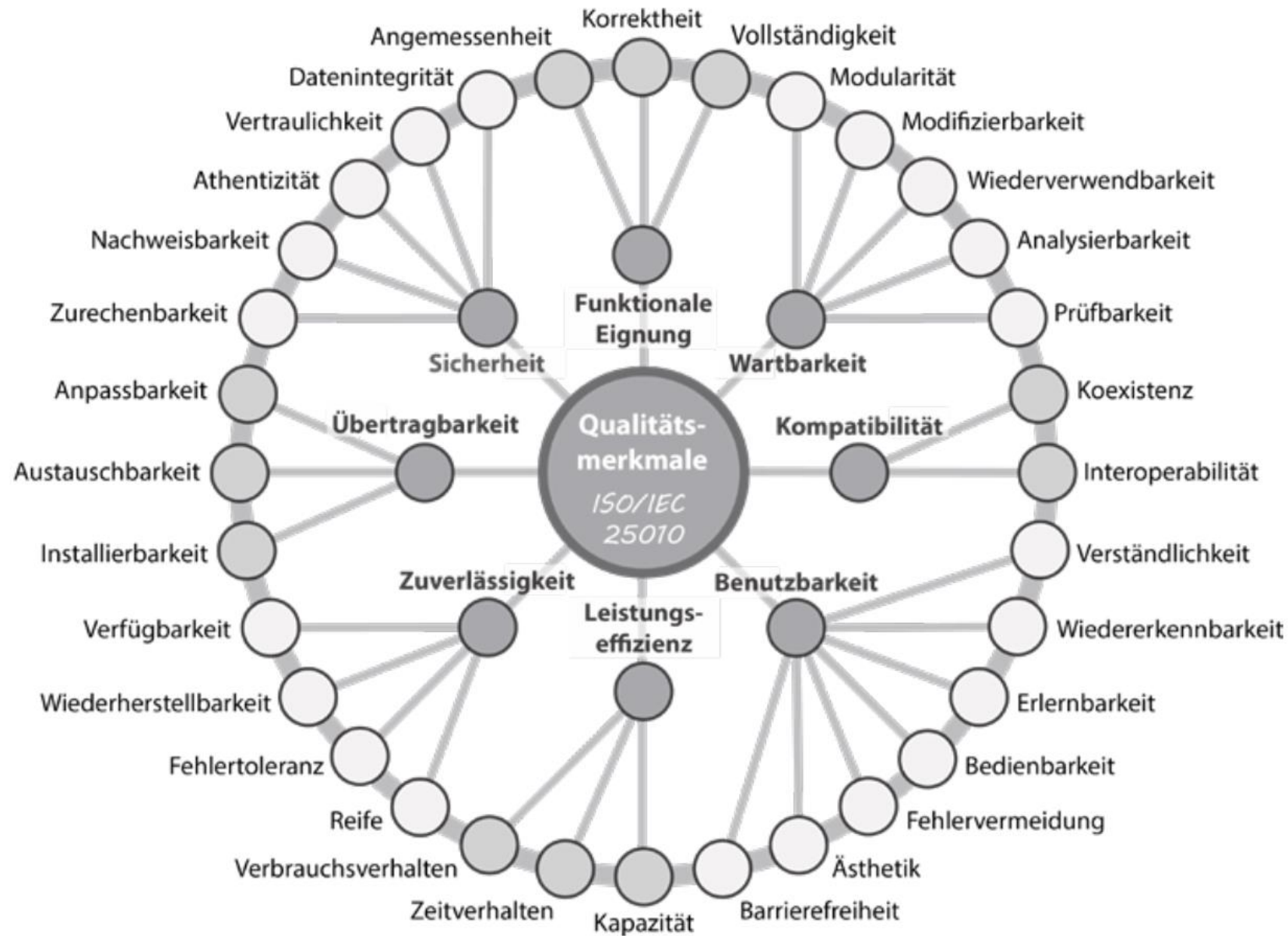
ISO 25010

– viele "-ilities"!



ISO 25010 (deutsche Begriffe)

Aus Stefan Thot "Vorgehensmuster für
Software-Architektur", 3. Auflage,
Seite 33



Beispielhafte Definitionen

4.2.2

performance efficiency

performance relative to the amount of resources used under stated conditions

NOTE Resources can include other software products, the software and hardware configuration of the system, and materials (e.g. print paper, storage media).

4.2.4.5

user interface aesthetics

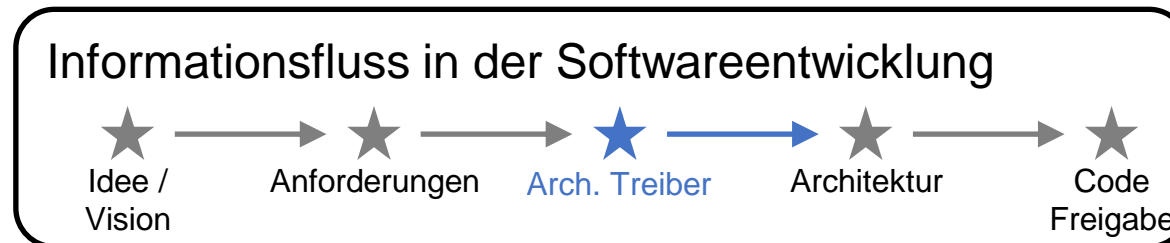
degree to which a user interface enables pleasing and satisfying interaction for the user

NOTE This refers to properties of the product or system that increase the pleasure and satisfaction of the user, such as the use of colour and the nature of the graphical design.

- sind sehr offen formuliert
- viel Raum für Interpretationen lassen

Probleme mit all diesen "-ilities"

- In den Anforderungsdokumenten fehlen sie oft
- Falls vorhanden, sind sie oft nur vage formuliert
 - "Das System muss schnell reagieren"
 - "Die App muss ein positives Nutzererlebnis bieten"
- Deshalb: **Treibererhebung wichtiger Teil der Architekturarbeit** (Anforderungsdokumente allein reichen nicht aus)
 - **Sammlung** und **Validierung** potenzieller **Architekturtreiber** aus allen Quellen (ähnlich wie bei RE-Aktivitäten)
 - Eingabe:
 - Anforderungsdokumente
 - alle Stakeholder (bei fehlenden, unklaren Qualitätsanforderungen)
 - **Spezifizieren Sie** die Treiber auf **messbare Weise**
- Treiber sind **wichtig**: Grundlage für Designentscheidungen



Architektur-Szenario: Definition

- “An **architecture scenario (architecture quality driver)** is a crisp, concise description of a situation that the system is likely to face, along with a definition of the response required of the system”
[Rosanzki, Woods, 2005]
- *Übersetzung: "Ein **Architekturszenario (Architekturqualitätstreiber)** ist eine klare, prägnante Beschreibung einer Situation, mit der das System wahrscheinlich konfrontiert wird, zusammen mit einer Definition der vom System geforderten Reaktion.*
- Architektur-Szenarien
 - sind ein zentrales Artefakt bei der Architekturgestaltung und -bewertung
 - Ermöglichen die genaue Beschreibung von Qualitätsanforderungen

Architektur-Szenario-Vorlage

Kategorisierung	
Szenarioname	<i>Prägnanter, kurzer Name</i>
Szenario-ID	<i>Eindeutige Kennzeichnung</i>
Status	<i>[Offen, Erhoben, In Planung, Entworfen, In Umsetzung, Realisiert, Erledigt]</i>
Priorität	<i>[Hoch - Mittel - Niedrig]</i>

Beschreibung		Quantifizierung
Umgebung	<i>Kontext und/oder Ausgangssituation, die auf dieses Szenario zutrifft</i>	<i>Messbare Effekte, die die Umgebung betreffen</i>
Stimulus	<i>Das Ereignis, der Auslöser oder die Bedingung, die sich in diesem Szenario ergeben</i>	<i>Messbare Effekte, die den Stimulus (Auslöser) betreffen</i>
Antwort	<i>Die erwartete Reaktion des Systems auf das Szenario-Ereignis (Black-Box-Ansicht, die keine Einschränkungen für das Design vorsieht)</i>	<i>Messbare Effekte, die auf die Reaktion treffen Messbare Indikatoren, dass das Szenario durch die Architektur umgesetzt wurde</i>

Architecture Scenario Template

Categorization	
Scenario Name	<i>Concise short name</i>
Szenario ID	<i>Unique Identifier</i>
Status	<i>[Open, Elicited, Under Design, Designed, Under Realization, Realized, Done]</i>
Priority	<i>[High – Medium – Low]</i>

Description		Quantifizierung
Environment	<i>Context and/or initial situation applying to this scenario</i>	– <i>Measurable effects applying to the environment</i>
Stimulus	<i>The event, trigger or condition arising from this scenario</i>	– <i>Measurable effects applying to the stimulus</i>
Response	<i>The expected reaction of the system to the scenario event (black box view putting no constraints on the design)</i>	– <i>Measurable effects applying to the response</i> – <i>Measurable indicators that the scenario has been achieved by the architecture</i>

Beispiel für ein Architektur-Szenario

Kategorisierung	
Szenarioname	<i>Verbindung, Nutzer mit begrenzter Bandbreite</i>
Szenario-ID	<i>AD.01.Performance</i>
Status	<i>Realisiert</i>
Priorität	<i>Hoch</i>

Beschreibung		Quantifizierung
Umgebung	<i>Ein Nutzer in einem ländlichen Gebiet verwendet die Logistik-App. Die verfügbare Bandbreitenverbindung ist auf UMTS-HSPA beschränkt.</i>	– <i>Bandbreite < 7,2 Mbit/s</i>
Stimulus	<i>Der Benutzer möchte sich die verfügbaren Transportaufträge in seiner Gemeinde anzeigen lassen.</i>	– <i># Anzahl der Transportaufträge = 10</i>
Antwort	<i>Die App zeigt alle verfügbaren Transportaufträge innerhalb von 0,5 Sekunden an.</i>	– <i>alle Transportaufträge angezeigt < 0,5s</i>

Beispiel für ein Architektur-Szenario

Es ist **nicht zwingend erforderlich, die Vorlage zu verwenden**. Sie können einen Treiber auch als **Textabsatz** formulieren:

"Ein Nutzer in einem ländlichen Gebiet nutzt die Logistik-App. Die verfügbare Bandbreitenverbindung ist auf UMTS-HSPA ($< 7,2 \text{ Mbit/s}$) beschränkt.

Der Nutzer möchte sich die verfügbaren Transportaufträge ($\# < 10$) in seiner Gemeinde anzeigen lassen.

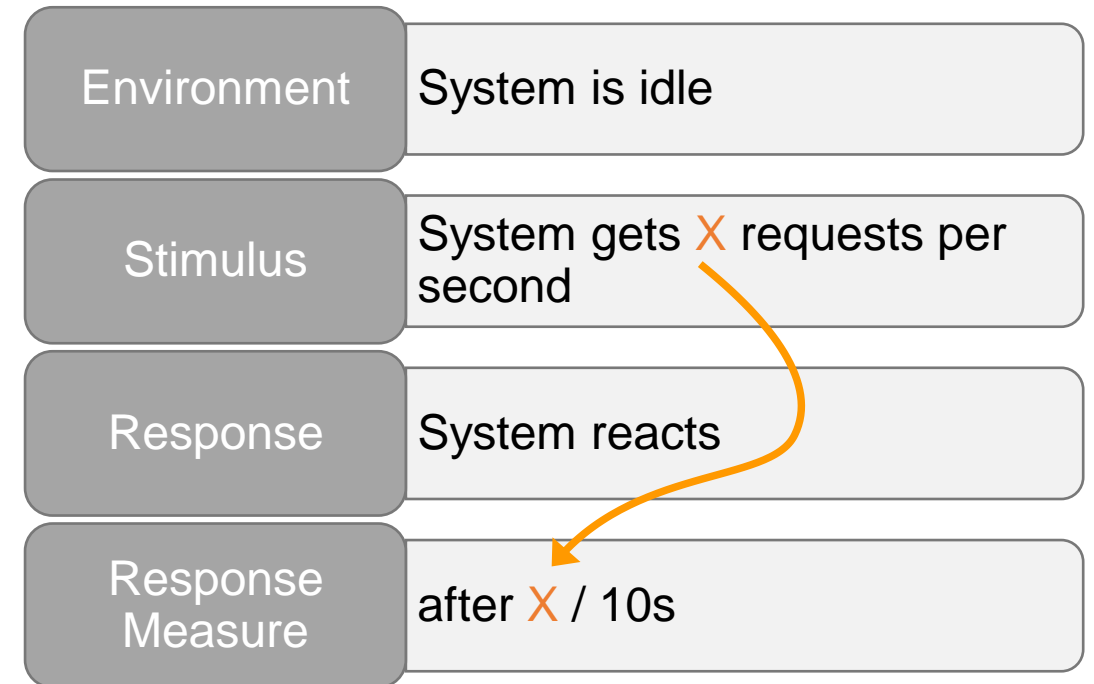
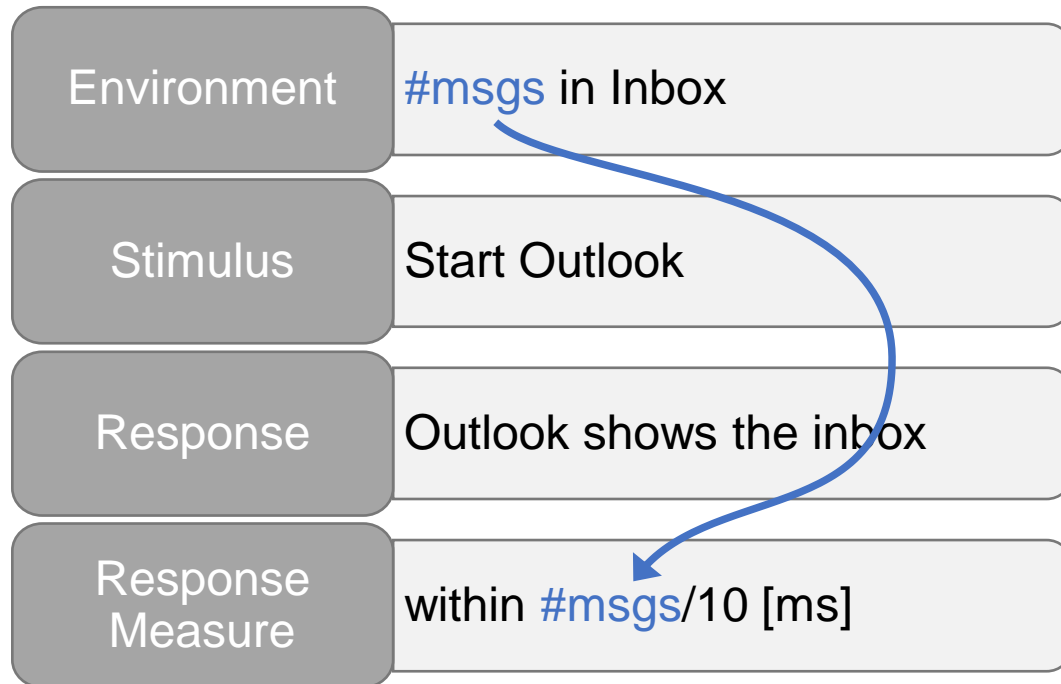
Die App zeigt alle verfügbaren Fahraufträge innerhalb von $0,5\text{s}$ an."

Die **Szenariovorlage** hilft jedoch **bei der Strukturierung** und wird daher empfohlen.

Weitere Beispiele:

- "Ein Benutzer möchte das System aktualisieren. Die Aktualisierung wird mit maximal 3 Klicks ausgelöst. "
- "Während des Betriebs fällt ein Server aus. Alle laufenden Operationen werden durch den Ausfall nicht beeinträchtigt.
- "Jede Benutzereingabe erzeugt innerhalb von $0,2\text{s}$ eine visuelle Reaktion.
- "Eine bestimmte neue Funktion soll implementiert werden. Ein Team von 5 Personen ist in der Lage, die Funktion innerhalb von drei Tagen zu realisieren.

Quantifizierung und Maßnahmen können relativ ausgedrückt werden!



Metriken zur Spezifizierung von Qualitätsattributen

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Beispiel:

- ~~"Das System sollte schnell sein"~~
- Besser: "Nach der Anmeldung wird dem Benutzer der Hauptbildschirm in max. 200ms angezeigt"

Tabelle von Ian Sommerville

Randbedingungen

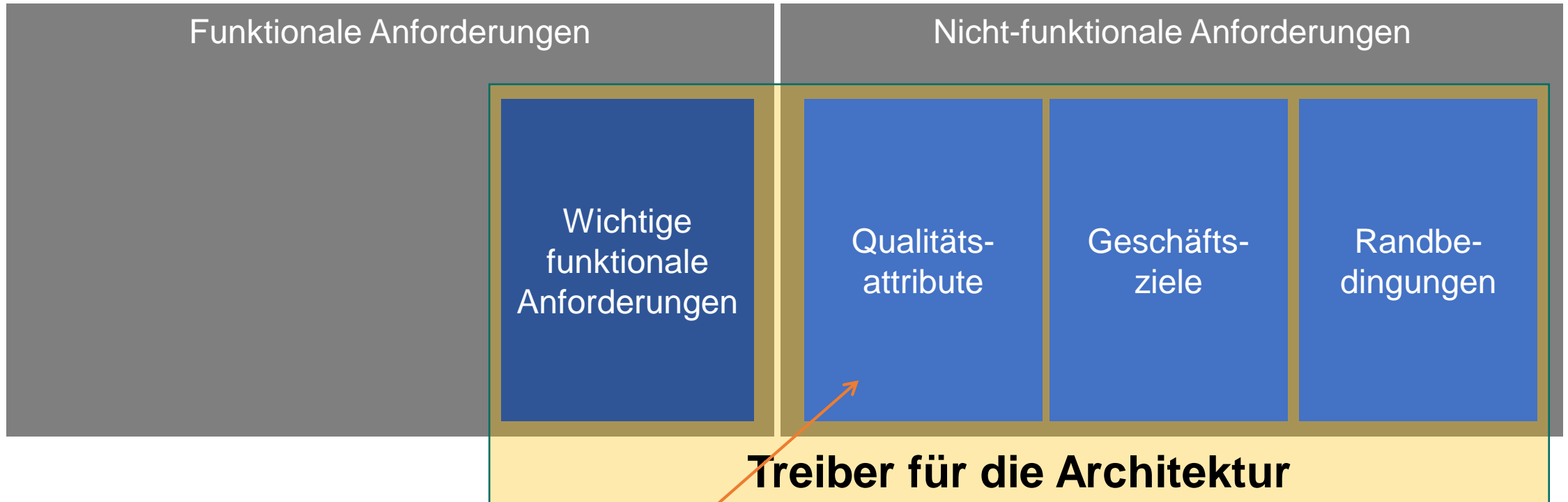
- Arten von Randbedingungen:
 - Randbedingungen für das System aus der Domäne/dem Fachbereich ("Um als Medizinprodukt zu gelten, muss es X, Y, Z erfüllen")
 - Einschränkungen aus der **Unternehmensstruktur** und –kultur (Teamstruktur/-größe, Sicherheitsrichtlinien, Strategien für Open Source/Closed Source)
 - **Technische** Randbedingungen ("Muss mit einer Oracle-Datenbank laufen")
 - **Gesetzliche** Auflagen ("Muss DSGVO-konform sein")
 - **Kosten** und **Zeit** ("Muss bis zum 2.2.2022 mit einem Budget von 222.222 € durchgeführt werden")
- Randbedingungen werden in **natürlicher Sprache** angegeben
- **Aufgepasst:** Hinterfragen Sie **technische Randbedingungen**, denn sie sind **oft nicht so fest**, wie sie scheinen:
 - Sie sind vielleicht überholt, wenn das Projekt endlich beginnt.
 - Sie sind vielleicht von nichttechnischen Personen formuliert

Notationen für Architektur-Treiber

Geschäftsziele	Randbedingungen	Qualitätsattribute	Wichtige funktionale Anforderungen
<ul style="list-style-type: none">• Natürliche Sprache	<ul style="list-style-type: none">• Natürliche Sprache	<ul style="list-style-type: none">• Architektur-Szenario-Vorlage• (Use Cases)	<ul style="list-style-type: none">• Use Cases• User Stories / Epics• Given-When-Then• Natürliche Sprache

– siehe "Anforderungsspezifikation" im Abschnitt Requirements Engineering

Übersicht



am besten mit
Architektur-Szenarien
beschrieben

Architektur-Szenario-Vorlage	
Metadaten	
System Name	Kurz, prägnanter Name
System ID	Einziges Dokument
Stadium	Entwurf, Planung, Entwurf, in Umsetzung, Realisiert, Fiktiv
Version	1.0.0, 1.0.1, 1.0.2
Beschreibung	
Umfeld	Kontext und/oder Ausgangssituation für dieses Szenario
Szenario	Das Ereignis, das das System oder ein Teil davon auslöst
Aktion	Die erwarteten Reaktionen des Systems auf das Szenario-Ereignis (Beschreibung des Verhaltens, das keine Einschränkungen für den Entwurf enthält)
Quantifizierung	
Umfeld	Welchen Auswirkungen hat das Szenario?
Szenario	Welchen Auswirkungen hat das Ereignis?
Aktion	Welchen Auswirkungen hat die Reaktion?

Software-Architektur (WIN) Prof. Dr. Johannes C. Schneider

Wiederholung: Warum in Architektur-Treiber investieren...

...wenn es so gute Requirements-Dokumente gibt?

- Anforderungen...
 - sind oft nicht gut analysiert und dokumentiert
 - sind meist nicht vollständig
 - decken häufig die Aspekte der Entwicklung und des Betriebs nicht ab
- Manchmal ist die Menge der Anforderungen so groß, dass Architekt:innen sie kondensieren müssen

Praktische Anwendung: Spezifizieren eines Architekturszenarios

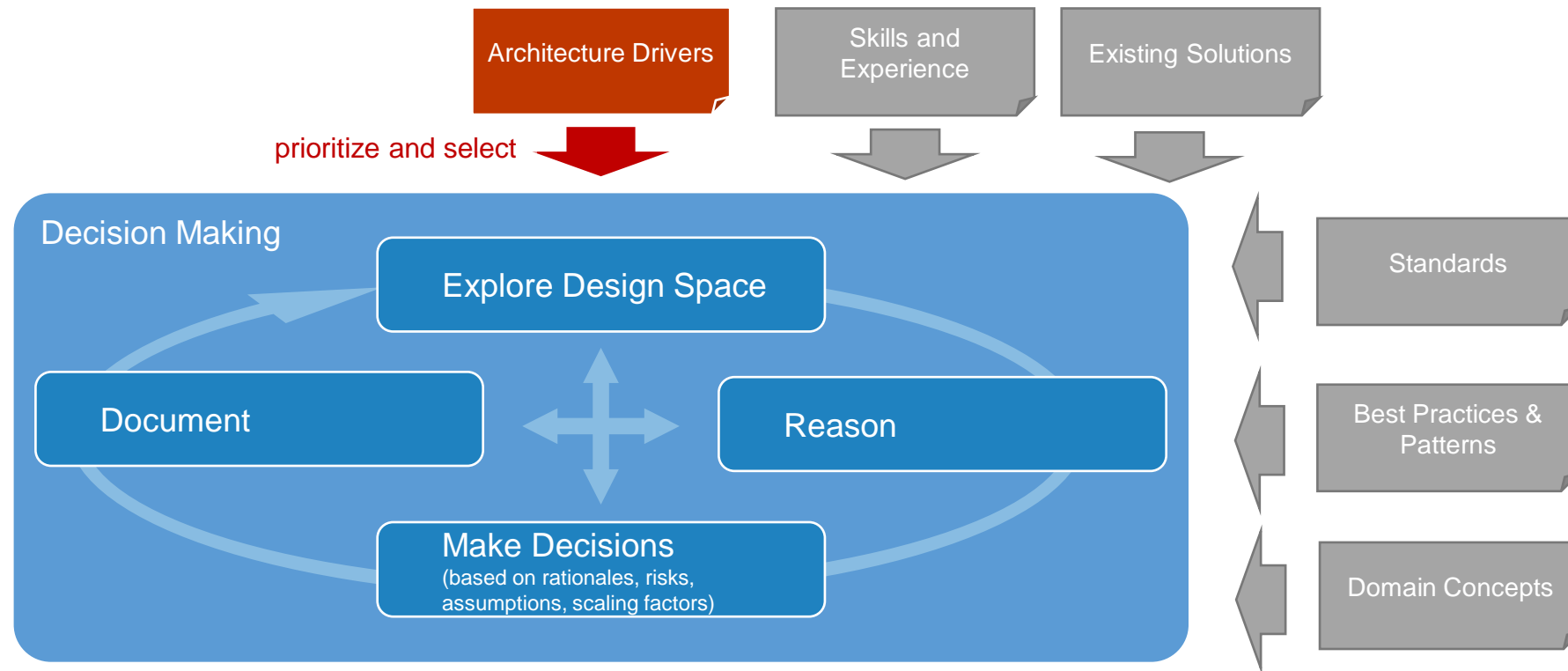


- Jedes Team:
 - Schreiben Sie ein oder zwei nicht-funktionale Anforderungen in das Formular der Szenario-Vorlage für Ihr Beispielsystem
 - Einige Beispiele:
 - Das System hat derzeit x Benutzer. y neue Benutzer registrieren sich innerhalb von z Tagen. Wie soll das System damit umgehen?
 - Im System läuft Version x. Version x+1 soll installiert werden. Was ist das erwartete Systemverhalten?
 - Der Benutzer ist online. Während er xyz macht, geht die App offline. Was sollte passieren?
 - Der Benutzer löst eine Operation auf dem Server aus aus. Wie schnell sollte die Antwort eintreffen?
 - Verwenden Sie messbare Werte bei der Quantifizierung

"Name"	Quantifizierung	
Umgebung		
Stimulus		
Antwort		

Wie kann man angemessene Designentscheidungen treffen?

- "Die Software-Architektur ist die Gesamtheit der wichtigsten Designentscheidungen, die für das System getroffen werden."
- Wir machen diese Entscheidungen **angemessen (adäquat)** für das zu erstellende System, indem wir die Treiber adressieren (und unsere Erfahrung benutzen)



Architekturlösungskonzept

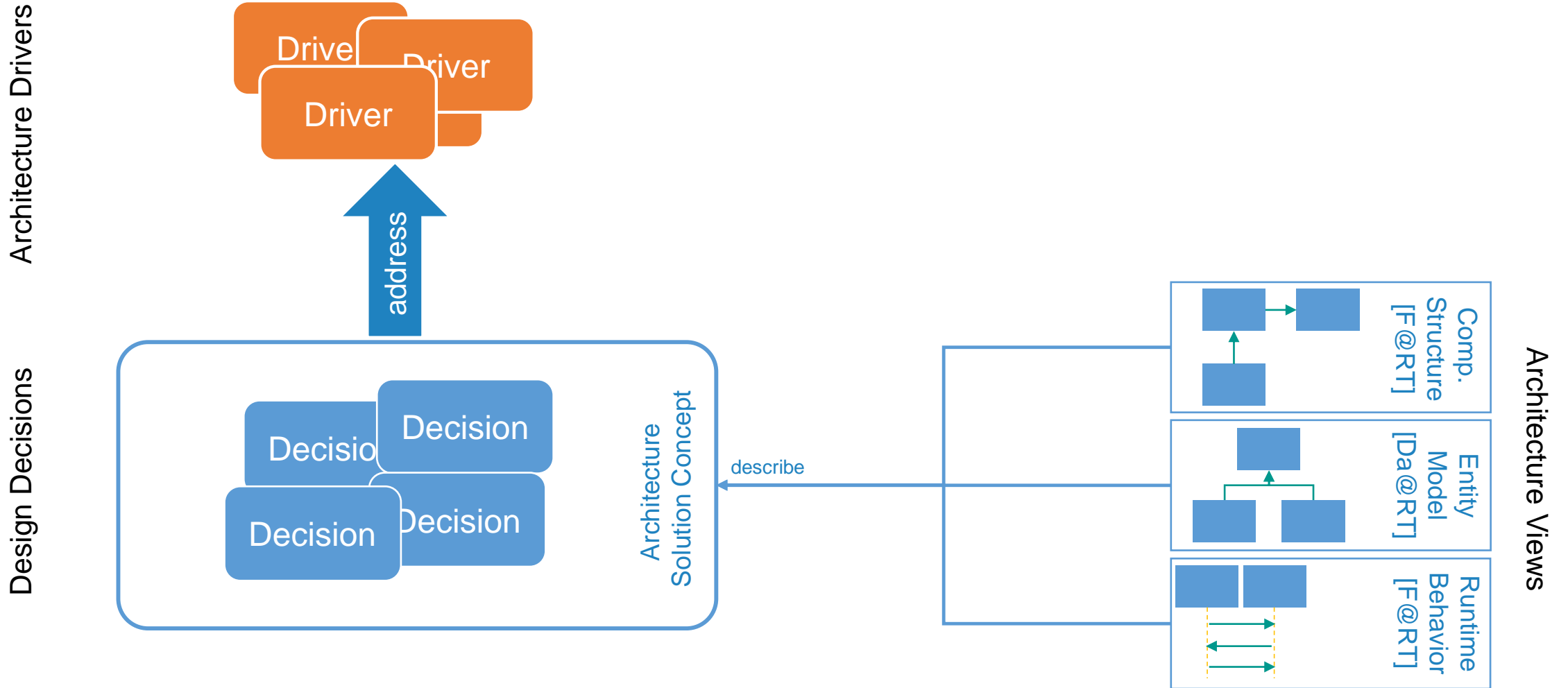
- Ein **Architekturlösungskonzept** (*architecture solution concept*) beschreibt die Lösung für eine Reihe von kohärenten Architekturtreibern, um die Verständlichkeit und Einheitlichkeit zu erleichtern.
- Dabei handelt es sich in der Regel um **querschnittliche Aspekte** (*cross-cutting aspects*), die das System als Ganzes betreffen

Standardaspekte in Informationssystemen

- Benutzerschnittstellen (mobil, Web, Desktop)
- Session-Verwaltung
- Persistenz von Daten, Transaktionen
- Datenintegrität
- Systemverwaltung und -betrieb
- Ausnahmen (Exceptions) und Fehlerbehandlung
- Logging und Nachverfolgen
- Konfigurationsmanagement
- Parallelisierung
- Mehrsprachigkeit
- Software-Aktualisierung
- Skalierung und Clustering
- (Hohe) Verfügbarkeit
- ...

*Müssen als
Architekturtreiber
ausdetailliert werden*

Architecture Solution Concepts




Beispiel für ein Architekturlösungskonzept

- Architektur-Treiber:
 - Neue Daten werden dem Benutzer innerhalb von 1 Sekunde angezeigt.
 - Nutzer mit begrenzter Bandbreite (< 7,2 Mbit/s) sollten die App nutzen können
 - Die Nutzung des Datenvolumens sollte 10 MB pro Monat nicht überschreiten.

- Design-Entscheidungen:

Performance Concept

- ClientDTOs als Kommunikationsdaten zwischen Client und Server 
- Bildschirmspezifische API-Endpunkte, die vom Backend bereitgestellt werden
- Modell-Mapping (Backend/Server-Modell <-> Client-Modell) mit dedizierten Mapping-Komponenten pro Bereich
- Getrennte Ereignismodelle für eine schlanke Kommunikation zwischen Client und Server: ClientEvents

Data Transfer Object (Datentransferobjekt)

- Objekt/Klasse, die zusammengehörende Daten zusammenfasst
- Kann leicht in einer Datenbank gespeichert werden
- Einfach zu serialisieren, z.B. als JSON

```
public class Person {  
    private long id;  
    private String firstName;  
    private String lastName;  
    private String email;  
    private String phoneNumber;  
    private Address address;  
    // ...constructor, getter, setter  
}  
  
public class Address {  
    private long id;  
    private String street;  
    private String house;  
    private String zip;  
    private String city;  
    // ...constructor, getter, setter  
}
```

```
{  
    "id": 42,  
    "firstName": "Johannes",  
    "lastName": "Schneider",  
    "status": "REGISTERED",  
    "email": "johannes.schneider@htwg-konstanz.de",  
    "phoneNumber": "+ 49 7531 206-0",  
    "address": {  
        "id": 12,  
        "city": "Konstanz",  
        "zip": "78462",  
        "street": "Alfred-Wachtel-Straße",  
        "house": "8"  
    }  
}
```

Client Data Transfer Objekt (Client-Datenübertragungsobjekt)

- Objekt-/Klassenspezifisch für den Datenaustausch mit (Web-/Mobil-)Clients
- nur wirklich benötigte Daten werden übertragen
- interne Referenzen über id und eigene Endpunkte
- in diesem Beispiel: die Adresse kann über einen eigenen Endpunkt abgerufen und im Client zwischengespeichert werden

```
{
  "id": 42,
  "firstName": "Johannes",
  "lastName": "Schneider",
  "status": "REGISTERED",
  "email": "johannes.schneider@htwg-konstanz.de",
  "phoneNumber": "+ 49 7531 206-0",
  "address": {
    "id": 12,
    "city": "Konstanz",
    "zip": "78462",
    "street": "Alfred-Wachtel-Straße",
    "house": "8"
  }
}
```

weniger Daten

ClientPerson.json:

```
{
  "id": 42,
  "firstName": "Johannes",
  "lastName": "Schneider",
  "addressId": 12
}
```

ClientAddress.json:

```
{
  "id": 12,
  "city": "Konstanz",
  "zip": "78462",
  "street": "Alfred-Wachtel-Straße",
  "house": "8"
}
```

Weitere Beispiele von Lösungskonzepten

- Authentifizierungs- und Autorisierungskonzept
- Mehrmandantenfähigkeitskonzept
- Skalierbarkeitskonzept
- Logging-Konzept
- API-Versionierungskonzept
- Internationalisierungskonzept
- Eingabevalidierungskonzept
- Datenverarbeitungskonzept
- Continuous-Delivery-Konzept
- Monitoring-Konzept (Überwachung im Betrieb)
- Konzept zur Strukturierung des Quellcodes
- Konzept der Versionskontrollnutzung
- Systemkonfigurationskonzept
- Konzept zur Trennung der Dienste
- Konzept der Datenspeicherung
- ...

Beispiel einer informellen Dokumentation: Client-Datenmodell

Problem

The backend data model contains the information that is required for managing the workflow of all users.

- Thus it on the one hand contains sensitive data from the perspective of a single user.
- On the other hand it is more complex than required for displaying data to a single user.
- Additionally there are continuous changes in the backend that should not break the client.
- Finally it is rather confusing for clients if a lot of attributes are not used by the backend when they send data back

Idea

Separate client and backend data model. Map the backend data model to the client data model for every request.

- The client entities are simple Entities, containing less or adapted data
- REST controllers have the task of mapping the entities between the models
- For every domain a ModelMapper (ModelTransformer in legacy code) is available that can be injected in the controllers
- ModelMapper are hierarchically included, according to the domain dependencies

Discarded Alternatives

- Manipulating the backend data entities before returning them
 - Such a manipulation is bound to the actual entities, only removals are possible, no additions
 - It is hard to check if the actual manipulation has taken place
 - If a manipulated entity is stored in the database data is lost

Comments

- Currently the "manual" mapping approach is a bit time consuming, since a new mapping method needs to be written for every entity
 - A check if an automated mapping via reflection is possible should be made, since the attribute names are often identical

Wiki-Seite