SOFTWAREDOKUMENTATION – LIEBE AUF DEN ZWEITEN BLICK

Stephan Pirnbaum

IT Tage am 14.12.2022

BUSCHMAIS

- Dresdner IT-Consulting-Unternehmen
  - Zu Hause in der Java-Welt verbunden mit neuesten Technologien und Altbewährtem (z. B. React, Angular, Python, Neo4j, etc.)
  - Langjährige Forschungskooperationen mit Hochschulen und Engagement in Communities (JUG Saxony)
- Unsere Schwerpunkte
  - Strategische, zielgerichtete und nachhaltige Entwicklung von Geschäftsanwendungen
  - Software-Qualitätsanalysen und -Sicherung
  - Köpfe hinter jQAssistant: von Entwicklung über Workshops bis hin zu Consulting
- Unsere Kunden
  - Kleine Auswahl: ITZBund, Sächsische Aufbaubank-Förderbank, ASML, GlobalFoundries, Thyssenkrupp Steel, Telekom, COOP Schweiz

SOFTWAREDOKUMENTATION

SOFTWAREDOKUMENTATION*

*Disclaimer: Der Fokus liegt auf der Dokumentation der Architektur außerhalb des Source Codes.

Warum?

Was?

SOFTWAREDOKUMENTATION

Wie?

Warum?

Was?

SOFTWAREDOKUMENTATION

Wie?

… EIN KURZER RÜCKBLICK

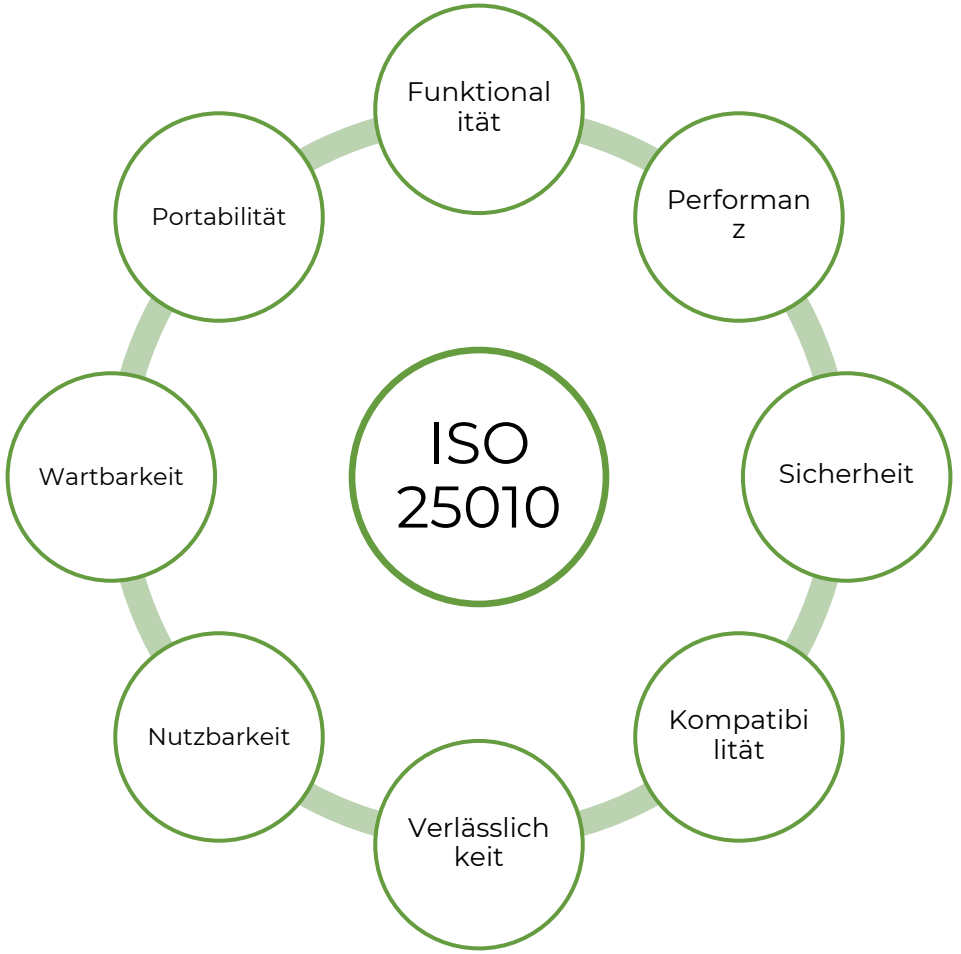
Software wird entwickelt, um Geschäftsmodelle effizient zu ermöglichen und Prozesse zu optimieren

Im Vordergrund der Entwicklung steht also die fachliche Domäne. Diese und geschäftliche Vorgaben bestimmen die Qualitätsziele

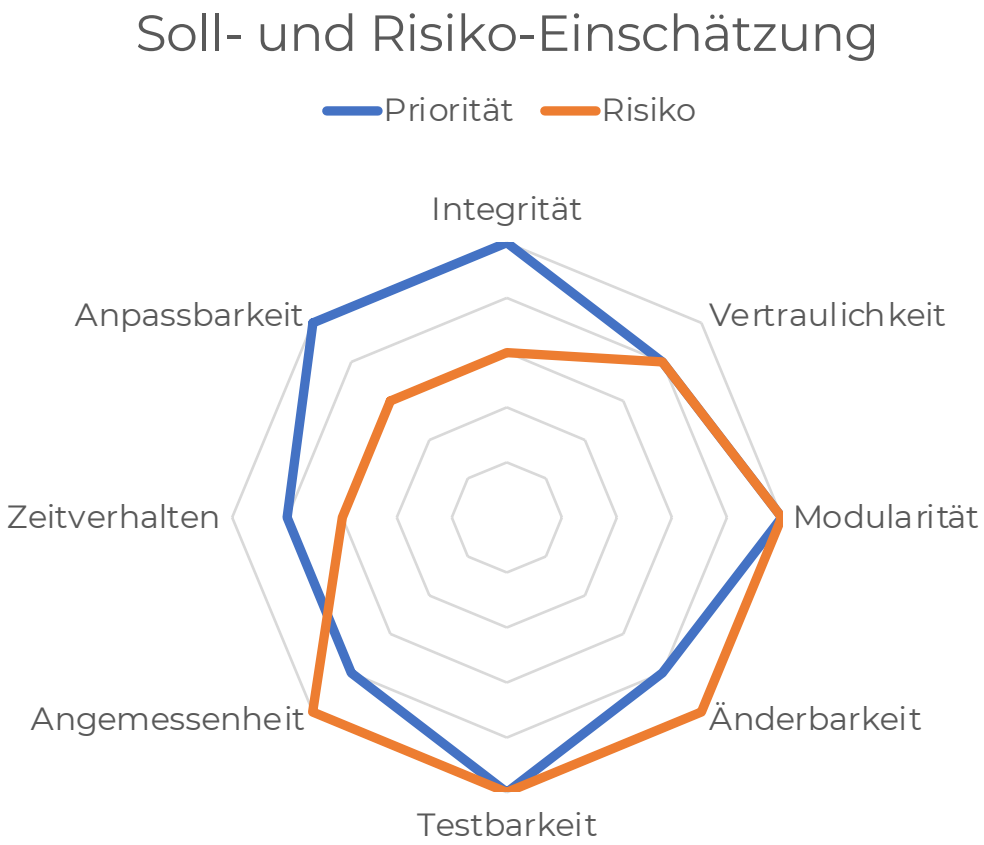
Architektur ist der **Enabler**

zur Erreichung von

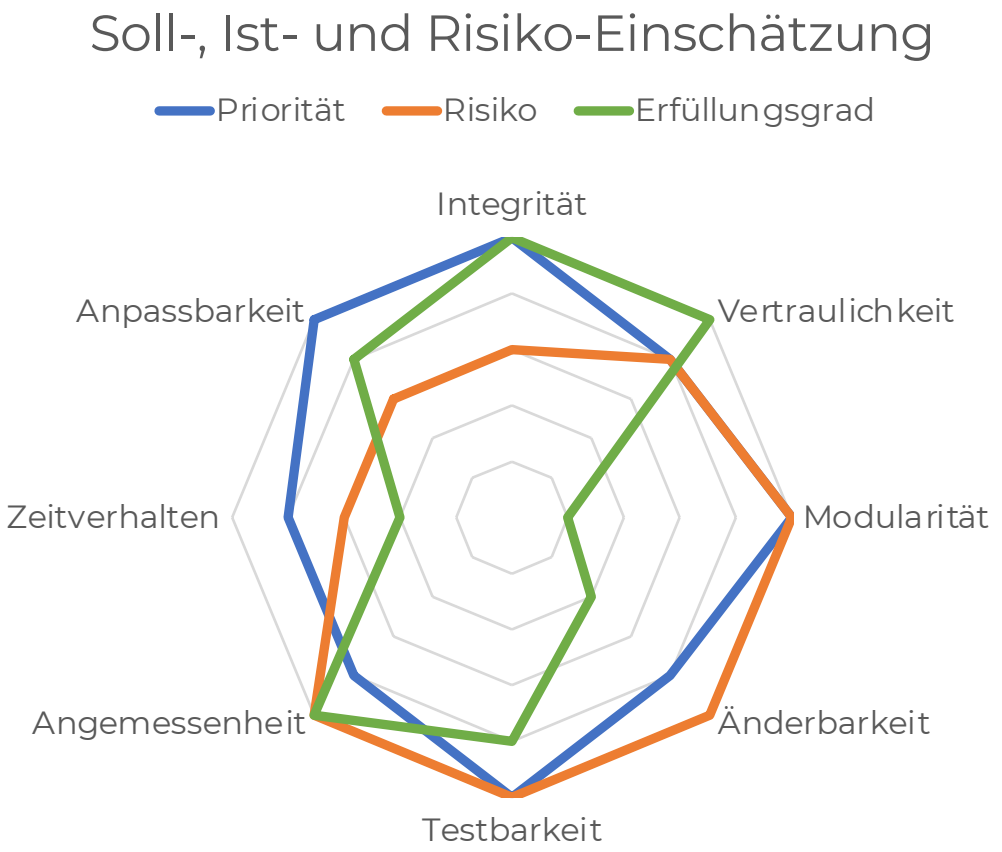
Qualitätszielen



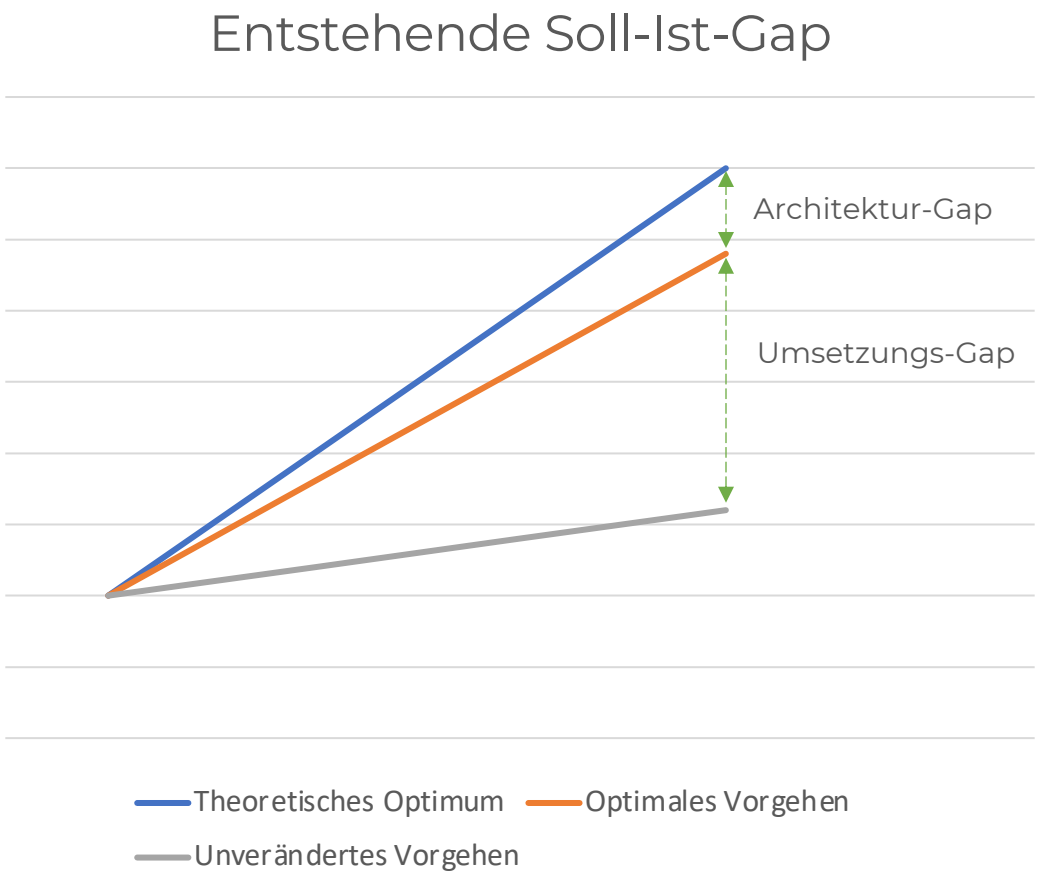
Architektur wird auf **Basis** gesetzter Qualitätsziele entworfen



Anforderungen und verfügbare Technologien **ändern** sich, Design und Implementierung laufen **auseinander**



- Abweichungen durch
  1. Diskrepanz zwischen Anforderungen und Architektur
  2. Auseinanderlaufen von Architektur und Implementierung



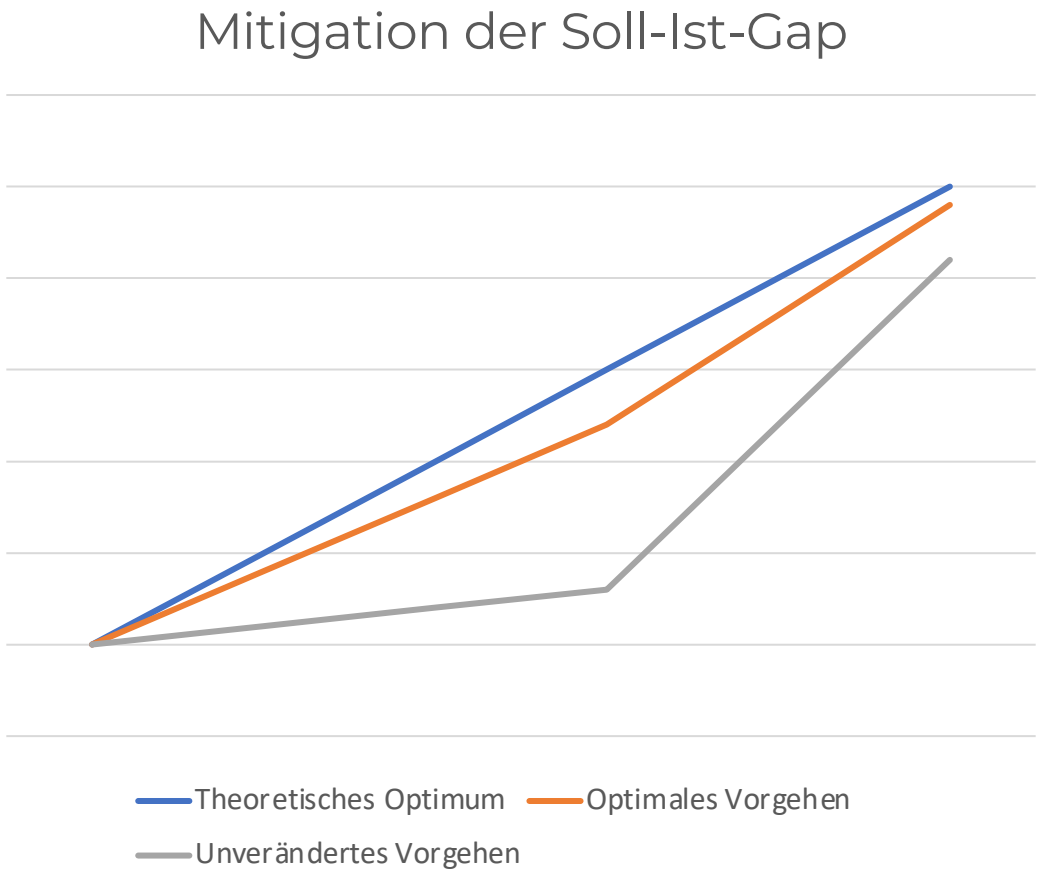
Entstehung von

Diskrepanzen ist ein

natürlicher Prozess

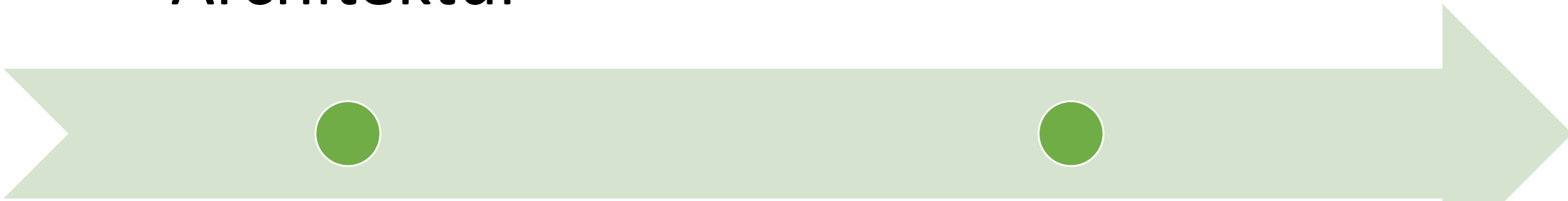
… welcher **abgeschwächt**

werden kann und muss



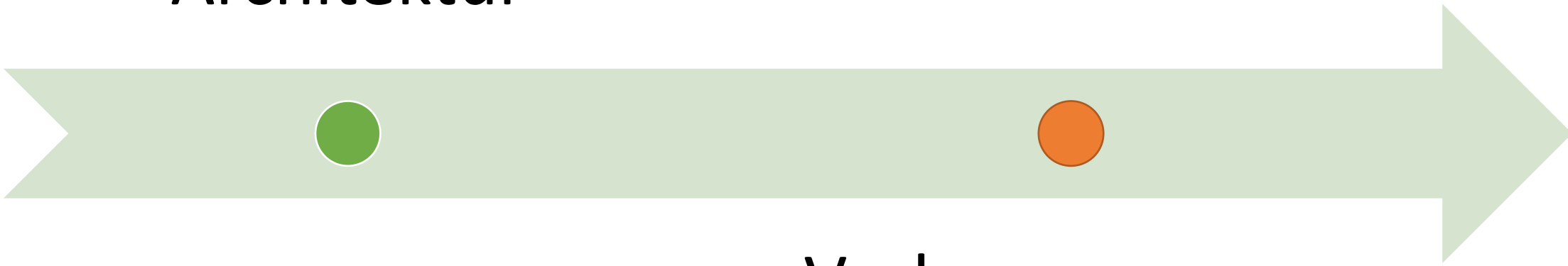
- Lösung erfolgt durch
  1. Anpassung der Architektur durch neue Entscheidungen
  2. Konsequente und aktuelle Dokumentation der Architektur
  3. Strikter Abgleich von Architektur und Umsetzung

Erarbeitung der Architektur

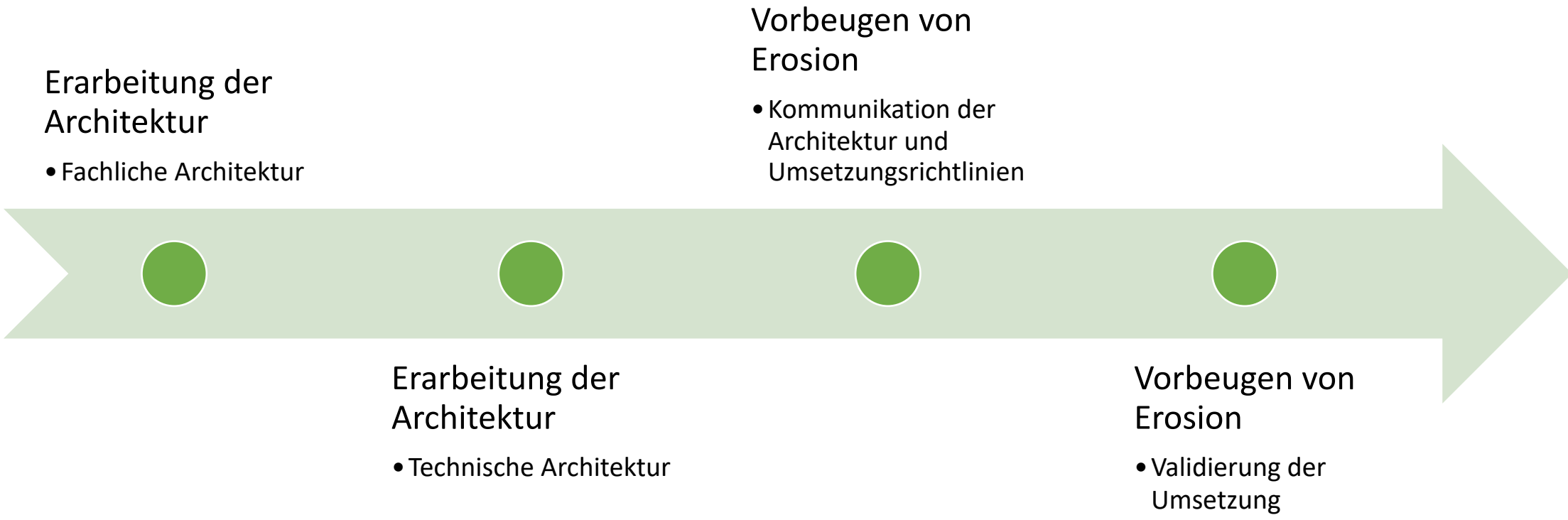


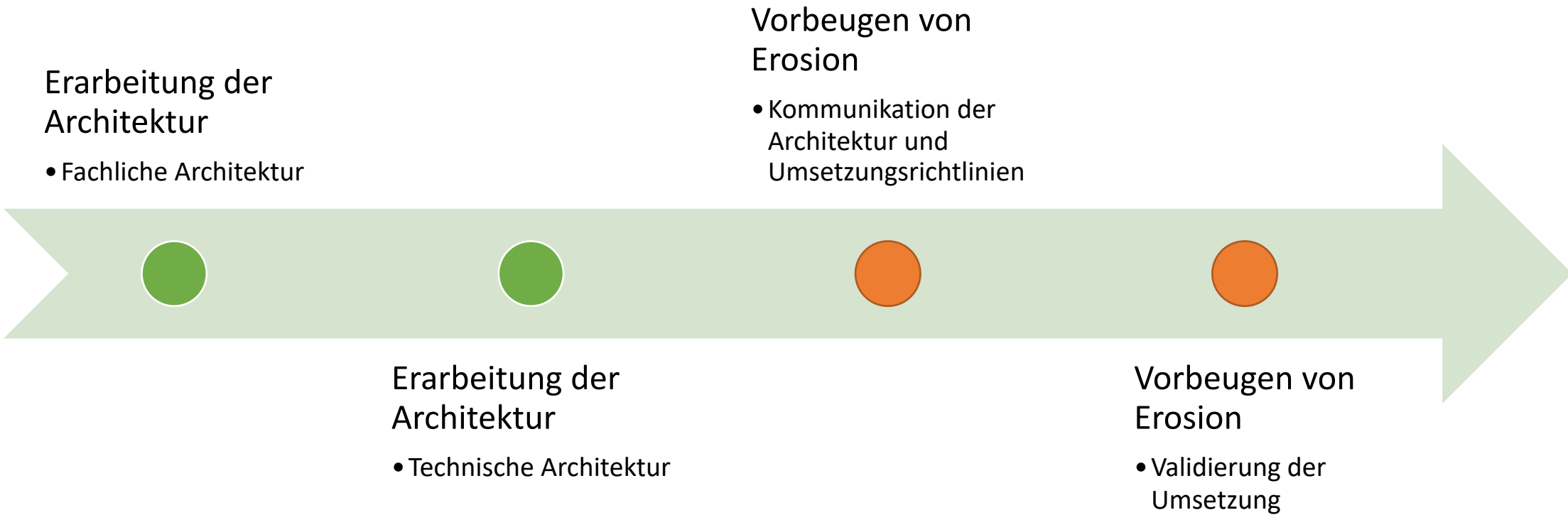
Vorbeugung von Erosion

Erarbeitung der Architektur



Vorbeugung von Erosion


Erarbeitung der Architektur
• Fachliche Architektur
Erarbeitung der Architektur
• Technische Architektur
Vorbeugen von Erosion
• Kommunikation der Architektur und Umsetzungsrichtlinien
Vorbeugen von Erosion
• Validierung der Umsetzung


Erarbeitung der Architektur
• Fachliche Architektur
Erarbeitung der Architektur
• Technische Architektur
Vorbeugen von Erosion
• Kommunikation der Architektur und Umsetzungsrichtlinien
Vorbeugen von Erosion
• Validierung der Umsetzung

Warum?

Was?

SOFTWAREDOKUMENTATION

Wie?

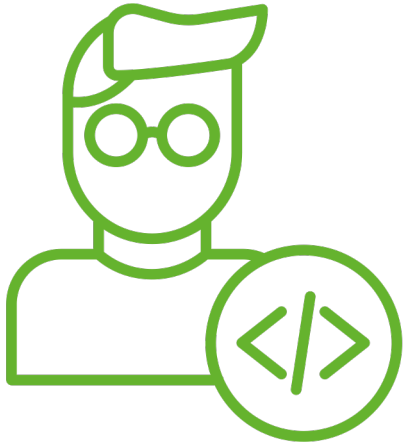
“Large documents are never kept up to date.”

Michael Nygard (2011)

“Documents that assist the team itself can have value, but only if they are kept up to date.”

“Nobody ever reads large documents, either.”

Icons made by Kiranshastry from Flaticon



Warum?

Blind akzeptieren oder ändern ist riskant

Icons made by Kiranshastry from Flaticon

Nur eine bekannte Architektur ist

eine umsetzbare Architektur

- Dokumentation muss über aktuelle Richtlinien informieren
  - Leicht konsumier- und produzierbar
  - Nachvollziehbare Änderungshistorie

- Dokumentation muss Relevanz für die Zielgruppe haben
  - Entwickler
  - Architekten
  - Requirements Engineers
  - ...

- Zielgruppen haben unterschiedliche Anforderungen
  - Bsp: Entwickler
    - Wie ist die Anwendung strukturiert?
    - Welche Architektur- und Designvorgaben sind zu befolgen?
    - Wie ist die Architektur im Code umzusetzen/umgesetzt?

- Bausteine guter (Architektur-)Dokumentation
  - Fachlicher Kontext und Kurzeinführung
  - Qualitätsziele und Rahmenbedingungen
  - Fachliche Architektur und Dekomposition
  - Technische Architektur und Entscheidungen

- Grundlage für Architektur-dokumentation: Arc42
  - https://www.arc42.de/overview/

1. Einführung und Ziele
   1. Aufgabenstellung
   2. Qualitätsziele
   3. Stakeholder
2. Randbedingungen
   1. Technische Randbedingungen
   2. Organisatorische Randbedingungen
   3. Konventionen
3. Kontextabgrenzung
   1. Fachlicher Kontext
   2. Technischer- oder Verteilungskontext
4. Lösungsstrategie
5. Bausteinsicht
   1. Ebene 1
   2. Ebene 2
   3. …
6. Laufzeitsicht
   1. Laufzeitszenario 1
   2. Laufzeitszenario 2
   3. …
7. Verteilungssicht
   1. Infrastrukturbene 1
   2. Infrastrukturebene 2
   3. …
8. Querschnittliche Konzepte
   1. Fachliche Struktur und Modelle
   2. Architektur- und Entwurfsmuster
   3. …
9. Entwurfsentscheidungen
   1. Entwurfsentscheidung 1
   2. Entwurfsentscheidung 2
   3. …
10. Qualitätsanforderungen
    1. Qualitätsbaum
    2. Qualitätsszenarien
11. Risiken und Technische Schulden
12. Glossar

Warum?

Was?

SOFTWAREDOKUMENTATION

Wie?

- Dokumentation wie Code behandeln
  - Dokumentation versioniert im Source Repository
  - Dokumentation versionierbar mit AsciiDoc oder Markdown
  - Diagramme möglichst generieren (PlantUML, ContextMapper, ...)

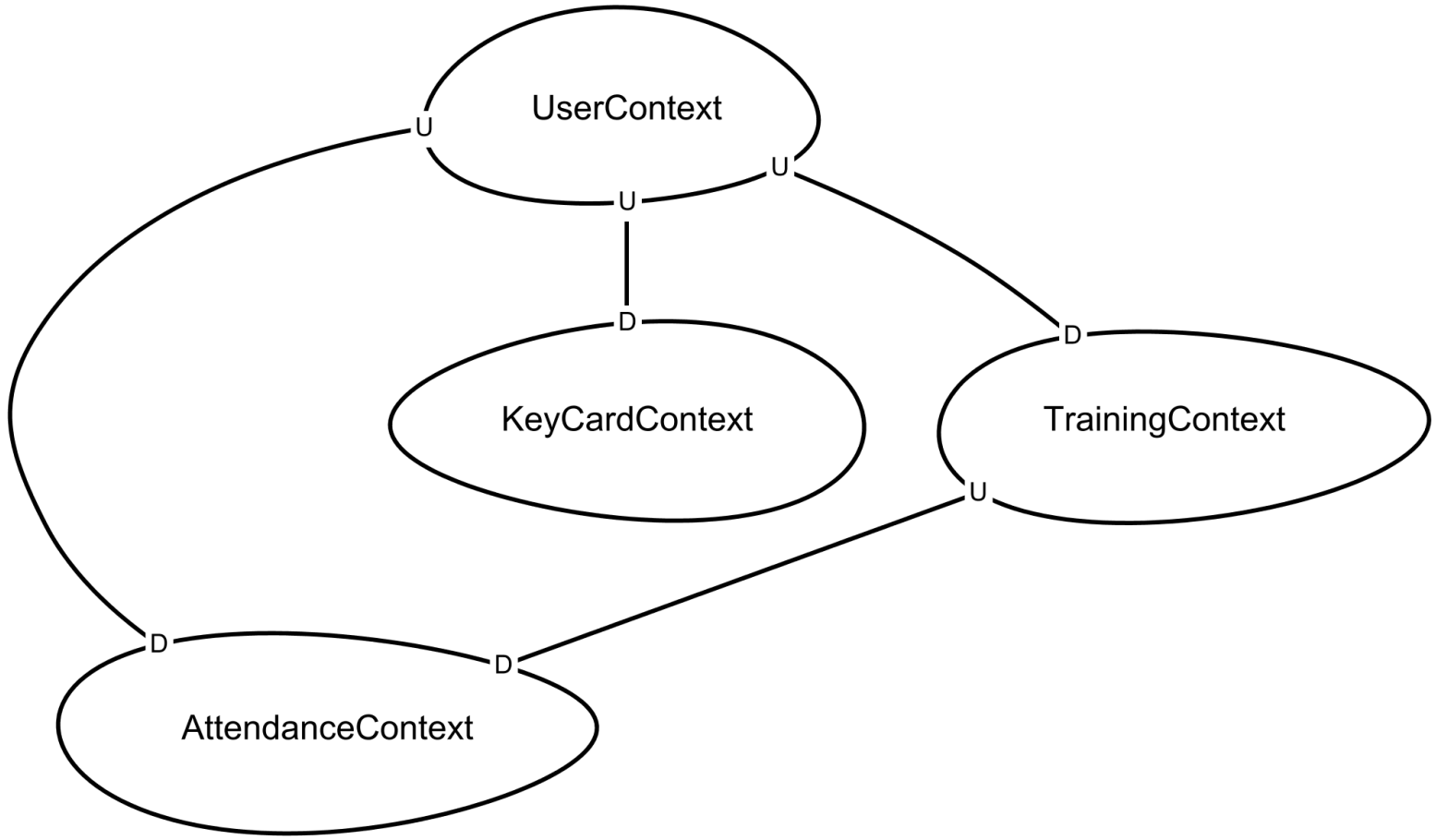
- Dokumentation wie Code behandeln
  - Leichte Auffindbarkeit und Änderbarkeit durch Entwickler/Architekten
  - Gesteigerte Motivation zum lesen UND schreiben durch Code-nähe
  - Tagesaktuelle Dokumentation durch Rendering während des Builds
    - Hosting zum Beispiel als Maven Site für nicht-technisches Publikum

- Dokumentation wie Code behandeln
  - Arc42 steht als AsciiDoc-Template zur Verfügung
  - Ablage unter /documentation im Projekt
    - Versioniert
    - Erreichbar
    - Nachvollziehbar

Live-Demo – Fachliche Architektur mit Context Maps

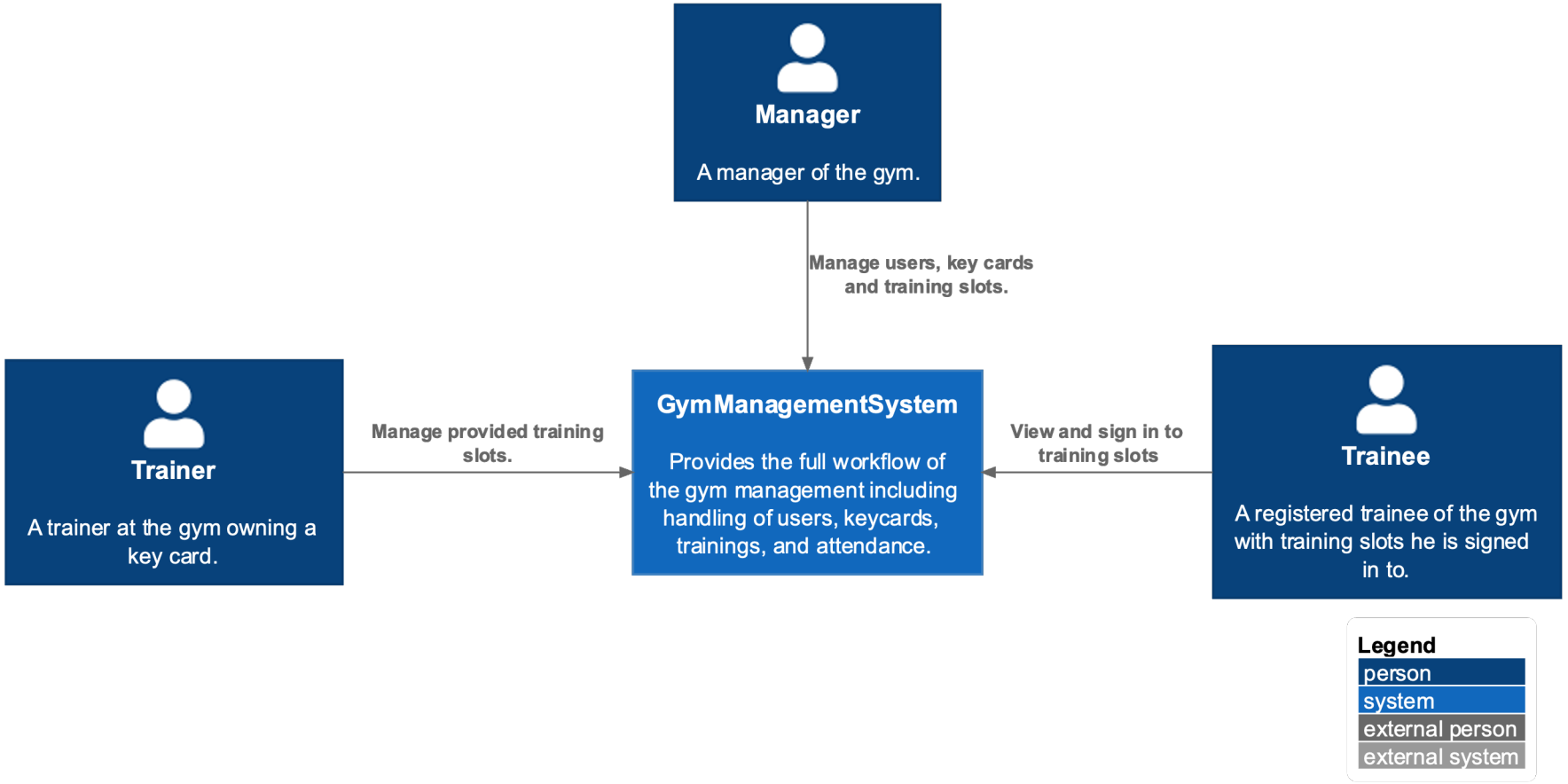
```
[[section-system-scope-and-context]]
== System Scope and Context

=== Business Context

…

==== Subdomains in the Gym Management System

image::context-mapper/Context-Map_ContextMap.png[]
```

Context-Map.cml



- Dokumentation der technischen Architektur ist Grundlage für
  - Umsetzung neuer Anforderungen durch Entwickler
  - Treffen neuer Entscheidungen bezüglich der Zukunft des Systems
  - Gemeinsame Kommunikation

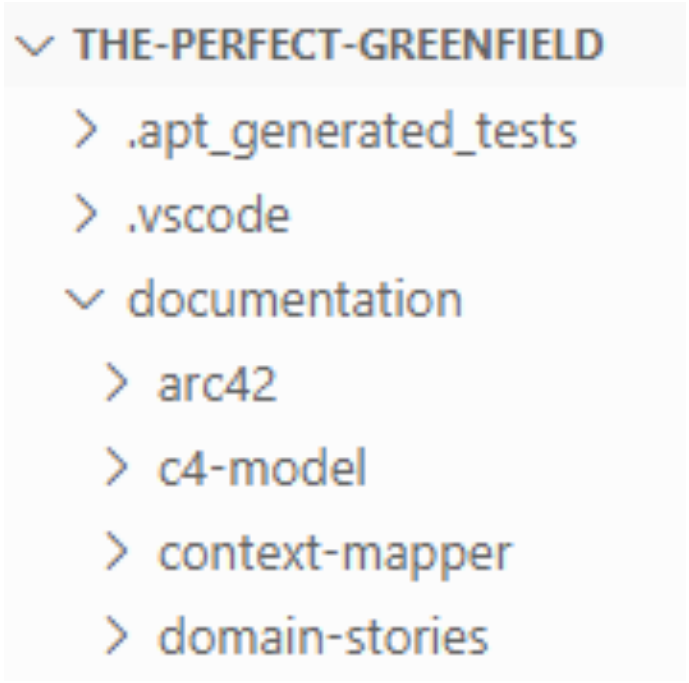
- Artefakte zum Beispiel erstellt mittels
  - PlantUML
  - Enterprise Architect / MagicDraw
    - wenn Features der Modellierung benötigt werden


Manager
A manager of the gym.
Manage users, key cards and training slots.
Trainer
A trainer at the gym owning a key card.
Manage provided training slots.
GymManagementSystem
Provides the full workflow of the gym management including handling of users, keycards, trainings, and attendance.
View and sign in to training slots
Trainee
A registered trainee of the gym with training slots he is signed in to.
Legend
person
system
external person
external system

Live-Demo – Technische Architektur mit dem C4-Model

```
[[section-building-block-view]]
== Building Block View

…
=== Level 1 - Context Diagram

[plantuml]
.Business Context of the Gym Management System
----
!include ../c4-model/Context-Diagram.puml
----
```

- Am Beispiel: [The Perfect Greenfield]

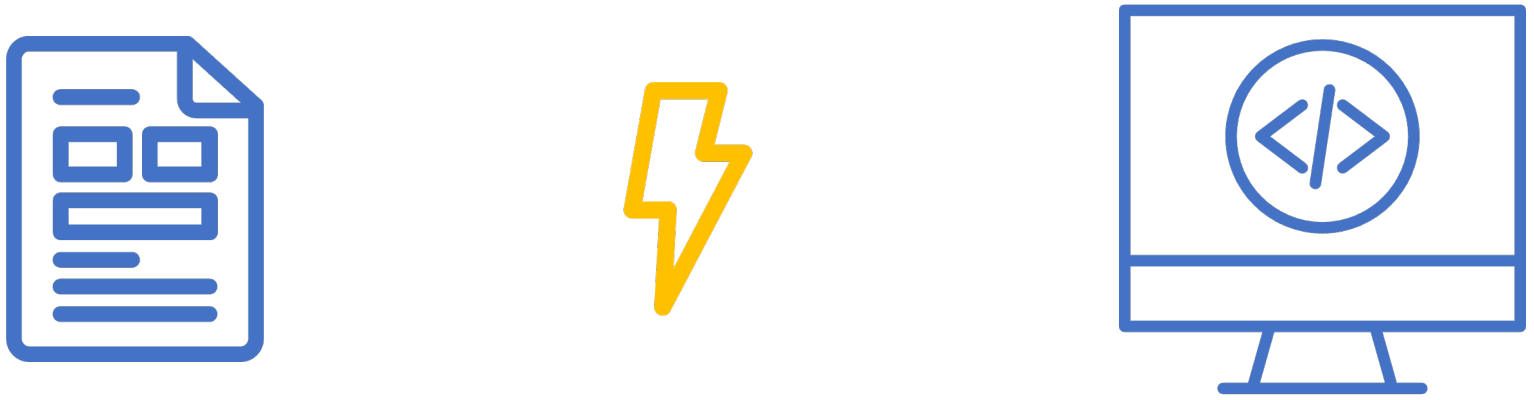


→ .adoc
→ .puml
→ .cml
→ .dst

Dokumentation schön und gut,

aber wer sorgt für die Einhaltung?

Documentation Code Gap



Icons made by Kiranshastry from Flaticon

- Zwei Möglichkeiten zum schließen der Gap
  1. Generieren von Dokumentation aus dem Source Code
  2. Automatisches Validieren des Codes gegen die Dokumentation

- Umsetzung mittels jQAssistant

jQAssistant ist ...

ein Open-Source Tool, welches sich ...

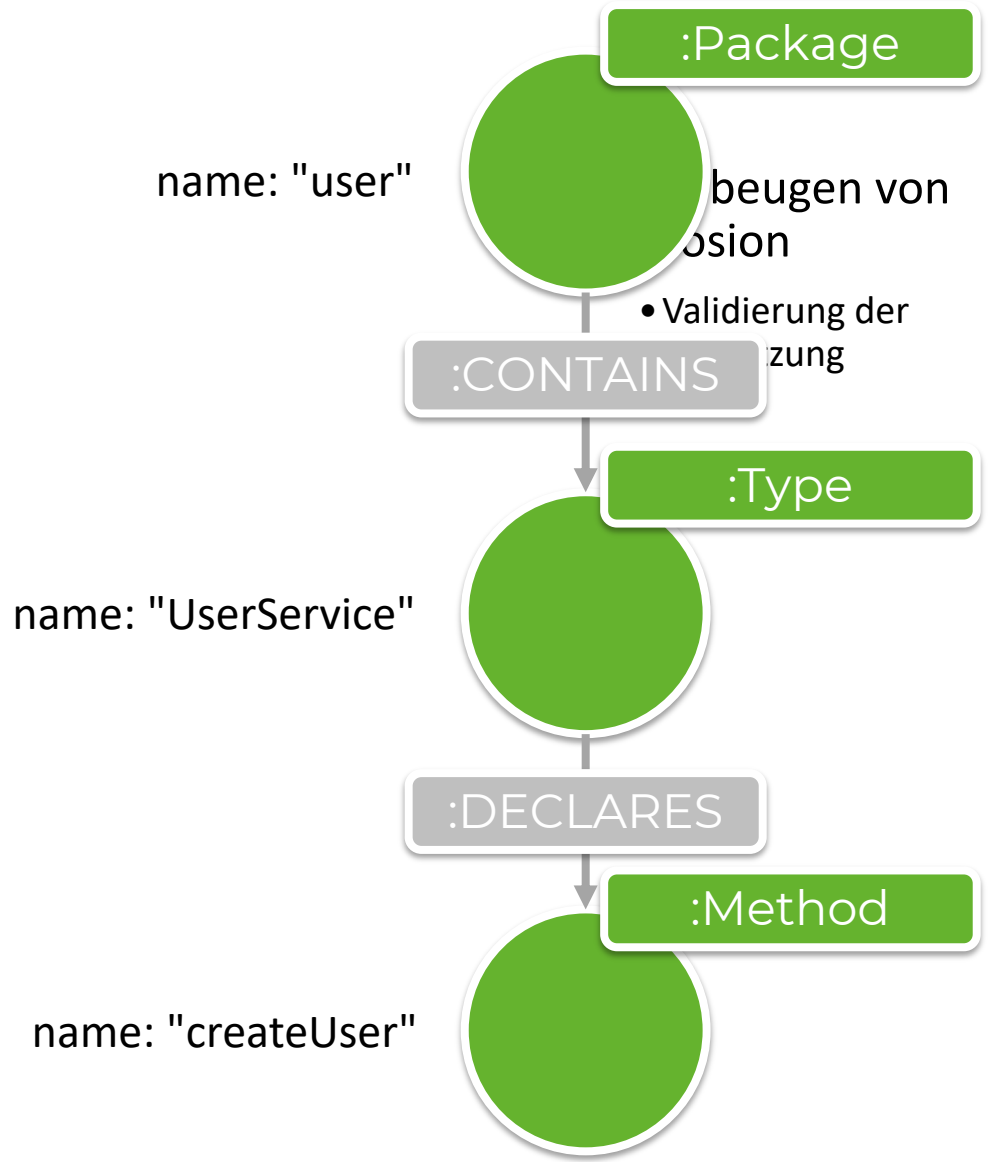
als Maven-Plugin in den Build-Prozess integriert und ...

verschiedene Quellen in eine Neo4j-DB einlesen, ...

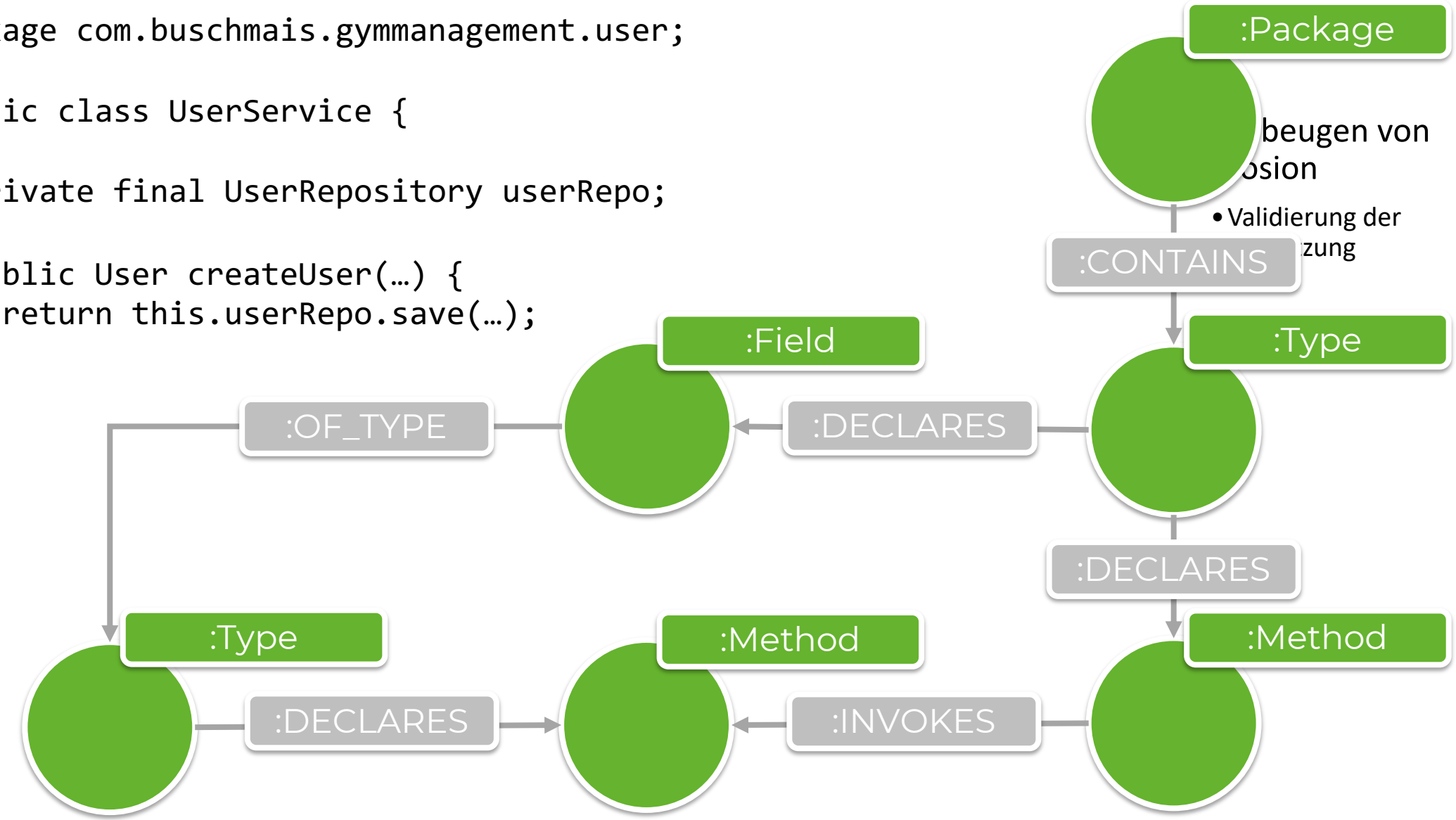
Strukturen (Architektur) validieren und ...

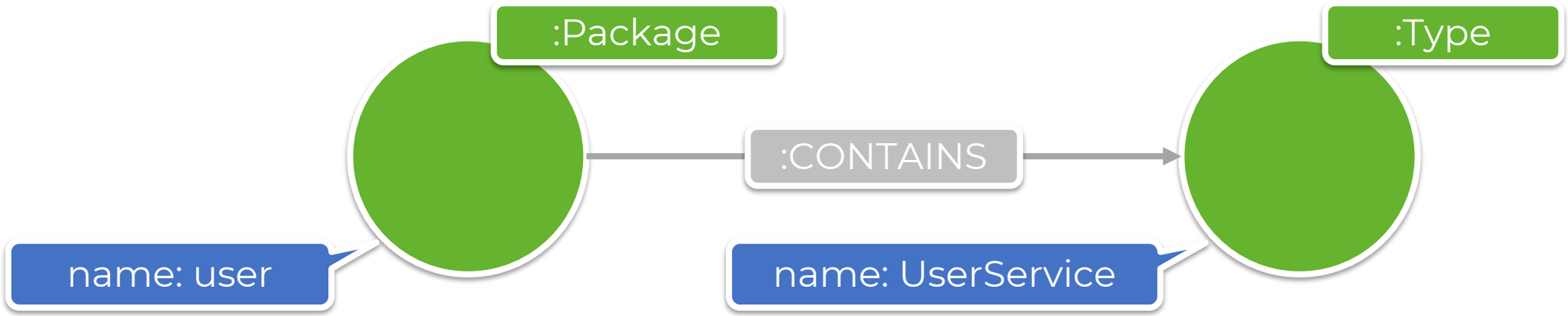
Dokumentation generieren kann

```
package com.buschmais.gymmanagement.user;

public class UserService {

  private final UserRepository userRepo;

  public User createUser(…) {
    return this.userRepo.save(…);
  }
}
```



```
package com.buschmais.gymmanagement.user;

public class UserService {

  private final UserRepository userRepo;

  public User createUser(…) {
    return this.userRepo.save(…);
  }
}
```






```
MATCH (p:Package)-[:CONTAINS]->(t:Type:Java)
WHERE  p.name = “user“ AND t.name = “UserService“
RETURN p, t
```

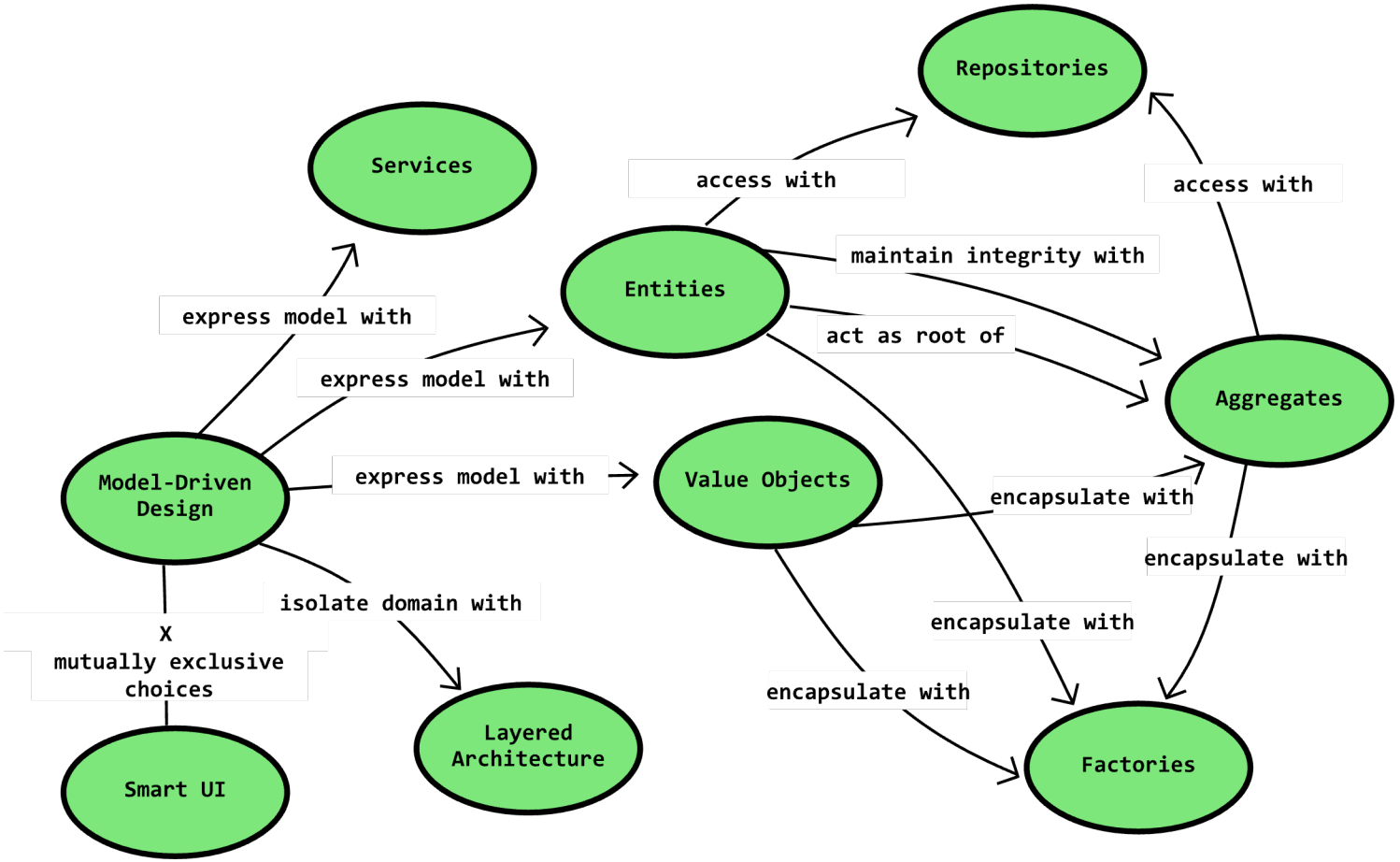
- Cypher-Statements können
  - Konzepte im Graph anreichern (“Concept”)
    - zusätzliche Labels, Beziehungen, Knoten
    - um diese später leichter verwenden zu können
  - Architekturvorgaben sicherstellen (“Constraint”)
    - Queries, welche ein nicht-leeres Ergebnis zurückgeben, gelten als verletzt
  - gruppiert werden (“Group”)

- Cypher-Statements können
  - in XML- oder AsciiDoc-Dokumenten geschrieben werden
  - bei verletzen den Build brechen
  - in Form von “Concepts” auch Diagramme generieren

Das Problem: Wie findet man Architektur im Code?

Hinter jeder Architektur liegt eine Mustersprache bestehend aus Konzepten und Beziehungen zwischen diesen Konzepten

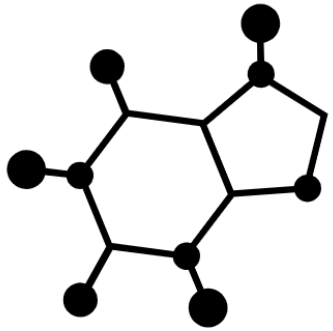
- Mustersprachen wie z.B.
  - Domain-driven Design
  - MVC
  - Gang of Four Patterns
  - individuelle Mustersprachen



https://khalilstemmler.com/articles/typescript-domain-driven-design/entities/

- Source Code muss **architekturexplizit** sein
  - klare Erkennbarkeit von Konzepten → leichtes konsumieren
  - klare Umsetzungsrichtlinieren → leichtes produzieren

Wichtig: Explizite Umsetzung von Konzepten darf nicht zusätzliche Komplexität bedeuten



xMolecules

Architectural abstractions in code

No all code is born equal. While some elements of a code base are of supportive nature, others represent significant architectural concepts and design patterns. Traditionally, developers have tried to express those concepts through naming conventions.

xMolecules is an umbrella project providing means to express architectural concepts in programming language native metadata and – if applicable – right within the type system.

→ **jMolecules** – for Java
→ **nMolecules** – for .NET
→ **phpMolecules** – for PHP

- jMolecules bietet Annotation und Abstraktionen
  - Domain-driven Design
  - Event-driven Architektur
  - CQRS
  - Layered + Onion Architectures

Architektur explizit implementiert ✓

Das war’s?

- jMolecules bietet ByteBuddy Integration
  - Generierung von JPA-Annotationen
  - Generierung von Spring-Annotation

- https://github.com/xmolecules/jmolecules-integrations

Architektur explizit implementiert ✓

Technische Komplexität reduzieren ✓

Wir bringen es zusammen:

Teil 1 – Bounded Contexts

- Ziel: Korrekte Umsetzung der Context Map
  - Herstellung der Identifizierbarkeit von Bounded Contexts
  - Überprüfung der Abhängigkeiten zwischen Bounded Contexts

9.1.5. 003 - Implement Bounded Contexts using jMolecules on Package-level

Status: Accepted

Context

The structure of the Gym Management System is heavily based on the functional decomposition.

Following the functional decomposition is required to keep maintenance and extension efforts low, have clarity about the location of the implementation, and, in case the application grows, allow for easier split of development teams.

Decision

Bounded Contexts need to be implemented as Java-packages inside the main-package `com.buschmais.gymmanagement`.

Each of these packages needs to be annotated with `o.j.d.a.BoundedContext` via a `package-info.java`. The annotation must specify the name of the bounded context completely, e.g. "UserContext".

Consequences

Implementation of features requires to follow the dependency directions as documented in section 3 of the ADR.

Table 3. Constraints

Id	Description	Severity	Status
adr:IllegalDependenciesBetweenBoundedContext	Labels all gym management artifacts and files with :GymManagement	MAJOR	SUCCESS

Wir bringen es zusammen:

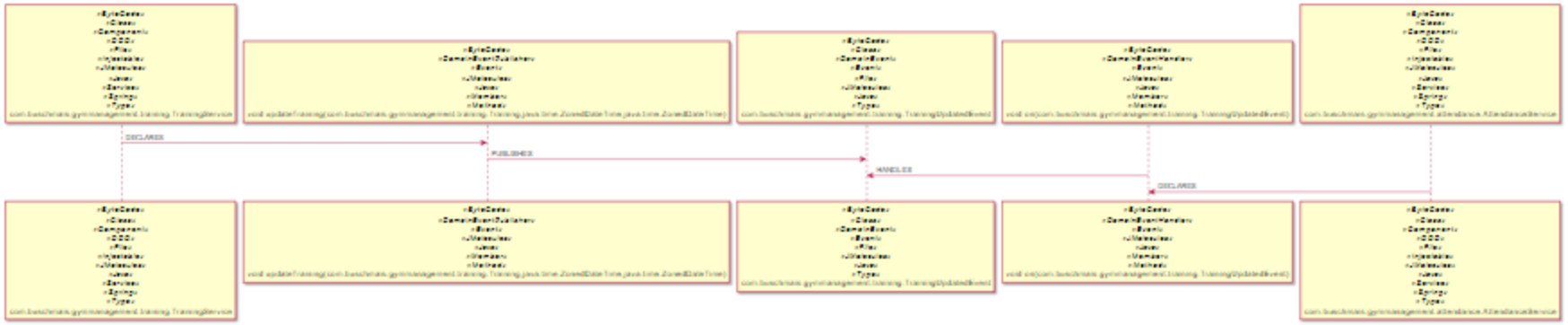
Teil 2 – Generierung eines Runtime-View Diagramms

- Ziel: Dokumentation des Use Case Training Update
  - Implementierung des Use Cases “Training aktualisieren”
  - Architekturexpliziter Source Code
  - Generierung eines Sequenzdiagramms

6. Runtime View

✔ *Visualization of the* `TrainingUpdateEvent` *flow* ☑

```
MATCH sequence=(t:Type{name: "TrainingService"})-[:DECLARES]->(:DomainEventPublisher)-
[:PUBLISHES]-()<-[:HANDLES]-(:DomainEventHandler)<-[:DECLARES]-()
RETURN nodes(sequence) AS participants,
       relationships(sequence) AS messages
```



BEST PRACTICES UND HINWEISE

- Schnelles Feedback für Entwickler ist wichtig
  - Lokale Ausführung von jQAssistant vor “git push --force” ;)
  - Integration in Build-Pipeline (insbesondere Feature-Branches)
  - Integration mit SonarQube (Plugin verfügbar)
  - Hosting der Maven Site zum Zugriff für alle

- Je eher, desto besser
  - Frühzeitige, versionierte Dokumentation
  - Frühzeitige Festlegung der Umsetzung
  - Frühzeitige Generierung von Dokumentation
  - Frühzeitige automatisierte Prüfung der Architektur

1. Einführung und Ziele

- Domain Stories
- ISO 25010

2. Randbedingungen

3. Kontextabgrenzung

4. Lösungsstrategie

5. Bausteinsicht

- C4-Model (Ebene 1 – 3)

6. Laufzeitsicht

- PlantUML Sequenz-diagramme

7. Verteilungssicht

- C4-Model (Deployment)

8. Querschnittliche Konzepte
- PlantUML Klassendiagramm

9. Entwurfsentscheidungen

- Architecture Decision Records

10. Qualitätsanforderungen

11. Risiken und Technische Schulden
- jQAssistant Reports

12. Glossar

- Ressourcen
  - ADR-Starter:

    https://github.com/buschmais/adr-starter
  - Software Analytics-Starter:

    https://github.com/buschmais/software-analytics-starter
  - The Perfect Greenfield:

    https://github.com/buschmais/The-Perfect-Greenfield

- Ressourcen
  - jQAssistant 101s:

    https://101.jqassistant.org/startpage/
  - Hands-Ons:

    https://vimeo.com/buschmais

- Resourcen
  - Context Maps → https://contextmapper.org/
  - C4-Model → https://github.com/plantuml-stdlib/C4-PlantUML
  - Strukturierung → https://www.arc42.de/
  - Architekturentscheidungen → https://git.io/JuMQk

VIELEN DANK!

WEITERE LINKS

- Case Study ONE DATA – BIG DATA MEETS CLOUD
- Artikel JavaSPEKTRUM Modernization Work Ahead
- Artikel OBJEKTspektrum Pragmatisch, praktisch, erfolgreich mit ADRs!
- BUSCHMAIS @Social Media:
  - LinkedIn
  - Twitter
  - Instagram
- Unsere nächsten BUSCHMAIS-Termine

BUSCHMAIS

KONTAKT

Stephan Pirnbaum

stephan.pirnbaum@buschmais.com

Tel. +49 351 320923-22

Twitter: @spirnbaum

BUSCHMAIS GbR

Leipziger Straße 93

01127 Dresden

Tel. +49 351 3209230

info@buschmais.com

www.buschmais.de