Orientation in Objects

Agile Architektur

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.1

Abstract

Eine gut geplante Softwarearchitektur stellt das Grundgerüst jeder wartbaren Software dar. Das steht in einem scheinbaren Widerspruch zu agilen Softwareprozessen, bei denen langfristige Planungen weitestgehend vermieden werden.

Die Session zeigt anhand zahlreicher konkreter Beispiele, wie man die Erstellung einer Softwarearchitektur in einzelne Aufgabenpakete unterteilt, dokumentiert und regelmäßig durch Reviews und automatisierte Architekturtests verifiziert. Das Ziel ist es, auch in einem iterativen Prozess eine langfristig tragfähige Architektur entstehen zu lassen.

Ihre Sprecher

Orientation in Objects

Thorsten Maier, Falk Sippach

Trainer, Berater, Entwickler

@ThorstenMaier

@sippsack

Orientation in Objects

Agiles Manifest

=

Individuen und Interaktionen *mehr als* ***Prozesse und Werkzeuge***

Funktionierende Software *mehr als* ***umfassende Dokumentation***

Zusammenarbeit mit dem Kunden *mehr als* ***Vertragsverhandlung***

Reagieren auf Veränderung *mehr als* ***das Befolgen eines Plans***

Orientation in Objects

Architektur in eigenen Worten:

fundamentale Strukturen,
Konzepte,
Entscheidungen
und Lösungsansätze

… die man nicht mehr leicht los bekommt!

Orientation in Objects

Agile Architektur

Widerspruch?

Orientation in Objects

Unsere These

Eine gute Architektur kann iterativ in kleinen Aufgabenpaketen entstehen.

Orientation in Objects

Agiles Manifest

***Die BESTEN Architekturen** […] entstehen*
durch selbstorganisierte Teams.

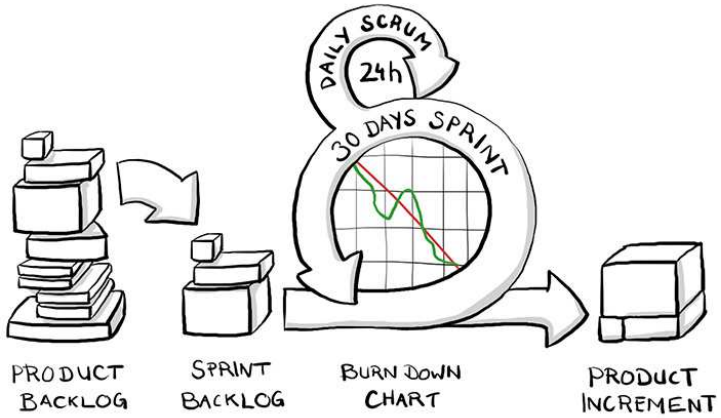
(11. Prinzip der agilen Softwareentwicklung)

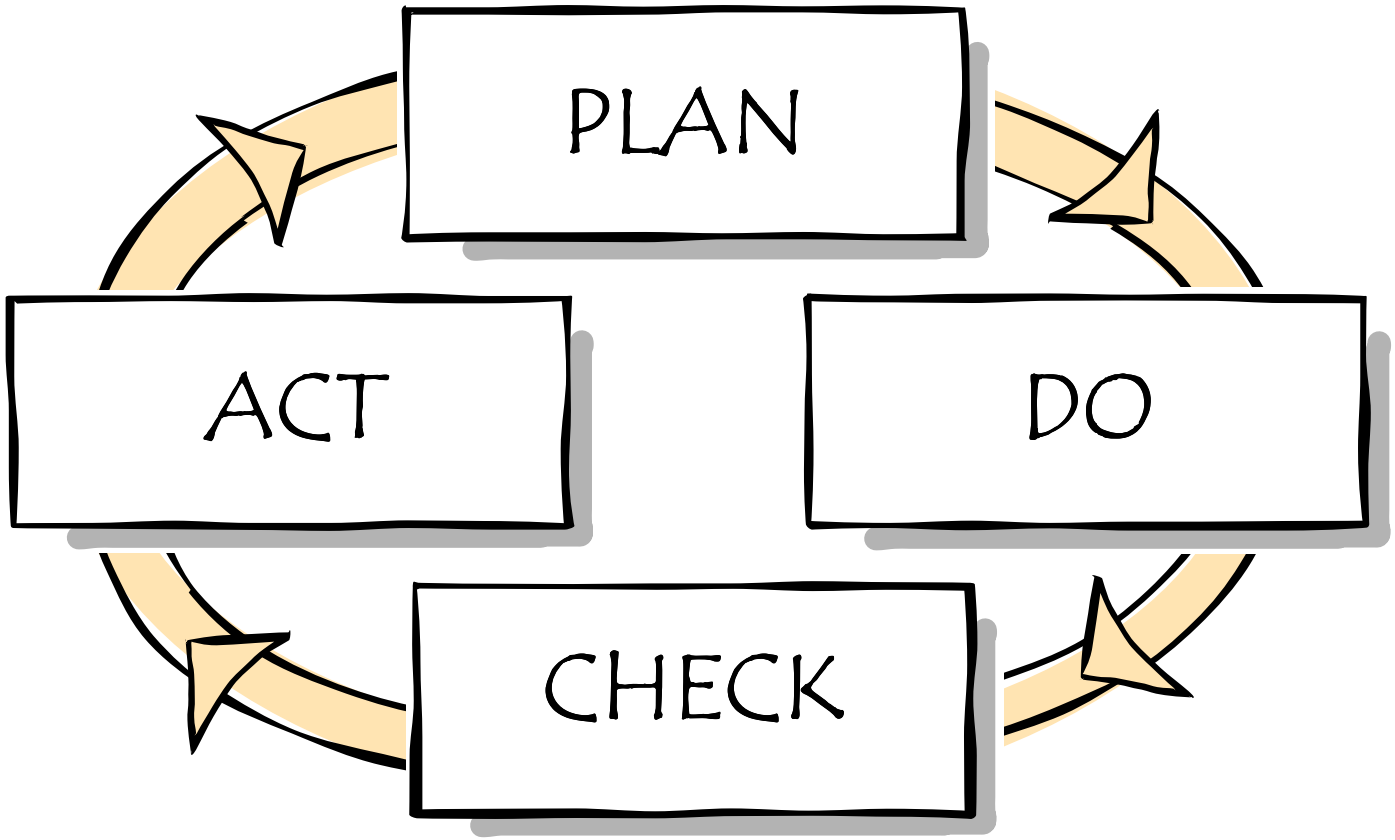
Herausforderung in einem agilen Prozess

Architekturarbeit aufteilen

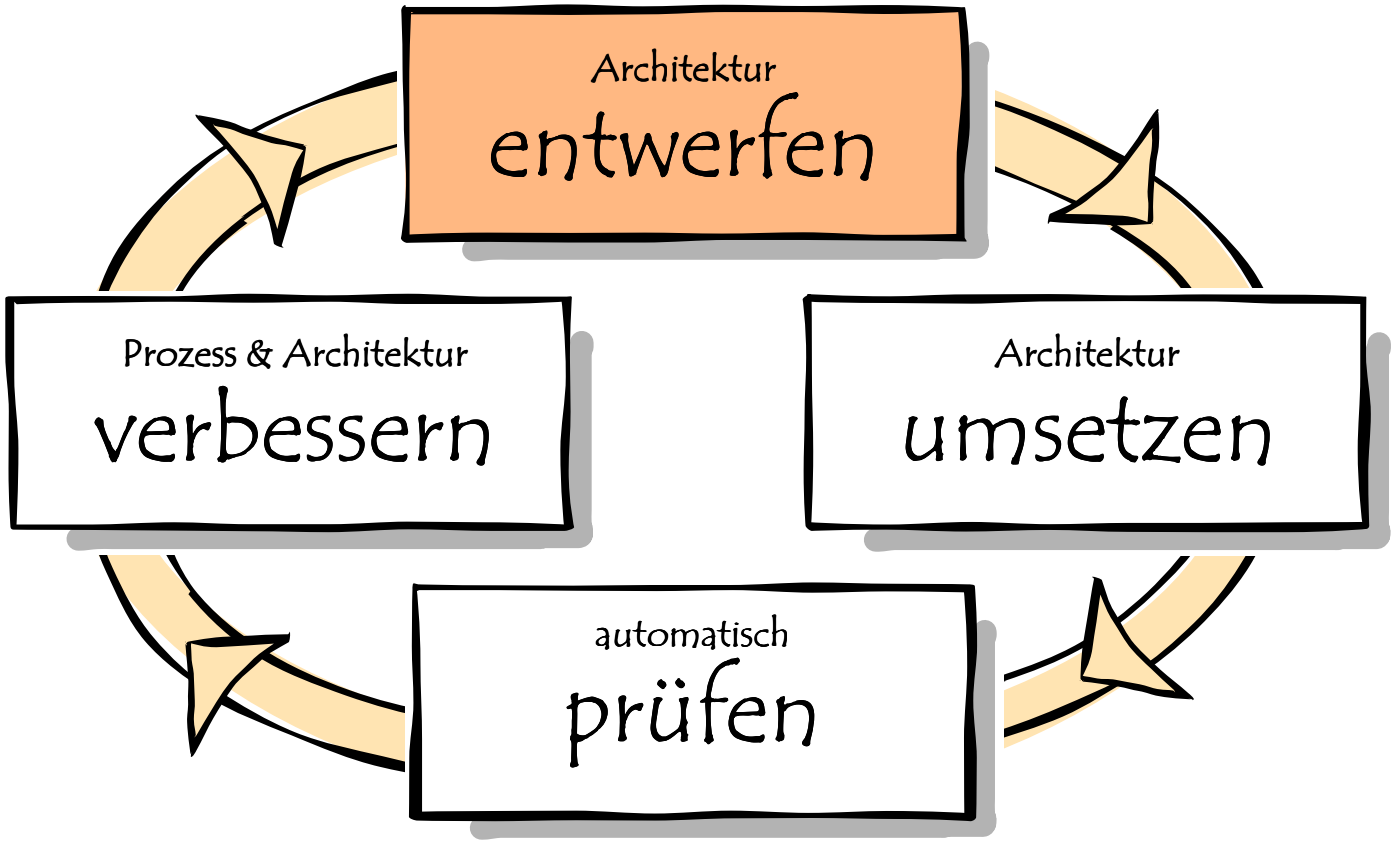
Darum geht es heute

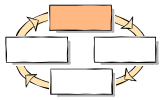
12 konkrete Arbeitspakete




PLAN
DO
CHECK
ACT

Orientation in Objects



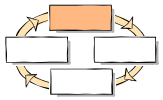


Orientation in Objects

1

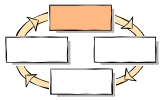
Nicht-funktionale Anforderungen konkretisieren

→ Software-Architektur



Orientation in Objects

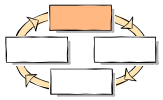
„Benutzerfreundlich“
„Schnell“
„Wartbar“



Orientation in Objects

Nicht-funktionale Anforderungen

müssen prüfbar sein



Orientation in Objects

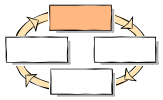
„Benutzerfreundlich“

=

Nachbearbeitung eines **Inkasso-Anrufes**

darf maximal **30 Sekunden** dauern und muss

nach **3 Klicks** abgeschlossen sein.

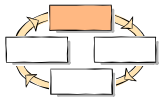


Orientation in Objects

„Wartbar“

=

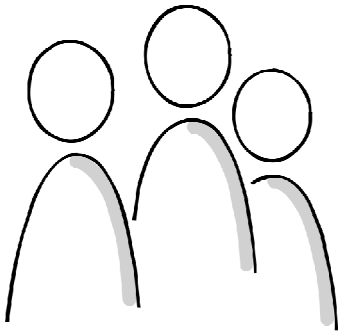
Ein **neues Krankenversicherungsprodukt** lässt sich mit einem durchschnittlichen Aufwand von **28 Personentagen** in die Anwendung integrieren.



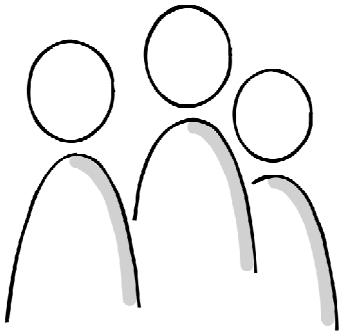
Orientation in Objects

2

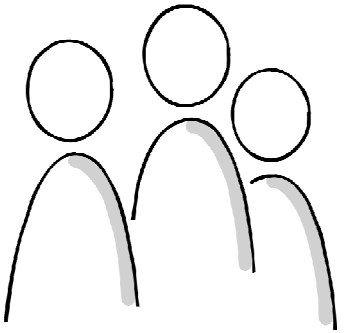
Stakeholder bewusst machen



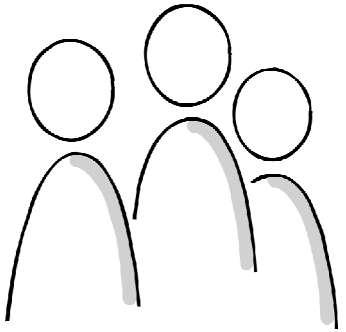
Leitung Entwicklung



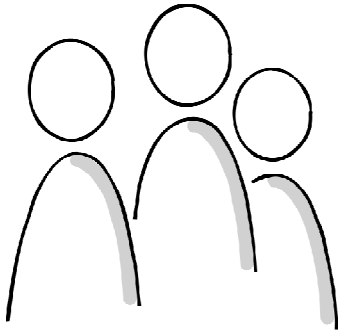
Leitung Wartung



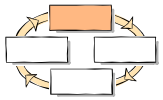
Kunde



Marketing



Endbenutzer



Orientation in Objects

Agiles Team

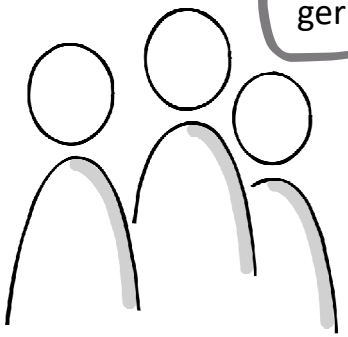
Geringe Kosten,
geringe Fluktuation

Wartbarkeit

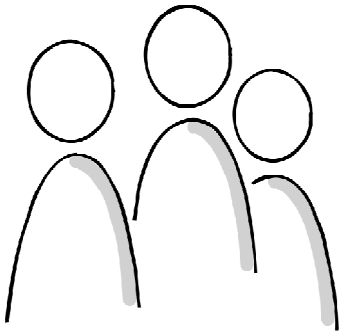
Geringe Kosten,
pünktliche Lieferung

Neue Feature
Time-to-market

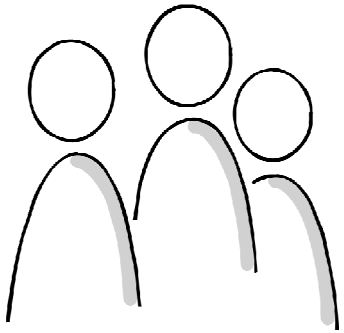
Schnell,
benutzerfreundlich



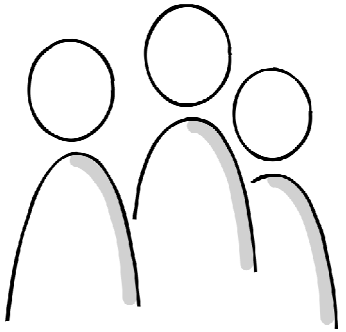
Leitung Entwicklung



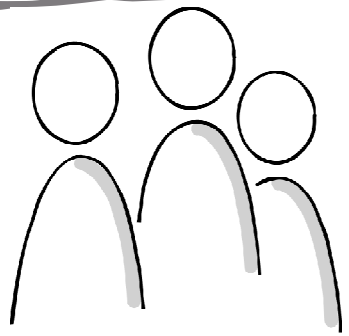
Leitung Wartung



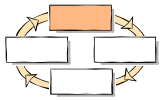
Kunde



Marketing



Endbenutzer

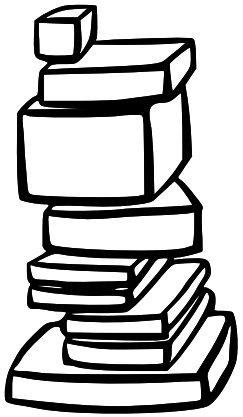


Orientation in Objects

3

Bausteine und Beziehungen entwerfen

Fachliche Anforderungen



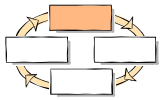
Abstraktion, Modularisierung, fachliche/vertikale Schnitte (Microservices)

Architektur

Lösungsstrategie

Technologieentscheidungen, Architekturstile/-prinzipien, Muster, Querschnittsaspekte

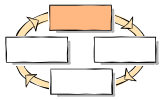
Randbedingungen
Qualitätsziele



Orientation in Objects

4

Architektur bewerten



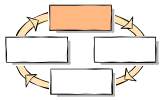
Orientation in Objects

Theorie:

Architekturen früh vergleichbar

Praxis:

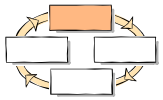
Analyse wird während Schadensbegrenzung spät im Projekt durchgeführt

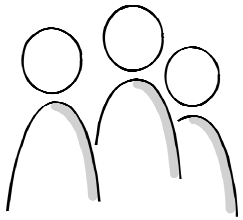


Orientation in Objects

Szenariobasierte Architekturbewertung

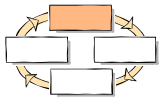
SAAM – Software architecture analysis method
ATAM – Architecture tradeoff analysis method
ACDM – Architecture-centric design method





Orientation in Objects

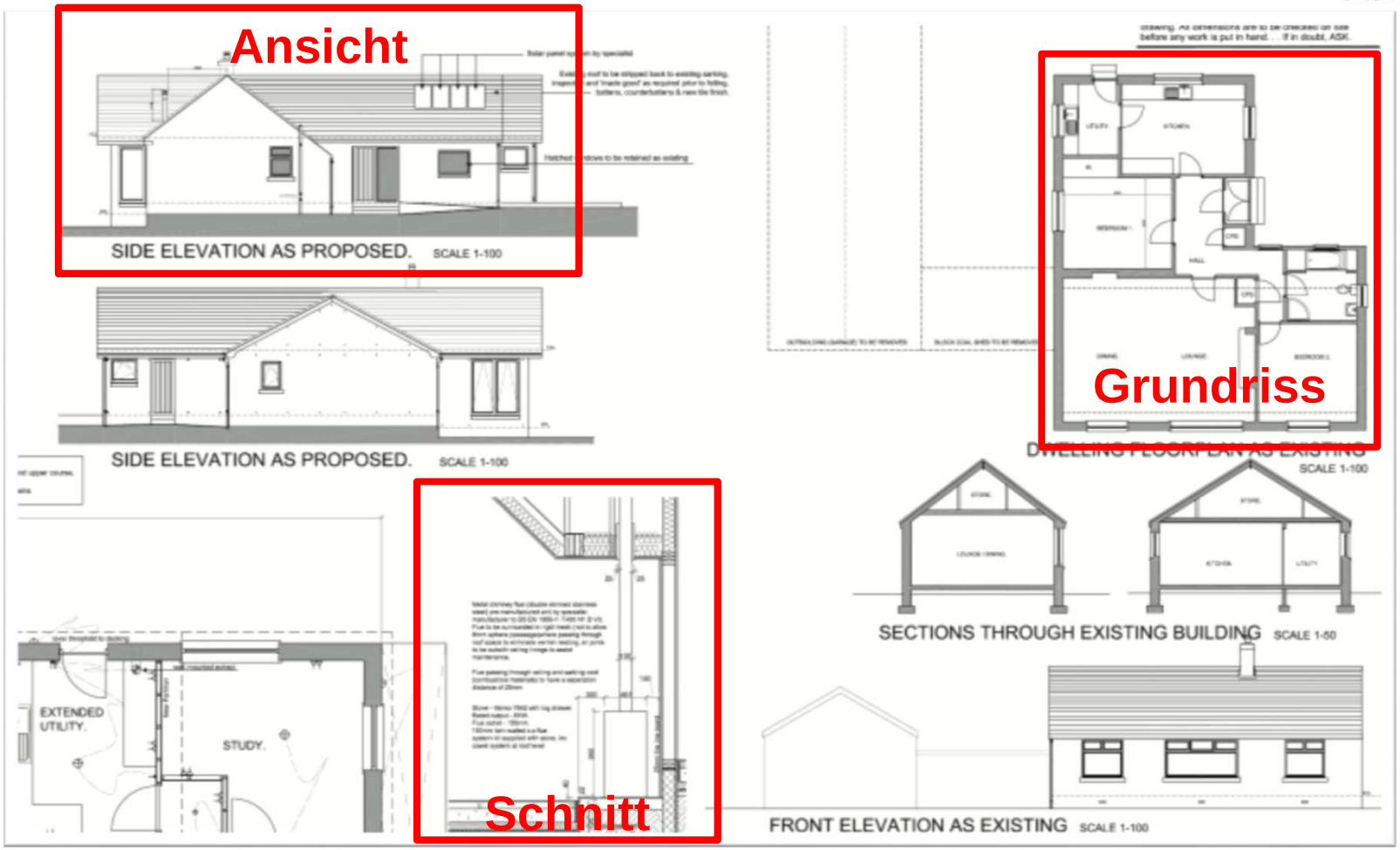
Stakeholder	Architektur 1		Architektur 2	
	Komponenten	Aufwand	Komponenten	Aufwand
Neues Krankenversicherungsprodukt	3	28d	2	20d
DHL statt Hermes	-	-	4	12d
...				
Ergebnis	**3**	**28d**	**6**	**32d**

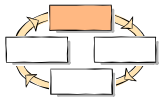


Orientation in Objects

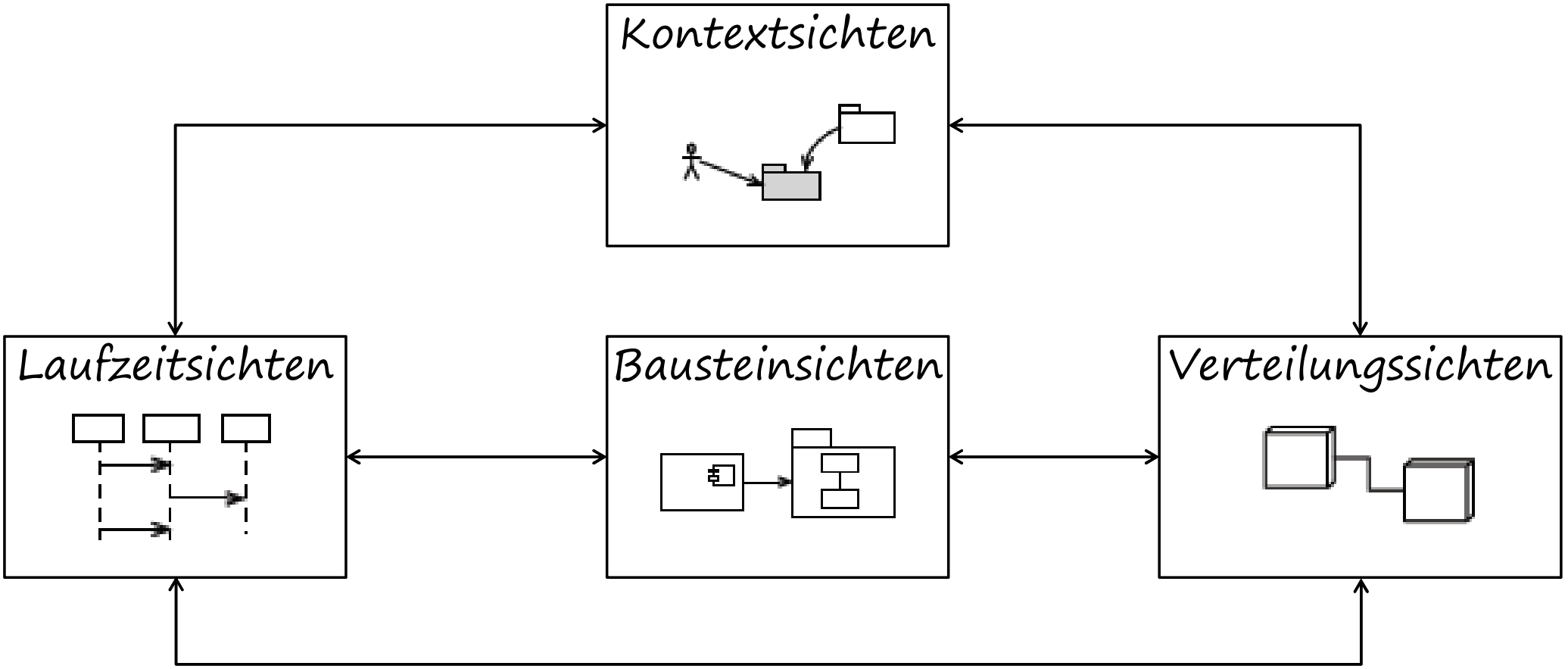
5

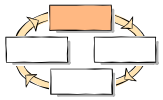
Architektur dokumentieren


Ansicht
SIDE ELEVATION AS PROPOSED. SCALE 1-100
SIDE ELEVATION AS PROPOSED. SCALE 1-100
Grundriss
SCALE 1-100
SECTIONS THROUGH EXISTING BUILDING SCALE 1-50
EXTENDED UTILITY.
STUDY.
Schnitt
FRONT ELEVATION AS EXISTING SCALE 1-100



Orientation in Objects

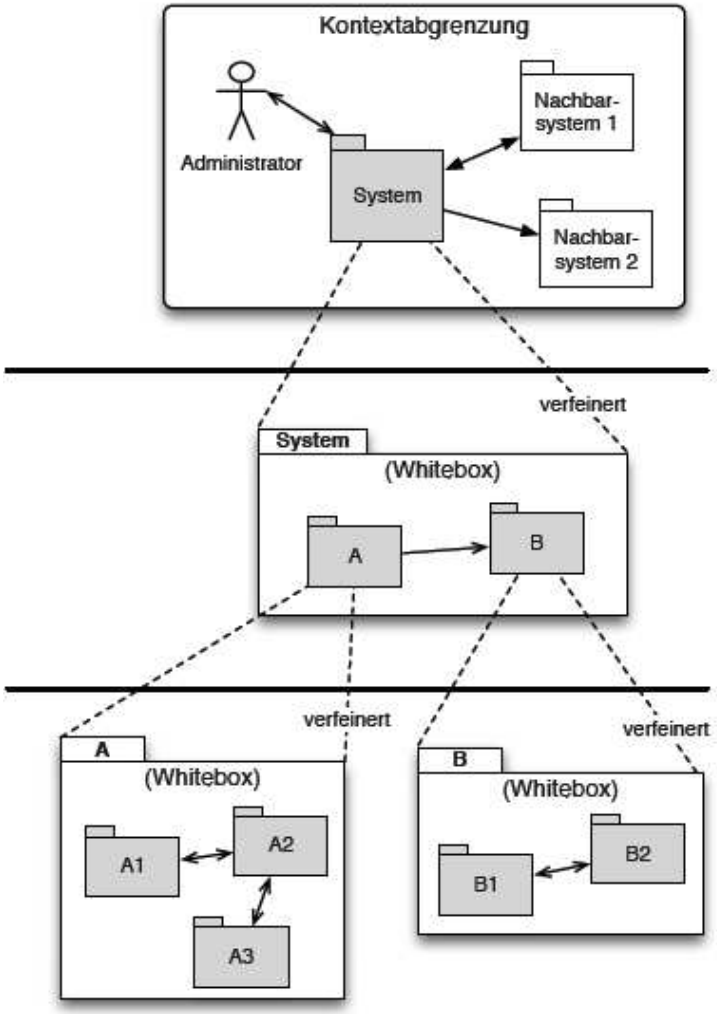




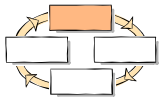
Level 0

Level 1

Level 2



Nicht zu detailliert, sondern pragmatisch und effizient!



Orientation in Objects

1. Einführung und Ziele
1.1 Aufgabenstellung
1.2 Qualitätsziele
1.3 Stakeholder

2. Randbedingungen
2.1 Technische Randbedingungen
2.2 Organisatorische Randbed.
2.3 Konventionen

3. Kontextabgrenzung
3.1 Fachlicher Kontext
3.2 Technischerkontext

4. Lösungsstrategie

5. Bausteinsicht
5.1 Ebene 1
5.2 Ebene 2
....

6. Laufzeitsicht
6.1 Laufzeitszenario 1
6.2 Laufzeitszenario 2
....

7.Verteilungssicht
7.1 Infrastruktur Ebene 1
7.2 Infrastruktur Ebene 2
....

8. Konzepte
8.1 Fachliche Struktur und Modelle
8.2 Typische Muster und Strukturen
8.3 Persistenz
8.4 Benutzeroberfläche
....

9. Entwurfsentscheidungen
9.1 Entwurfsentscheidung 1
9.2 Entwurfsentscheidung 2
....

10. Qualitätsszenarien
10.1 Qualitätsbaum
10.2 Bewertungsszenarien

11. Risiken

12. Glossar

arc42

Templates für:

Microsoft Word

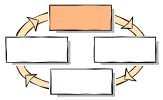
Confluence

Markdown, AsciiDoc

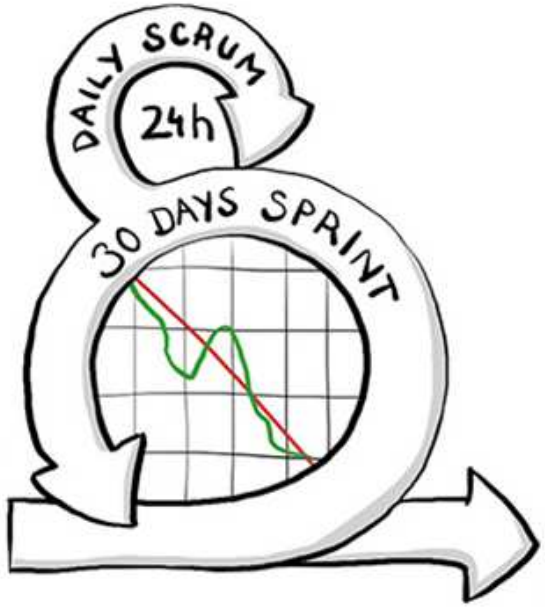
Latex, DocBook

HTML, EPUB

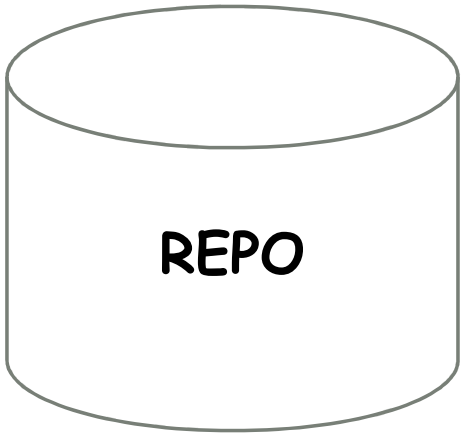
Textile



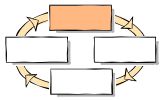
Orientation in Objects



Continuous Documentation



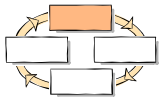
Documentation as Code



Orientation in Objects

6

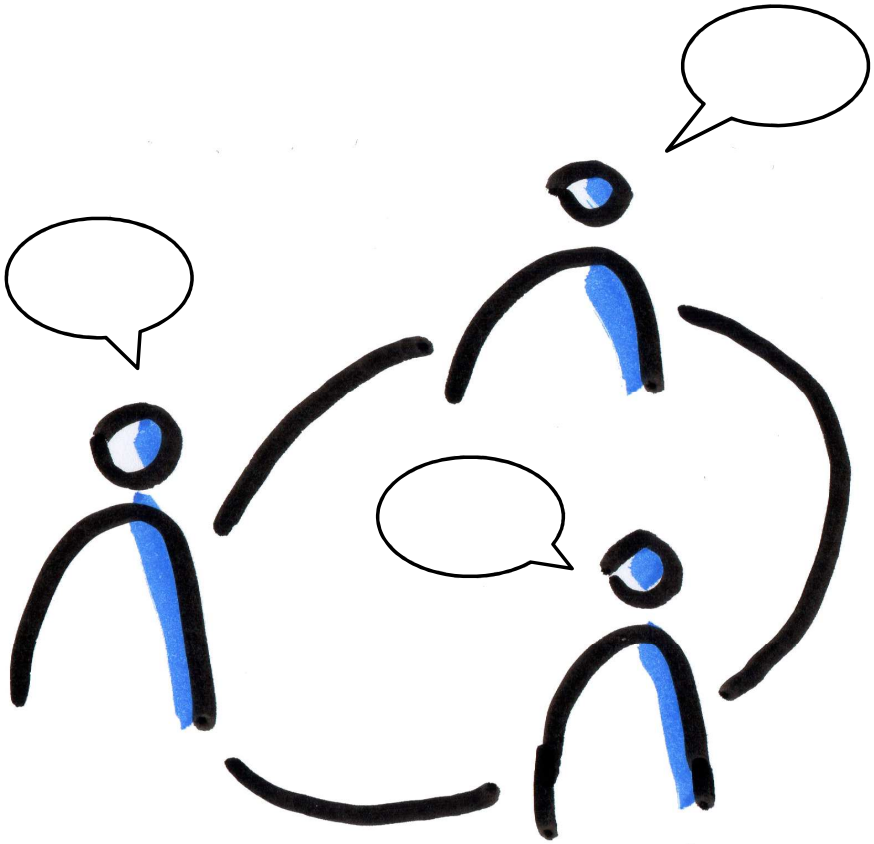
Architektur kommunizieren



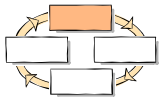
Orientation in Objects

"Man glaubt nur so lange, dass
***ein Entwurf perfekt ist**, bis man ihn*
***jemand anderem gezeigt hat**."*

(Software Architecture Documentation in Practice von Bachmann, Bass)



vs.

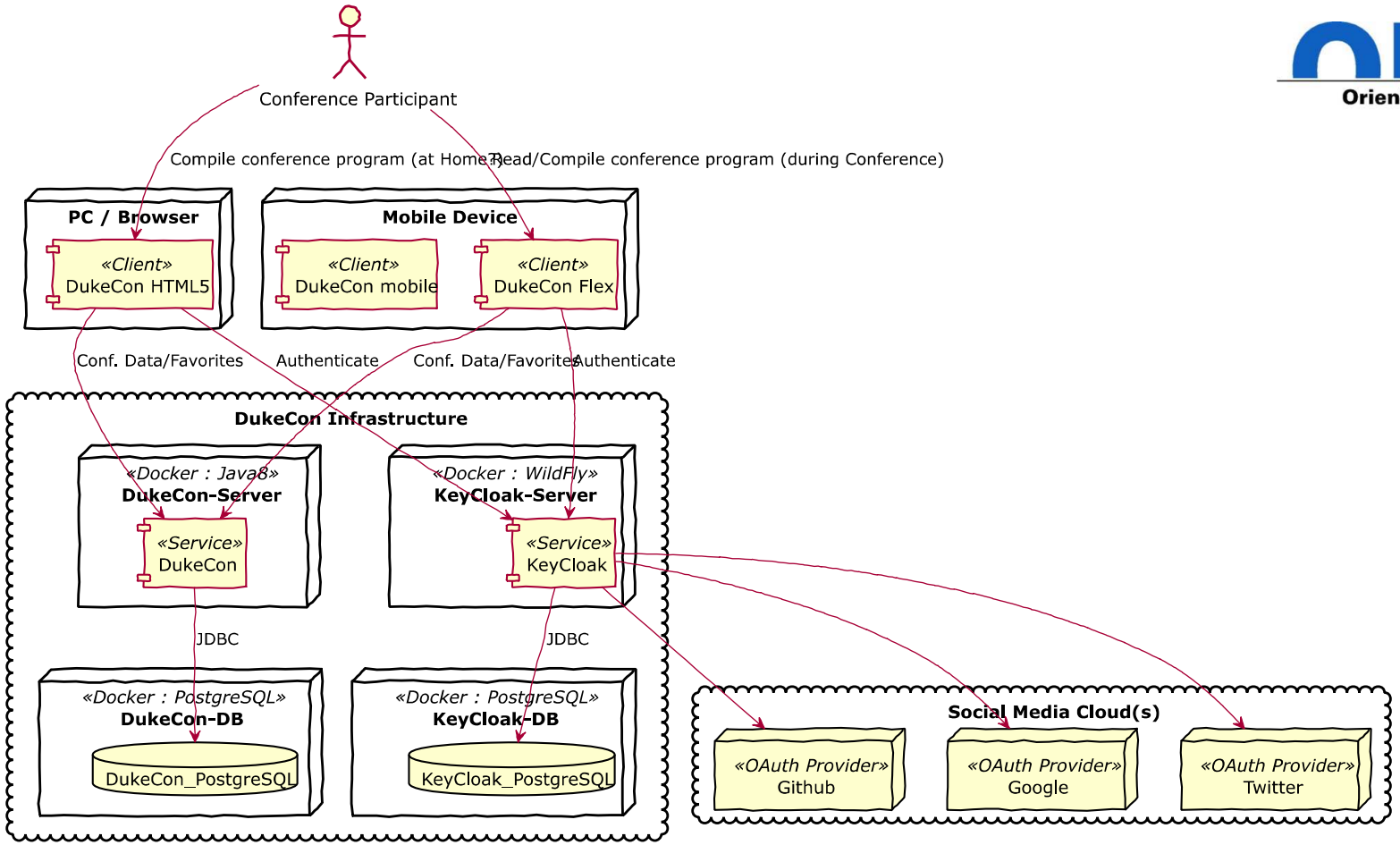


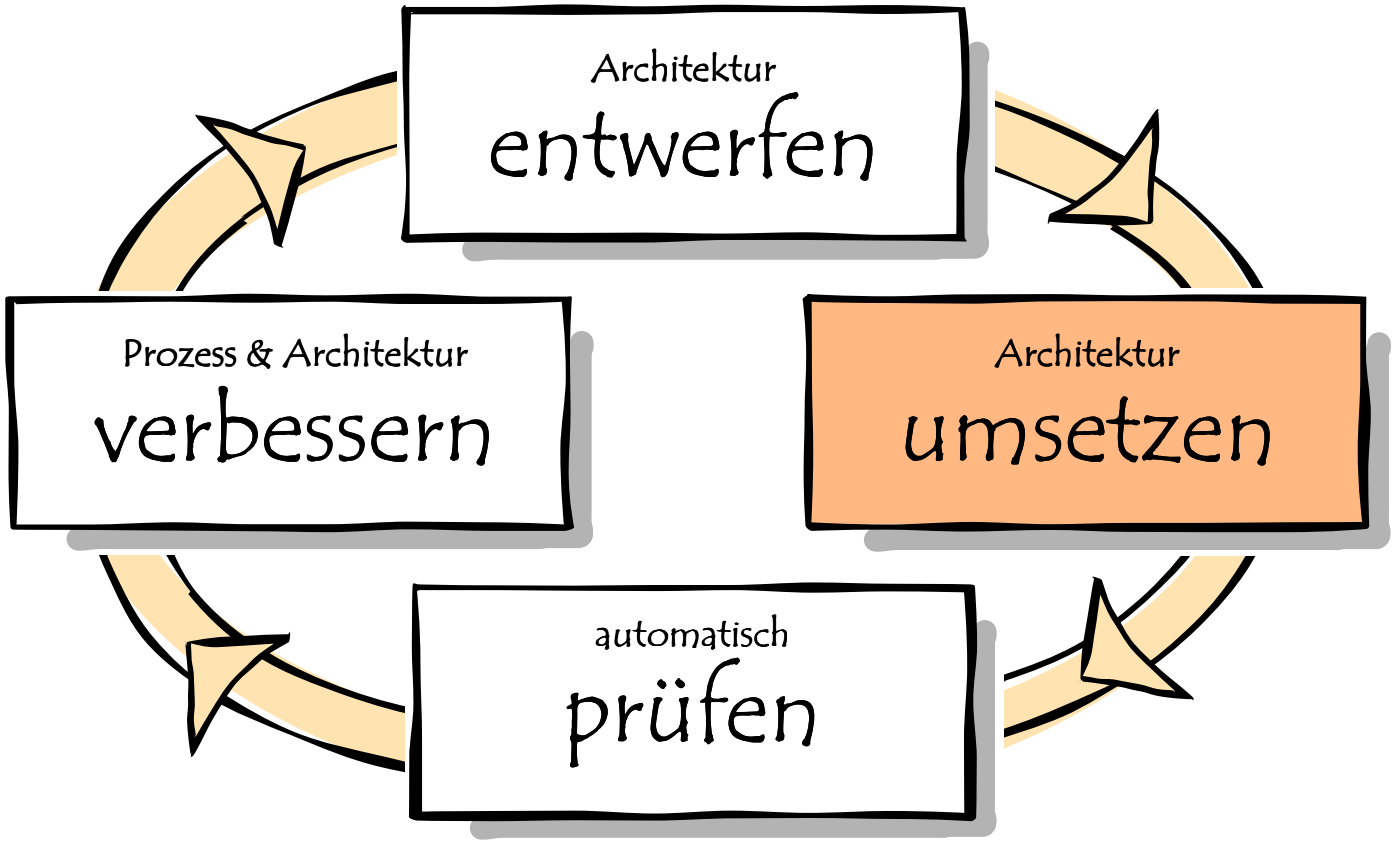
Orientation in Objects

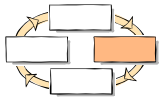
Wenn man die Architektur

nicht erklären kann,

ist sie ***zu kompliziert!***


Conference Participant
Compile conference program (at Home)
Load/Compile conference program (during Conference)
PC / Browser
«Client»
DukeCon HTML5
Mobile Device
«Client»
DukeCon mobile
«Client»
DukeCon Flex
Conf. Data/Favorites
Authenticate
Conf. Data/Favorites
Authenticate
DukeCon Infrastructure
«Docker : Java8»
DukeCon-Server
«Service»
DukeCon
«Docker : WildFly»
KeyCloak-Server
«Service»
KeyCloak
JDBC
JDBC
«Docker : PostgreSQL»
DukeCon-DB
DukeCon_PostgreSQL
«Docker : PostgreSQL»
KeyCloak-DB
KeyCloak_PostgreSQL
Social Media Cloud(s)
«OAuth Provider»
Github
«OAuth Provider»
Google
«OAuth Provider»
Twitter

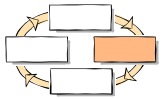

Architektur
entwerfen
Architektur
umsetzen
automatisch
prüfen
Prozess & Architektur
verbessern



Orientation in Objects

7

Dokumentation generieren



Quellcode
DB-Schema
UML-Modell
Schnittstellen
Konfiguration

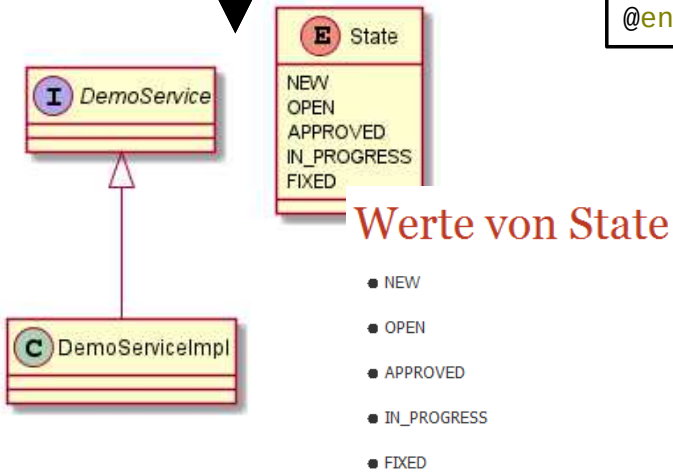
PlantUML +
AsciiDoc
generieren

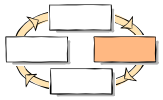
Maven
Gradle

```
@startuml

interface DemoService
DemoService <|-- DemoServiceImpl

enum State {
     NEW
     OPEN
     APPROVED
     IN_PROGRESS
     FIXED
}

@enduml
```

```
# Werte von State

* NEW
* OPEN
* APPROVED
* IN_PROGRESS
* FIXED
```

```
interface DemoService {
    String foobar();
}

class DemoServiceImpl implements DemoService {
    @Override
    public String foobar() {
        return "demo";
    }
}

enum State {
    NEW, OPEN, APPROVED, IN_PROGRESS, FIXED
}
```





Orientation in Objects

Swagger2Markup / swagger2markup

Watch 15 | Star 294 | Fork 54

Code | Issues 7 | Pull requests 0 | Pulse | Graphs

A Swagger to AsciiDoc or Markdown converter to simplify the generation of an up-to-date RESTful API documentation by combining documentation that's been hand-written with auto-generated API documentation.

Spring REST Docs SUCCESS

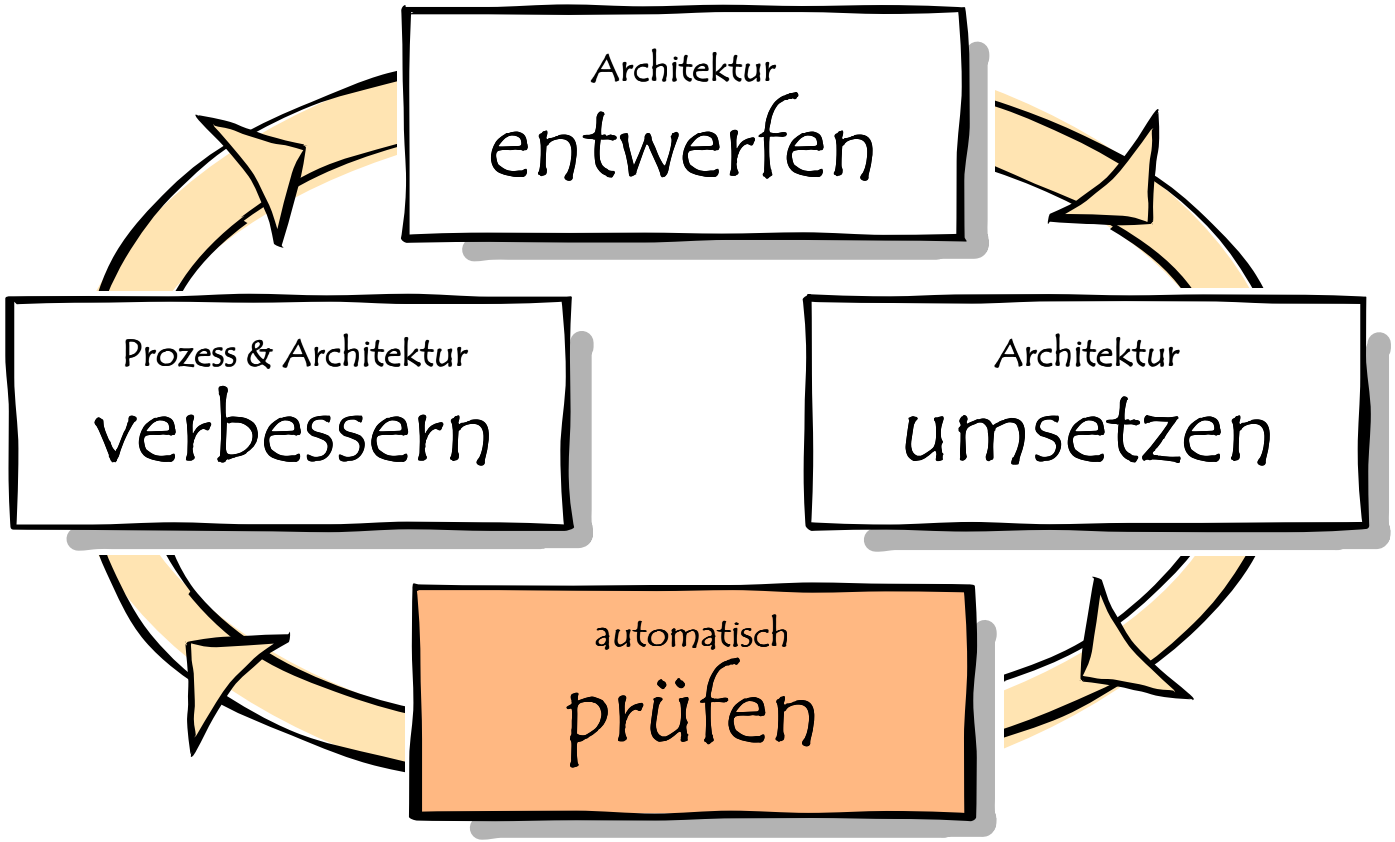
Overview

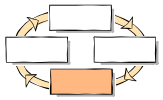
The primary goal of this project is to make it easy to document RESTful services by combining content that's been hand-written using Asciidoctor with auto-generated examples produced with the Spring MVC Test framework. The result is intended to be an easy-to-read user guide, akin to GitHub's API documentation for example, rather than the fully automated, dense API documentation produced by tools like Swagger.

JAX-RS Analyzer

Generates an overview of all JAX-RS resources in a project by bytecode analysis.

Generate automatisiert validieren!


Architektur
entwerfen
Architektur
umsetzen
automatisch
prüfen
Prozess & Architektur
verbessern



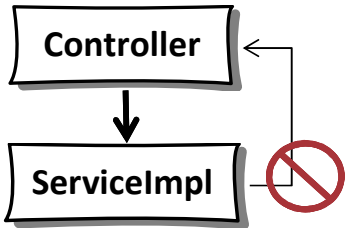
Orientation in Objects

8

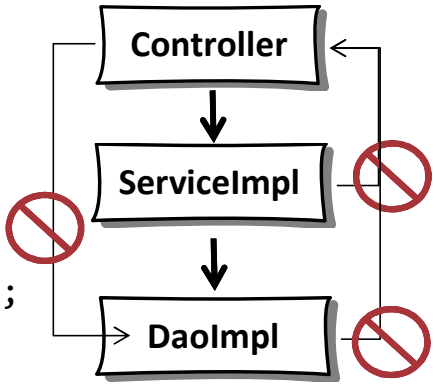
Automatisierte Architekturchecks einrichten

Orientation in Objects

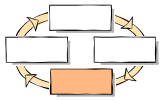
```
@Component
public class DemoServiceImpl implements DemoService {

  @Autowired
  private DemoController demoController;

  public void superServiceMethod() {
    demoController.doSomething();
  }
}
```



```java
@RunWith(SpringRunner.class)
@SpringBootTest
public class DepencencyTesterApplicationTests {

  @Autowired
  private DefaultListableBeanFactory beanFactory;

  @Test
  public void checkDependencies() {
    for (String beanName : beanFactory.getBeanDefinitionNames()) {
      for (String depName : beanFactory.getDependenciesForBean(beanName)) {
        checkNotAllowedConnection(beanName, depName, ".*ServiceImpl", ".*Controller");
        checkNotAllowedConnection(beanName, depName, ".*Controller", ".*DaoImpl");
        checkNotAllowedConnection(beanName, depName, ".*DaoImpl", ".*Controller");
      }
    }
  }

  private void checkNotAllowedConnection(String beanName, String depName, String beanNamePattern,
                  String dependencyNamePattern) {
    String msg = "Connection not allowed: " + beanName + " -> " + depName;
    assertFalse(msg, beanName.matches(beanNamePattern) && depName.matches(dependencyNamePattern));
  }
}
```

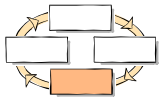


https://github.com/thorstenmaier/architecture-layer-check/



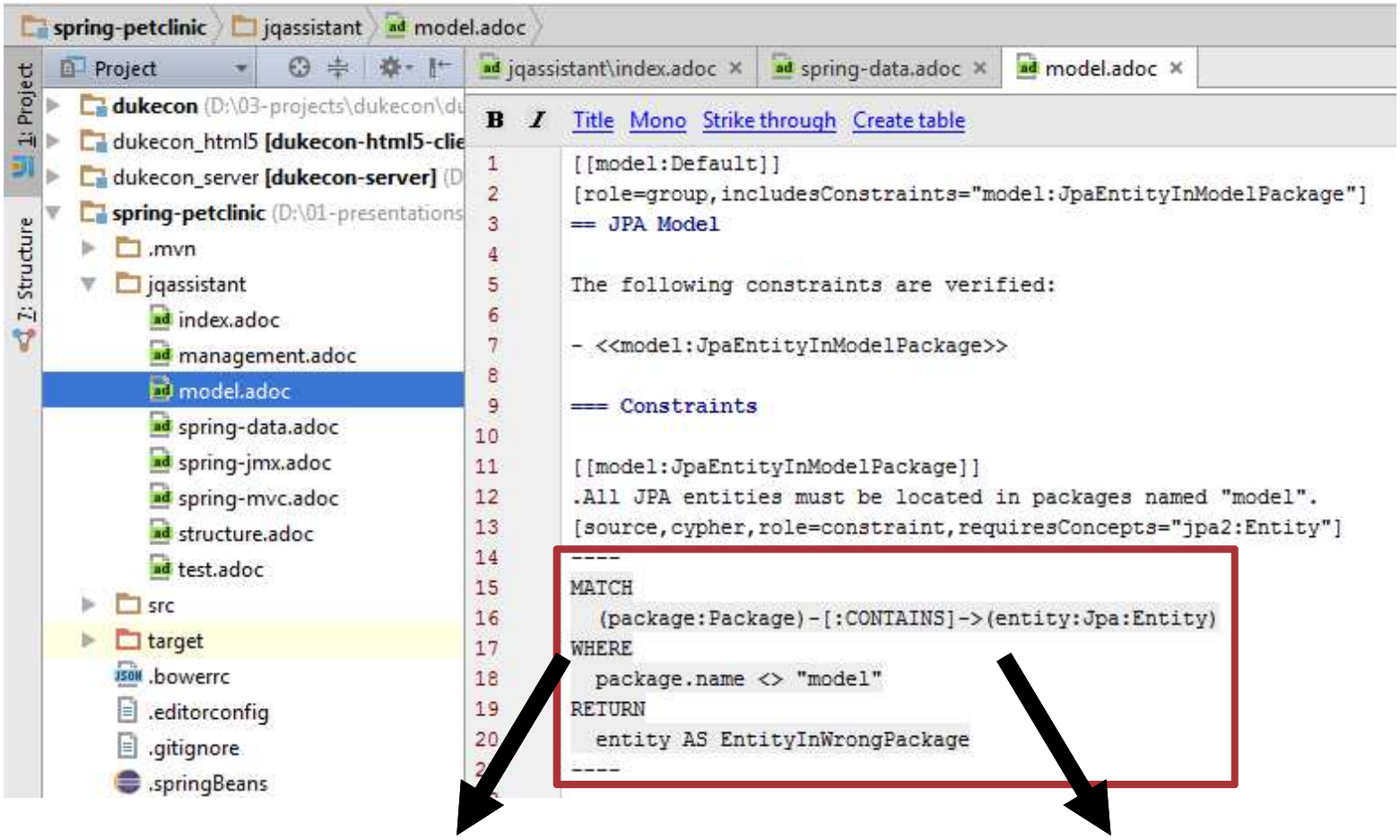
Orientation in Objects

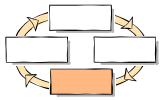
Dokumentation ausführen

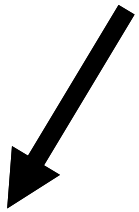


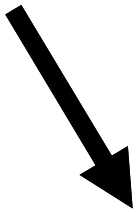
jQAssistant Living Documentation

Orientation in Objects









Orientation in Objects

JPA Model

The following constraints are verified:

- All JPA entities must be located in packages named "model".

Constraints

All JPA entities must be located in packages named "model".

```
MATCH
  (package:Package)-[:CONTAINS]->(entity:Jpa:Entity)
WHERE
  package.name <> "model"
RETURN
  entity AS EntityInWrongPackage
```

jQAssistant Report

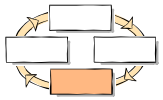
Groups

#	Group Name
1	default
2	management:Default
3	test:Default
4	structure:Default
5	model:Default

Constraints

- Move the mouse over a constraint to view a description.
- Click on a failed constraint to open a details view.

#	Constraint Name	Count
1	assertj:TestMethodWithoutAssertion ✓	
2	junit4:IgnoreWithoutMessage ✓	
3	structure:ImplementationDependencies ✓	
4	dependency:PackageCycles ✓	
5	structure:ControllerDependencies ✓	
6	structure:RepositoryDependencies ✓	
7	structure:ServiceDependencies ✓	
8	model:JpaEntityInModelPackage ✓	



jQAssistant

Sneaky Preview

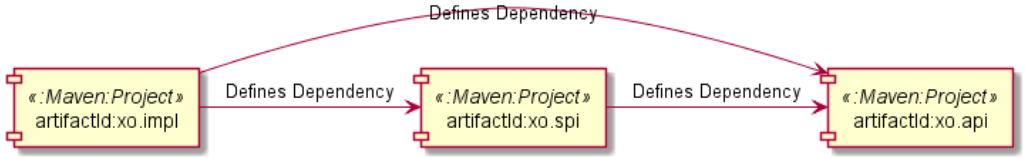
Architektur-definition in PlantUML

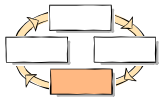
Generierte Cypher-Regel

Orientation in Objects

```
[[architecture:DefinedDependencies]
[plantuml,role=concept]
----
[artifactId:xo.impl] as impl <<:Maven:Project>>
[artifactId:xo.api] as api <<:Maven:Project>>
[artifactId:xo.spi] as spi <<:Maven:Project>>

impl -> api : Defines Dependency
impl -> spi : Defines Dependency
spi -> api : Defines Dependency
----
```

```
[[architecture:UndefinedDepende
[source,cypher,role=constraint,rec
There must not be dependencies
----
MATCH
  (p1:Maven:Project)-[:CREATES]
  (p2:Maven:Project)-[:CREATES]
  (t2)-[:DEPENDS_ON]->(t1)
WHERE NOT
  (p1)-[:DEFINES_DEPENDENCY
RETURN
  *
----
```

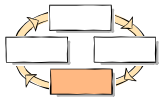




Orientation in Objects

10

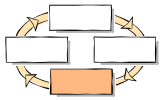
Code Reviews installieren



Orientation in Objects

Kennen Sie das?

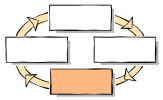
„Das ist historisch gewachsen.“



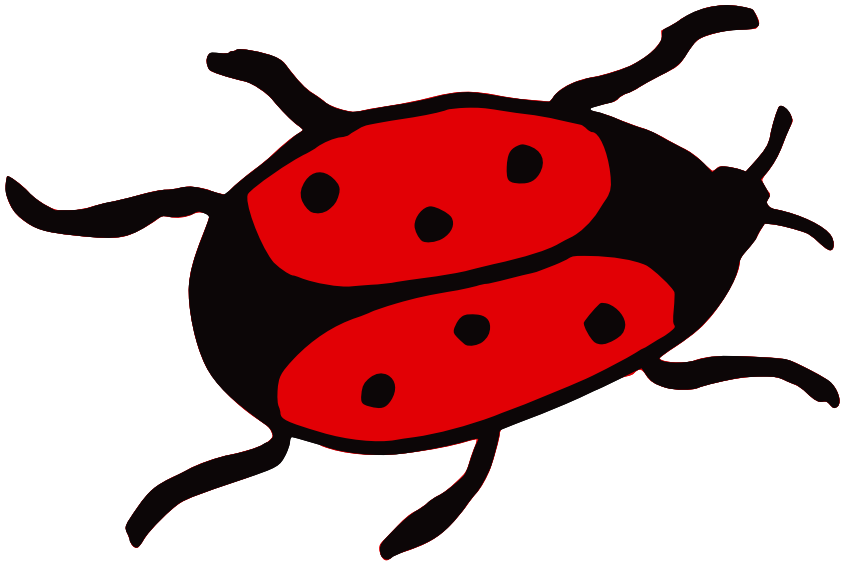
Orientation in Objects

…noch ein Klassiker

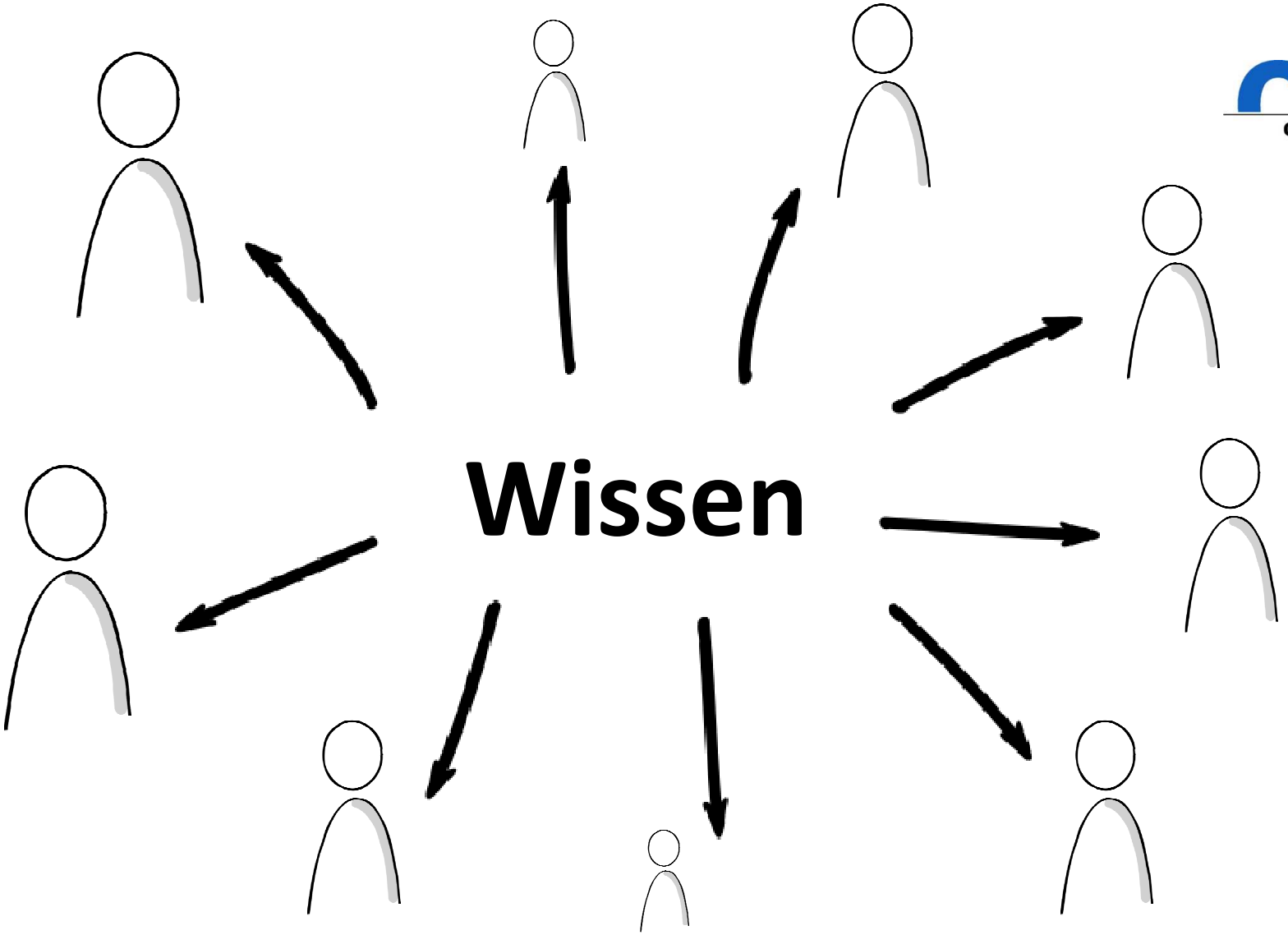
„Thomas ist leider gerade im Urlaub“

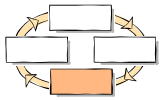


Orientation in Objects

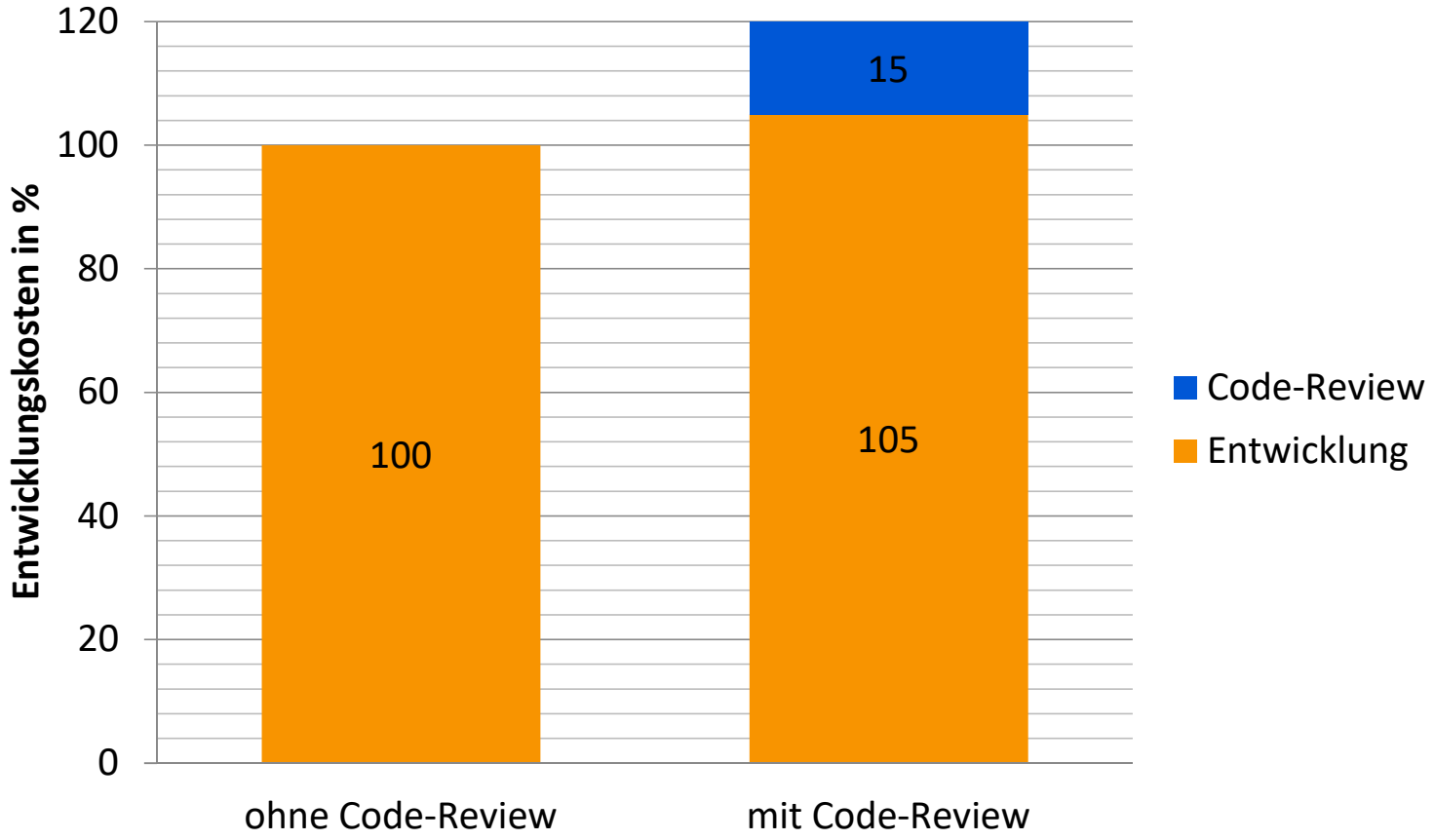


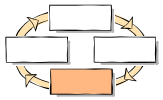
Bugs finden


Wissen

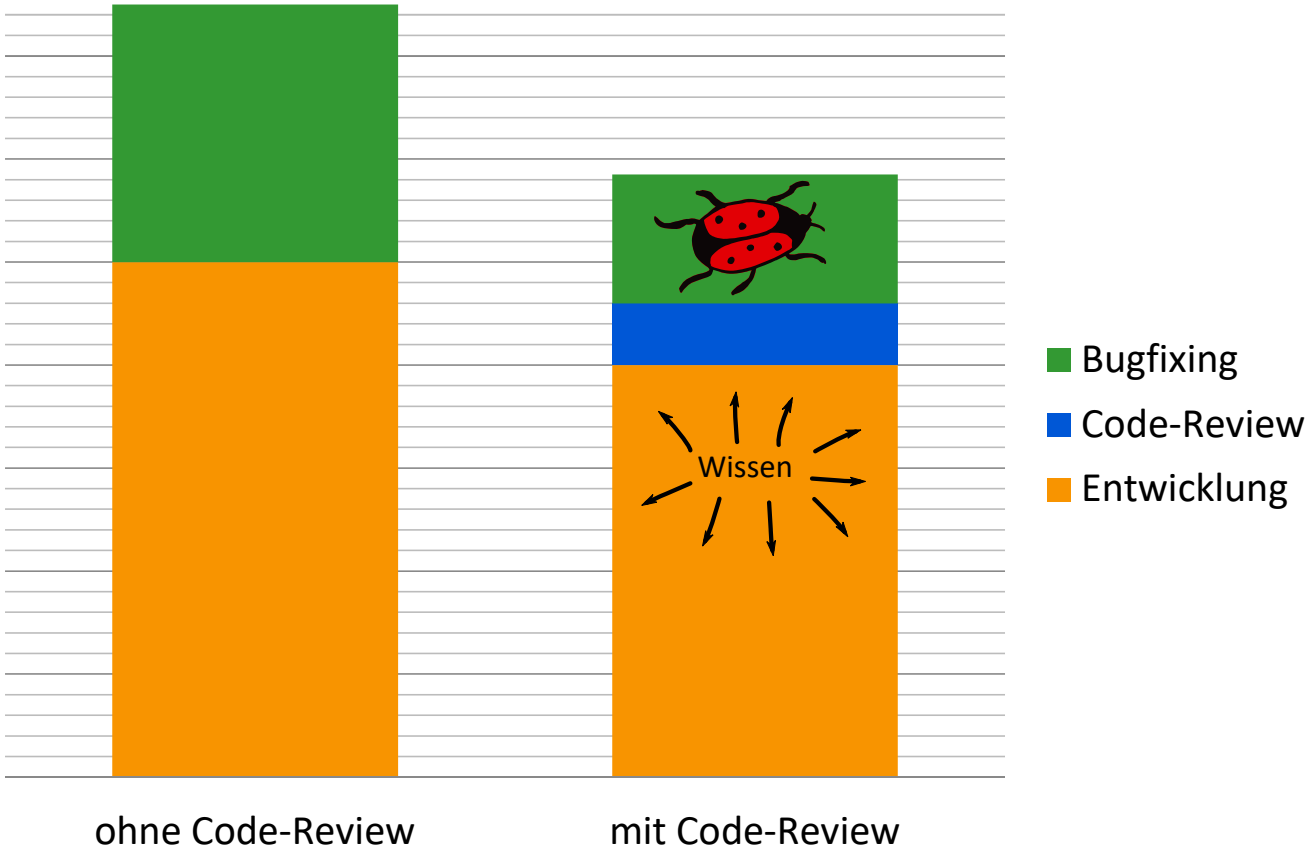


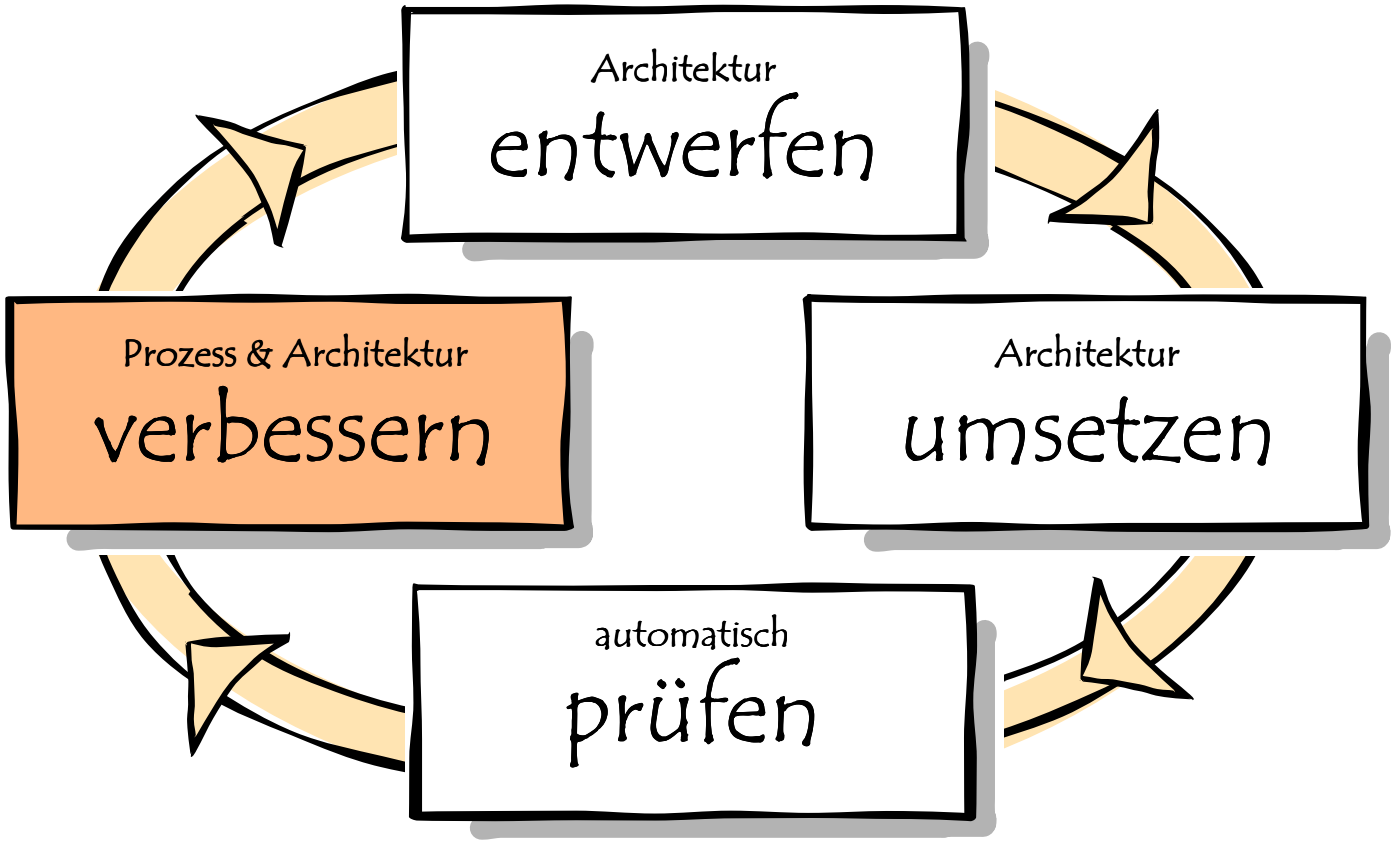
Orientation in Objects

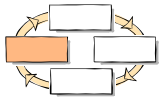




Orientation in Objects



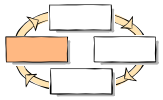

Architektur
entwerfen
Architektur
umsetzen
automatisch
prüfen
Prozess & Architektur
verbessern



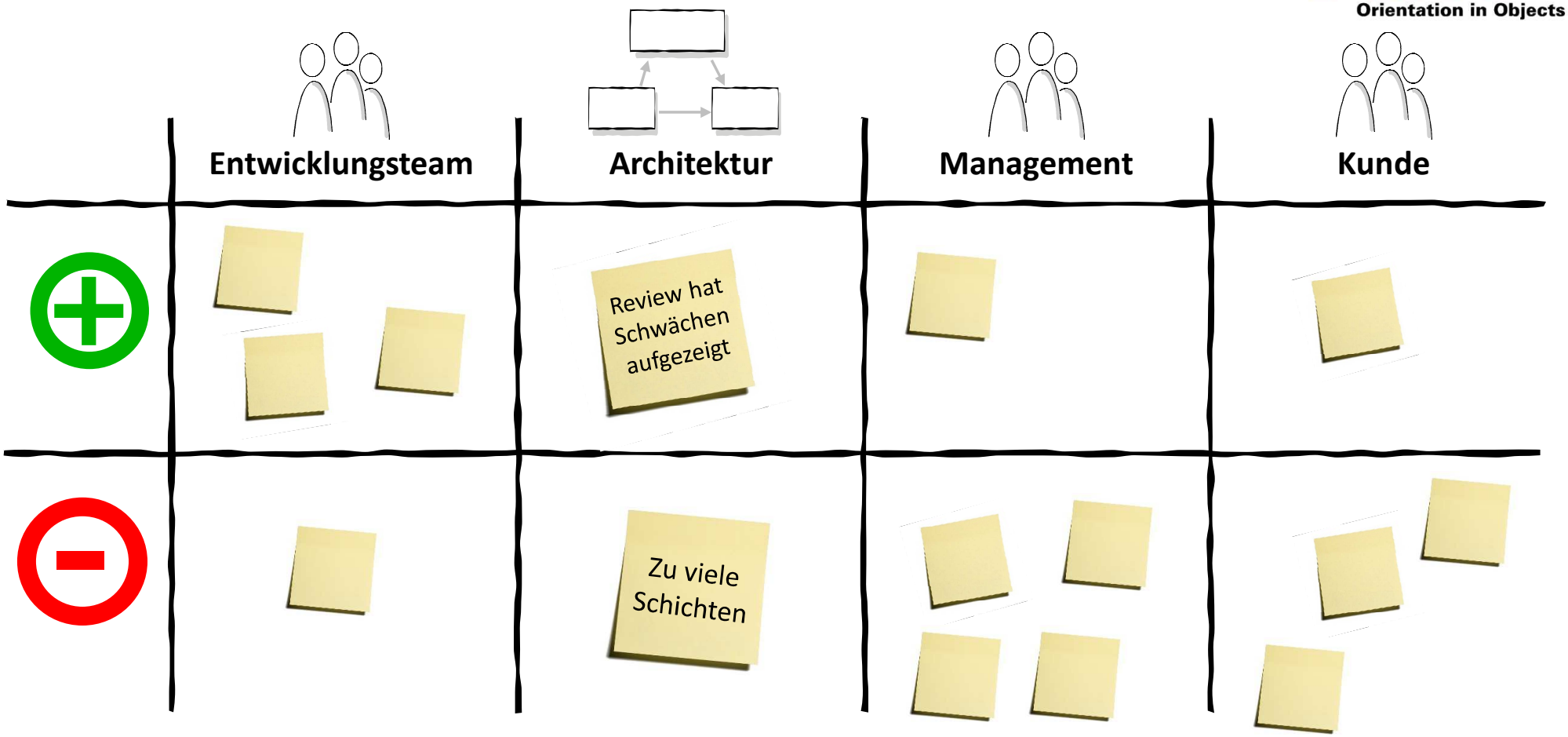
Orientation in Objects

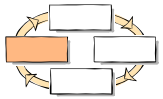
11

Retrospektive durchführen



Orientation in Objects

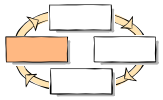




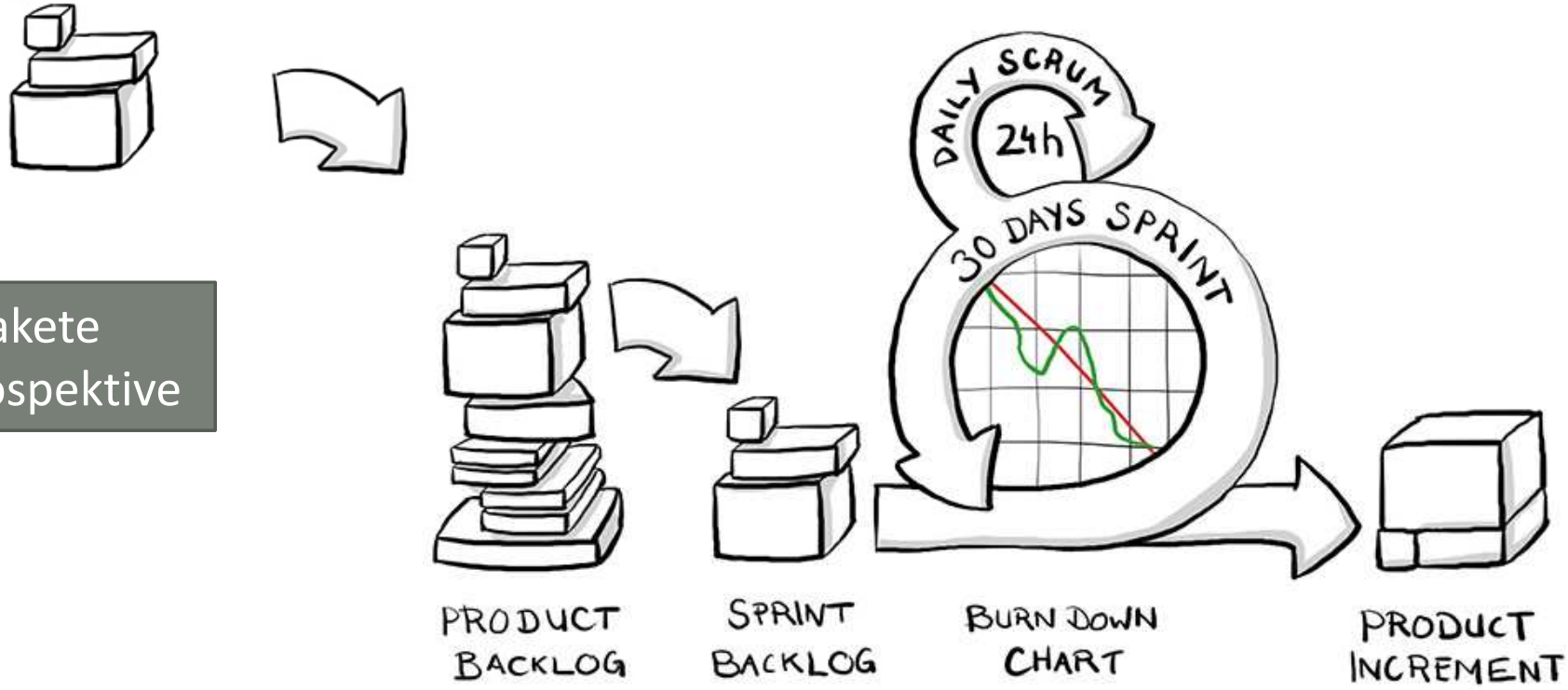
Orientation in Objects

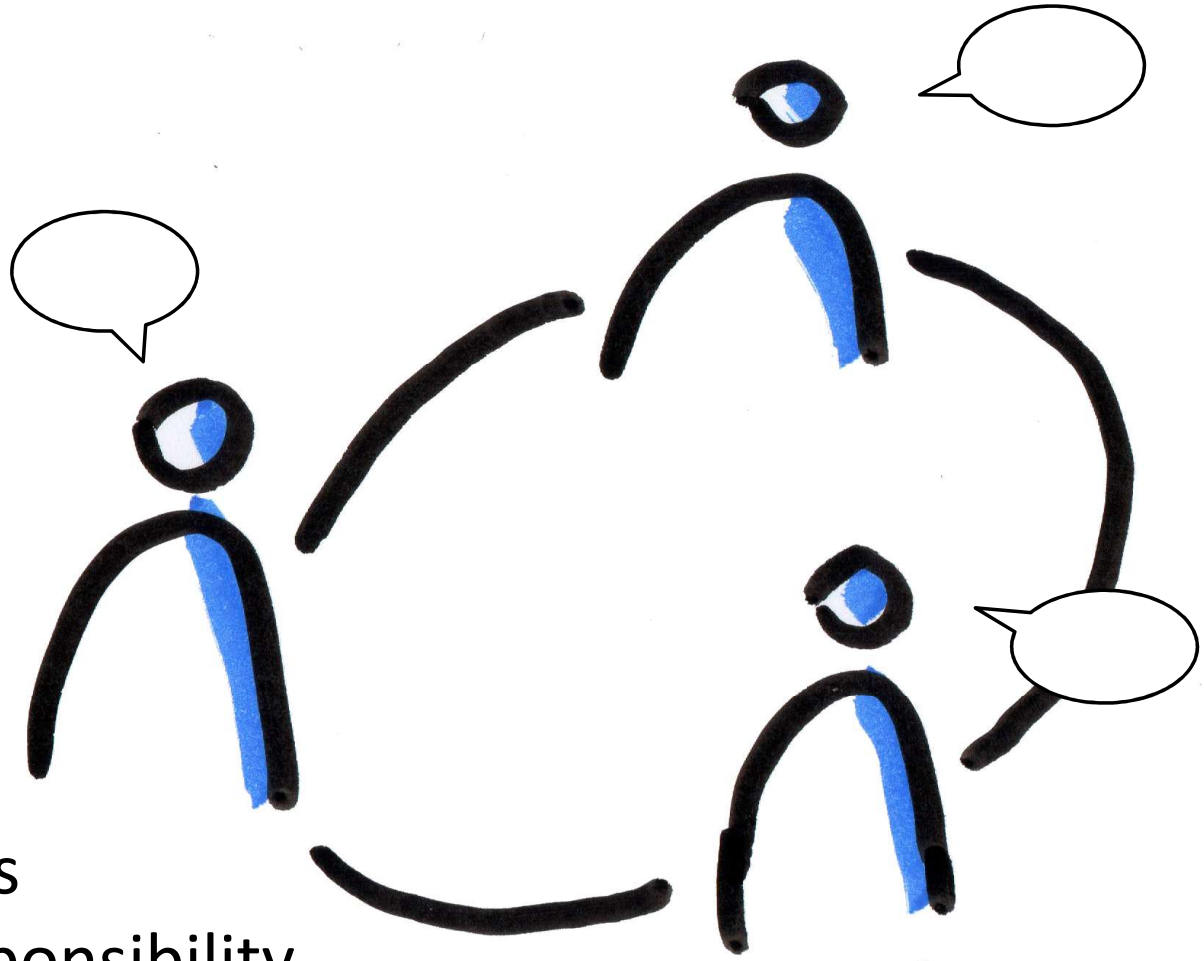
12

Verbesserungen einplanen und vorantreiben

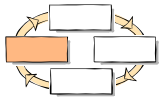


Orientation in Objects





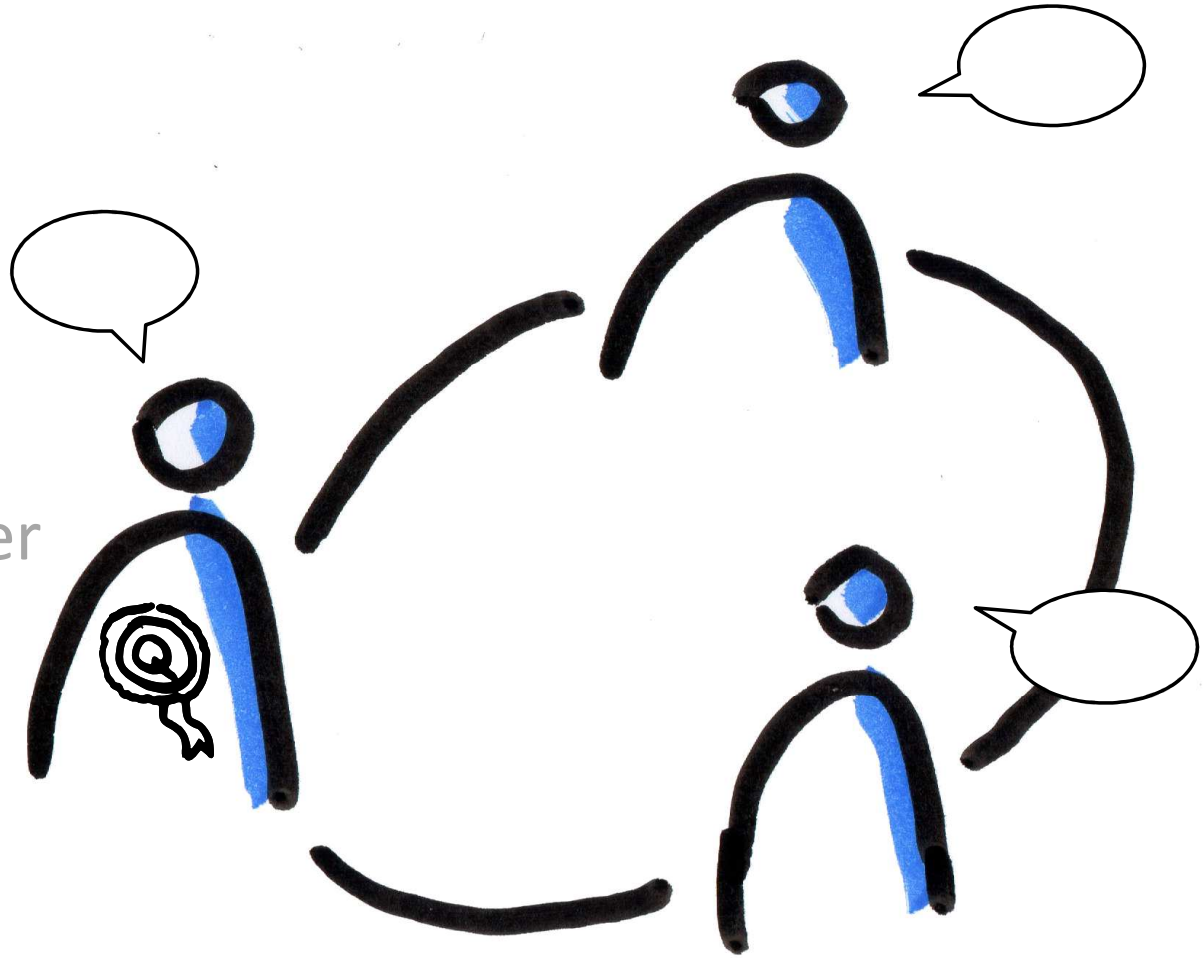
Something that's
everybody's responsibility
is no one's responsibility.

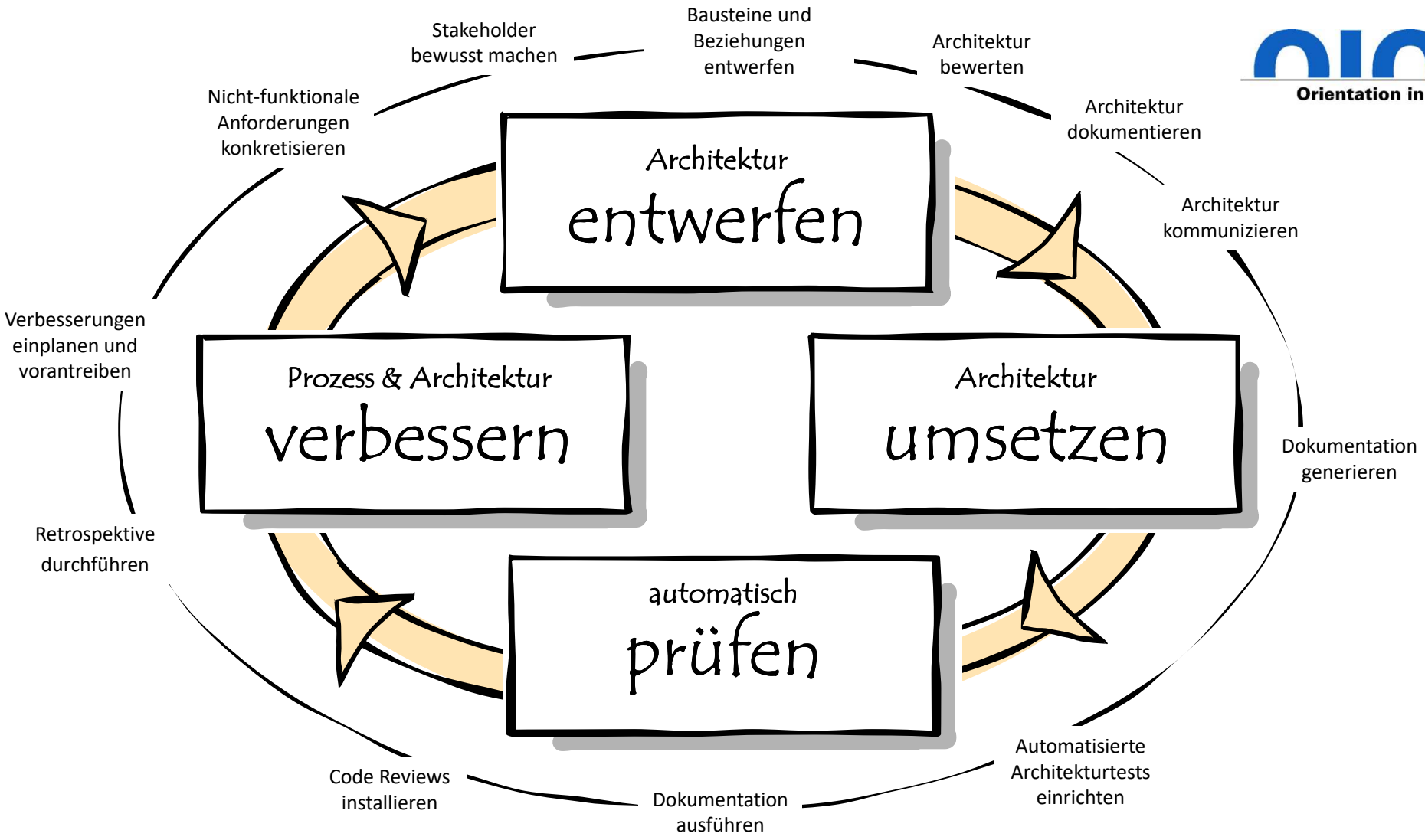


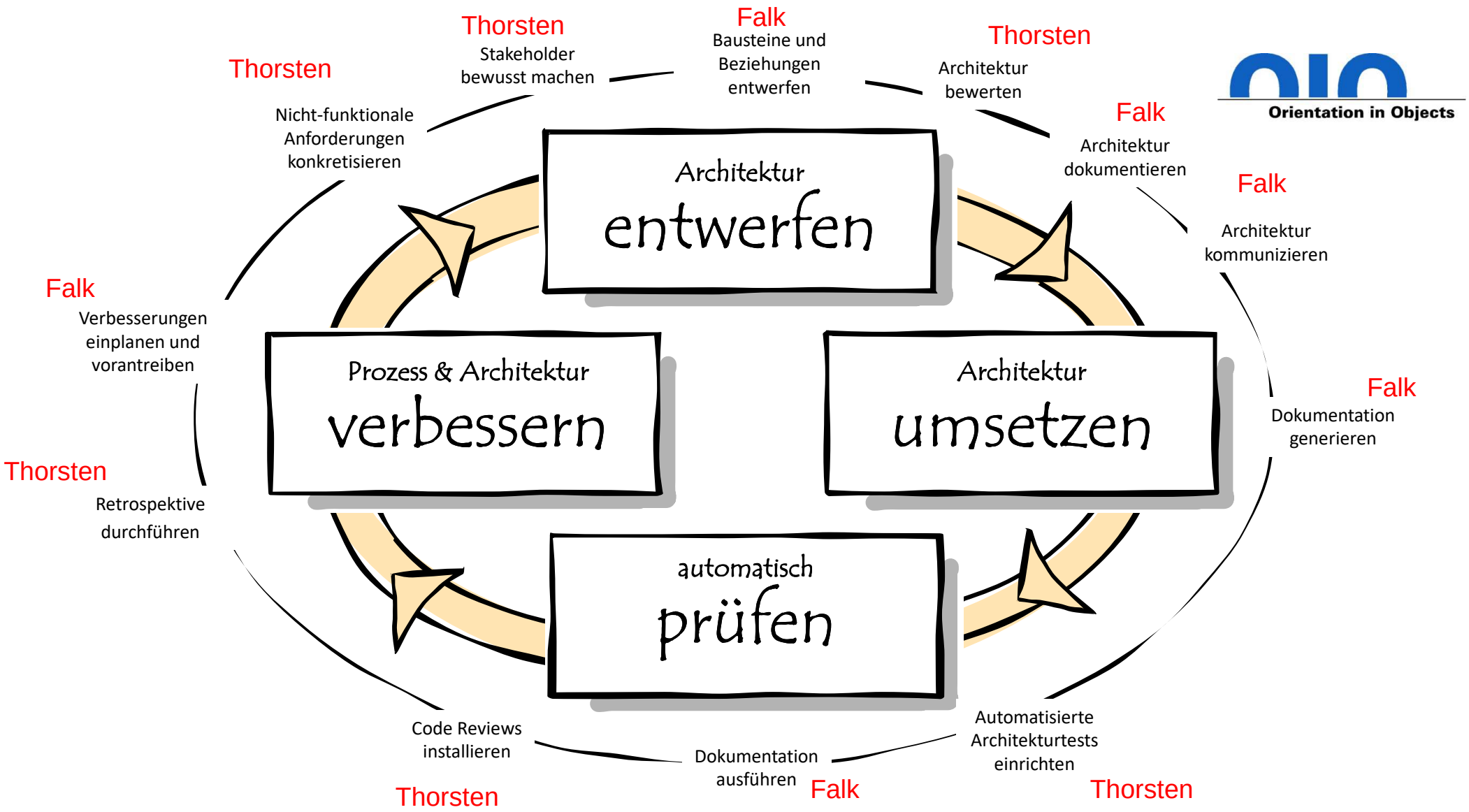
Orientation in Objects

Mitentwickelnder

Kümmerer/
Koordinator




Stakeholder bewusst machen
Bausteine und Beziehungen entwerfen
Architektur bewerten
Architektur dokumentieren
Architektur kommunizieren
Dokumentation generieren
Automatisierte Architekturtests einrichten
Dokumentation ausführen
Code Reviews installieren
Retrospektive durchführen
Verbesserungen einplanen und vorantreiben
Nicht-funktionale Anforderungen konkretisieren
Architektur
entwerfen
Architektur
umsetzen
automatisch
prüfen
Prozess & Architektur
verbessern


Thorsten
Stakeholder bewusst machen
Falk
Bausteine und Beziehungen entwerfen
Thorsten
Architektur bewerten
Falk
Architektur dokumentieren
Falk
Architektur kommunizieren
Falk
Dokumentation generieren
Thorsten
Automatisierte Architekturtests einrichten
Falk
Dokumentation ausführen
Thorsten
Code Reviews installieren
Thorsten
Retrospektive durchführen
Falk
Verbesserungen einplanen und vorantreiben
Thorsten
Nicht-funktionale Anforderungen konkretisieren
Architektur
entwerfen
Architektur
umsetzen
automatisch
prüfen
Prozess & Architektur
verbessern
Orientation in Objects

Orientation in Objects

Vielen Dank für Ihre Aufmerksamkeit!

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de