

Agile Architektur



Wie viel Stabilität verträgt Agilität?

Gerhard Müller [@gmtng](https://twitter.com/gmtng)

München, 19. Juli 2017

Kontext TNG Technology Consulting

- IT-Consulting-Firma in / um München
- Gründung 1.1.2001, aktuell etwa 284 Kollegen
- Viel Individual-Software-Entwicklung, möglichst nach agilen Methoden (Scrum, Kanban) und mit modernen Tools
- Besondere Spezialität: unterspezifizierte Aufgaben, Herausforderungen
- Liebe zur Technik, kein Branchenfokus
- Technologie-Schwerpunkte: Java, JavaScript, Python, C#, Scala, ...



Agenda



1. Software-Architektur und Agilität

2. Typische Herausforderungen
3. Software-Architektur “richtig”
4. Ansätze bei TNG
5. Zusammenfassung

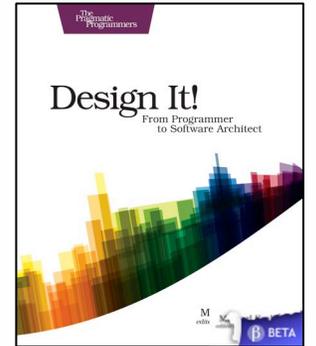
1



Software-Architektur und Agilität

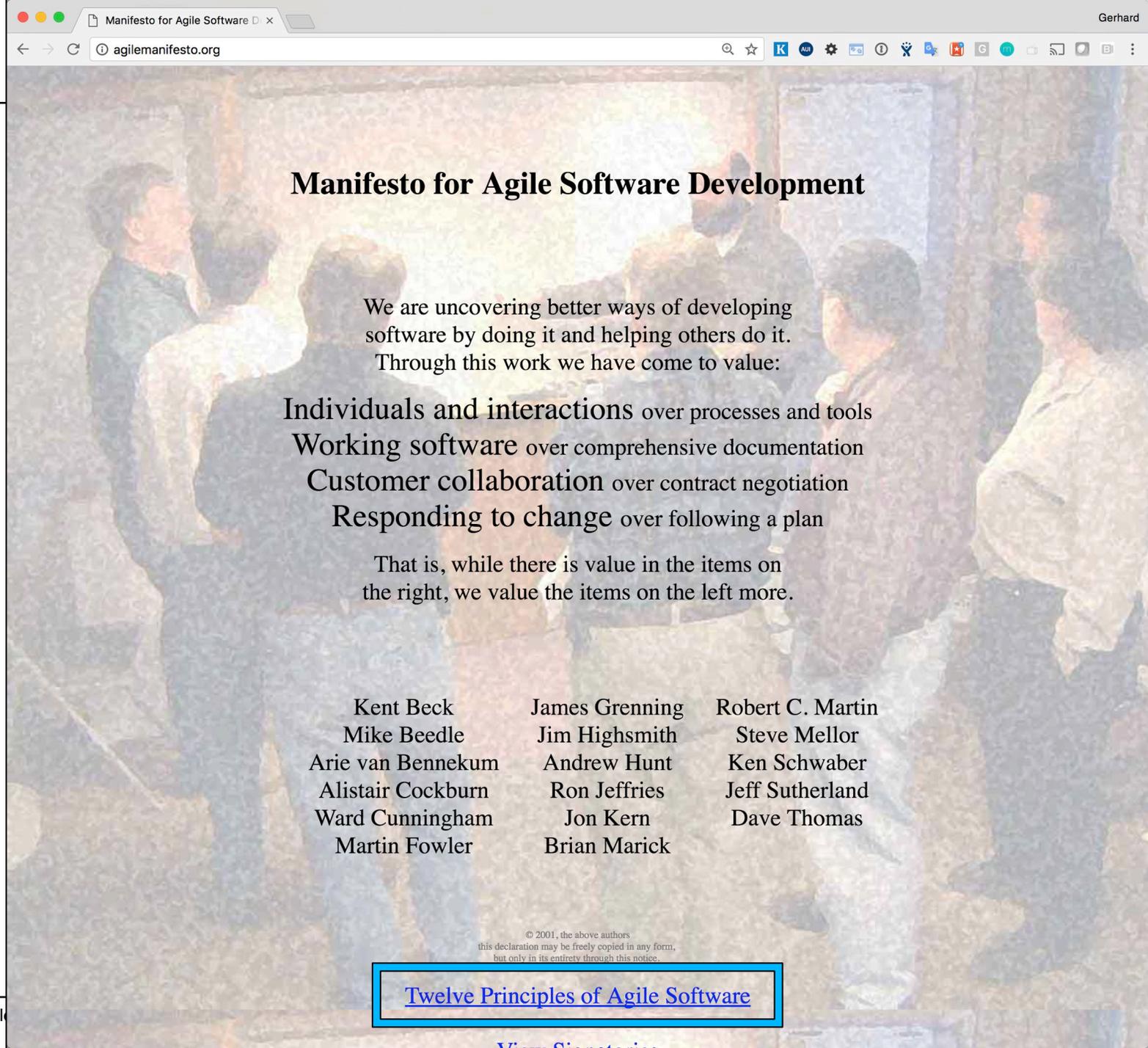
Was will man mit (guter) Software-Architektur erreichen?

- Aus „Design It!: From Programmer to Software Architect“:
„Seven ways software architecture helps us build amazing software:
 1. ... turns a big problem into smaller more manageable problems.
 2. ... shows people how to work together effectively.
 3. ... provides a vocabulary for talking about complex ideas.
 4. ... looks beyond features and functionality.
 5. ... connects business goals with the practical realities of implementation.
 6. ... helps us avoid costly mistakes.
 7. ... enables agility.“



Agil und Architektur: passt das zusammen?

- Das agile Manifest und die 12 Prinzipien



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

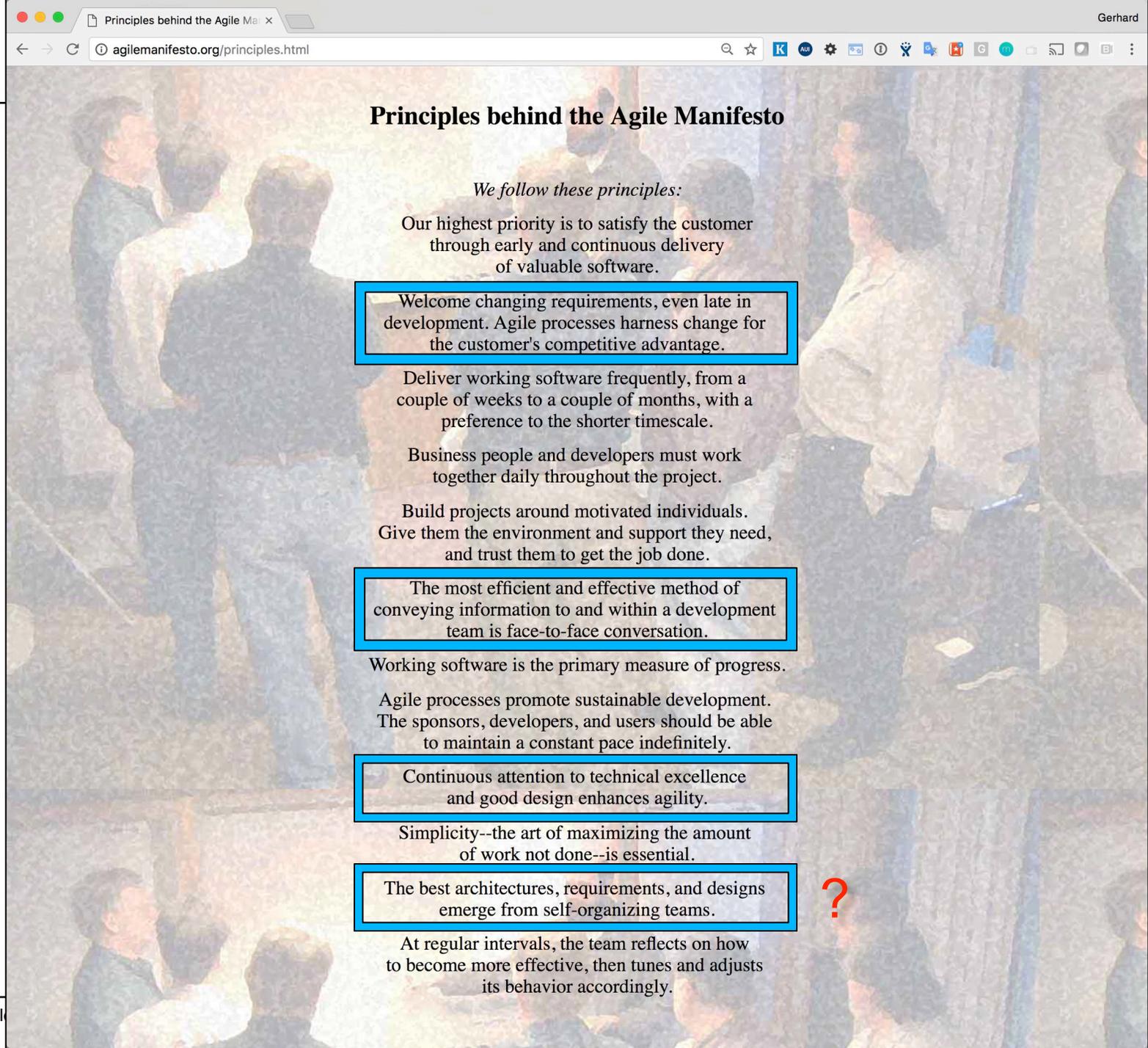
Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

© 2001, the above authors
this declaration may be freely copied in any form,
but only in its entirety through this notice.

[Twelve Principles of Agile Software](#)

Natürlich!

Agil und Architektur: passt das zusammen?



Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

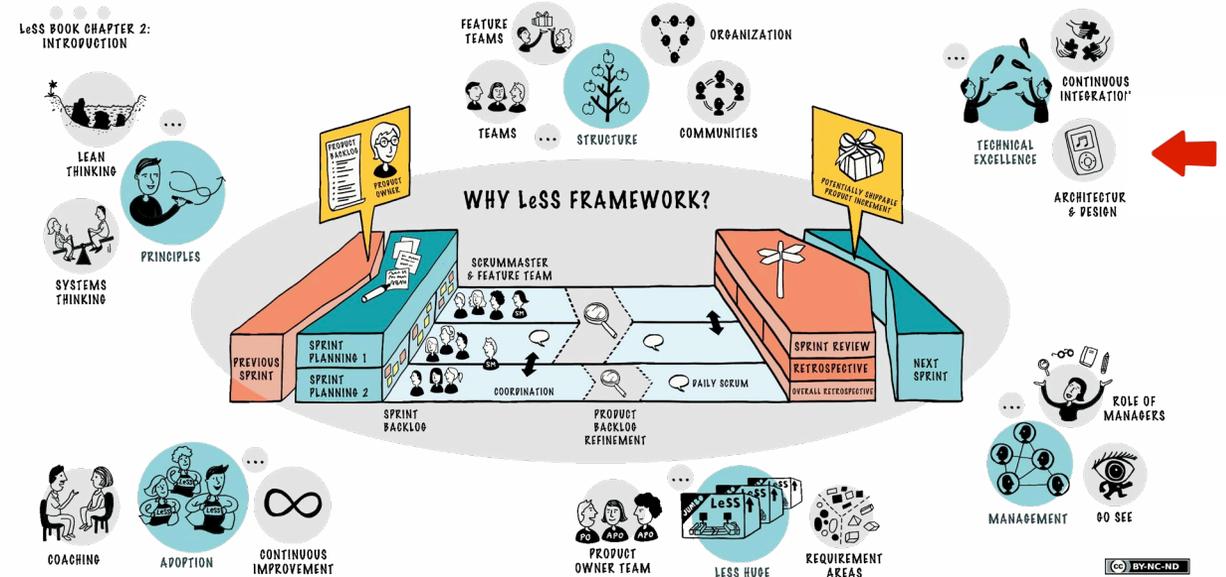
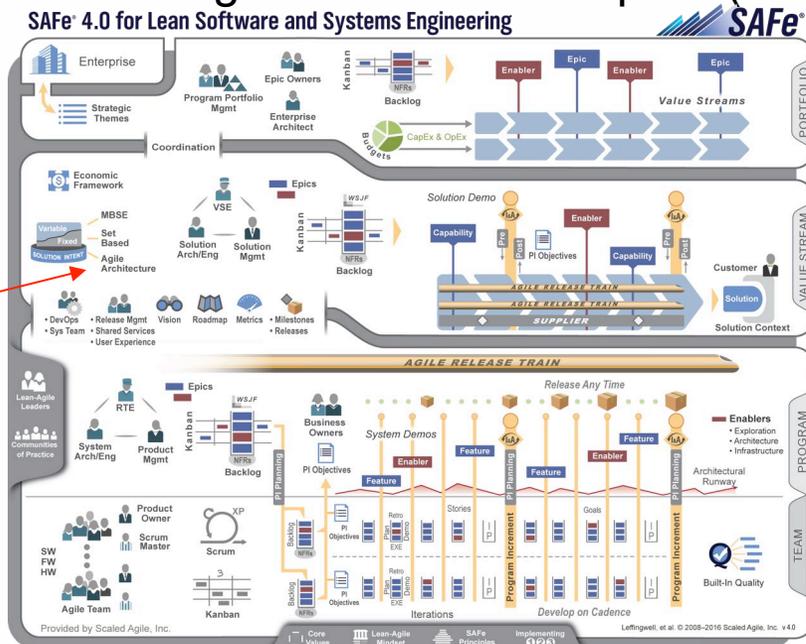
The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.



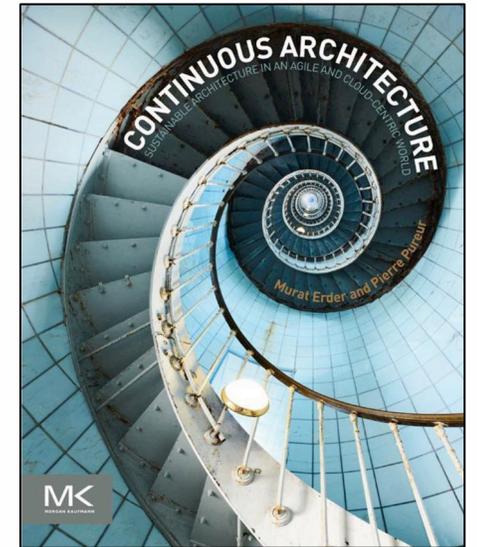
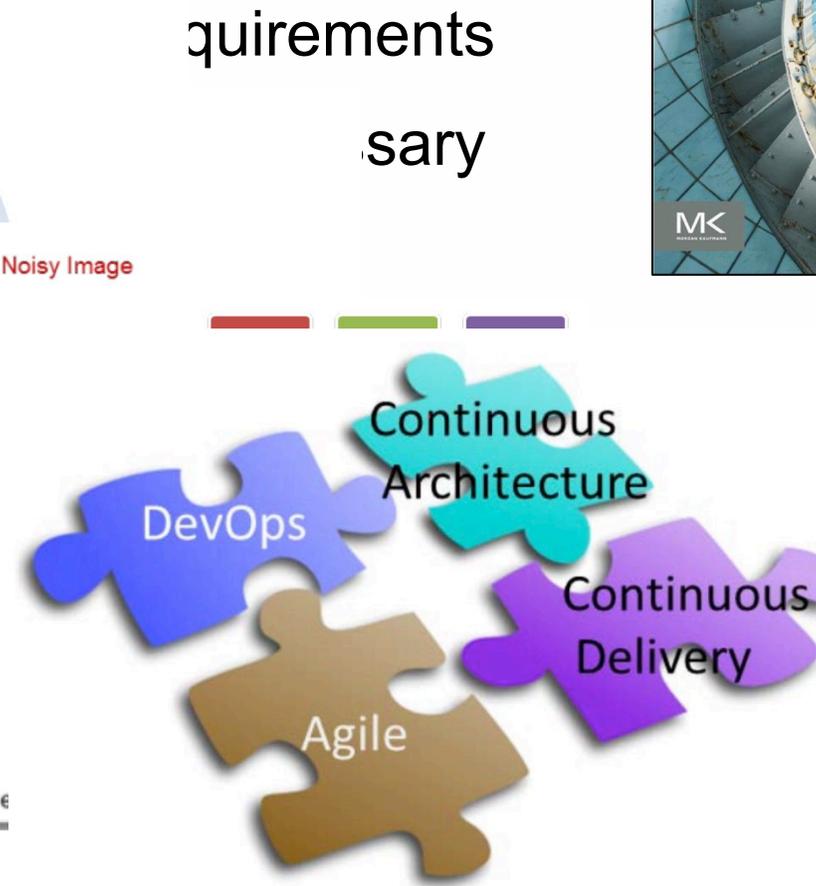
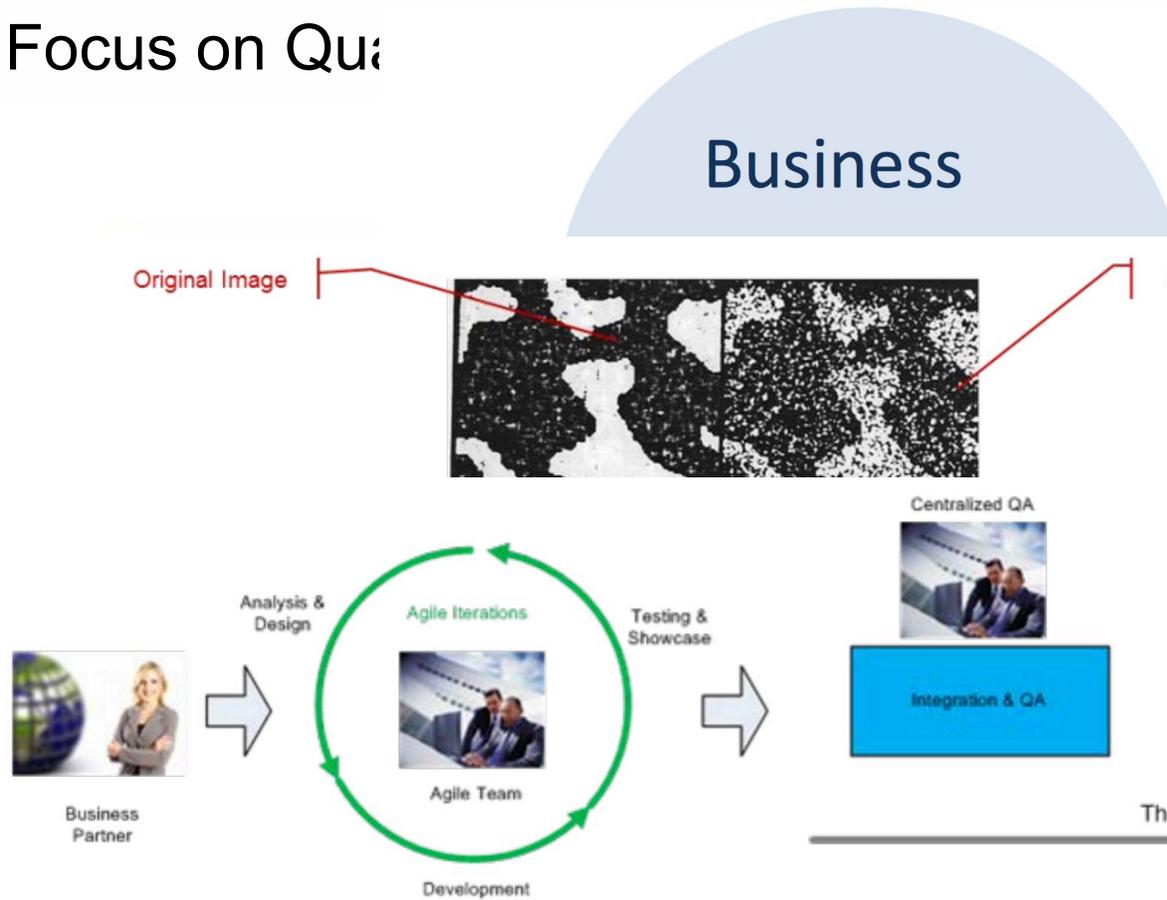
Wo findet man in agilen Prozessen Architektur-Arbeit?

- Definition of Done – Qualitätsattribute = „Rahmenbedingungen“
- In kleinen Teams implizit, z.B.: Sprint 0 (oder sogar davor), Sprint Planning 2, Spikes, Community of Practice, immer!
 - Patterns: <http://www.hillside.net/plop/2015/papers/riverhounds/17.pdf>: Architecture in the Backlog, Architectural Trigger, Architectural Spike, Technical Debt Management
- In skalierten agilen Prozessen explizit (SAFe, LeSS, etc.):



“6 Principles of Continuous Architecture”

1. Architect products, not just solutions for projects
2. Focus on Quality
- 3.
- 4.
- 5.
- 6.

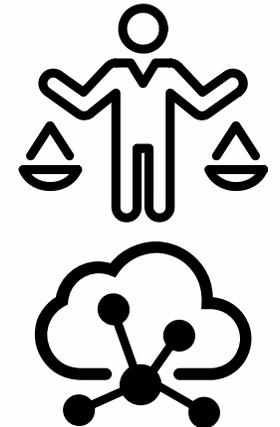
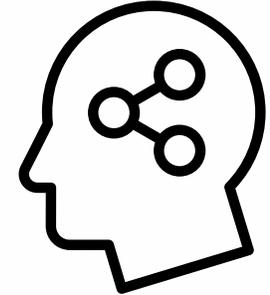


2

Typische Herausforderungen aus der TNG-Praxis

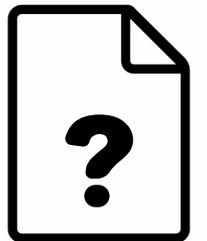
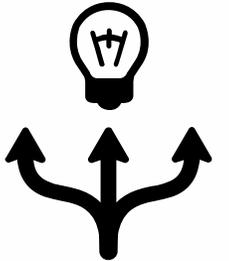
Typische Herausforderungen aus der TNG-Praxis (1/2)

- A) Die Architektur muss in die Köpfe aller relevanten Personen!
 - Sonst wird gegen die hoffentlich durchdachten Ideen verstoßen
 - Architektur-Themen müssen in Projekten die entsprechende Stellung haben
 - Bei mehreren Teams oft wirklich Architekten notwendig
 - Automatisiertes Testen, auch der Architektur, ist notwendig
 - Produktionscode-Architektur & Test-Architektur: beide betrachten!
 - Dokumentation muss geeignet sein
- B) Entscheidungen gut treffen & dokumentieren
 - Symptom: Die gleichen Themen werden wieder und wieder diskutiert
- C) Neue, offenere Architekturen:
API-Management, Cloud, Ökosysteme



Typische Herausforderungen aus der TNG-Praxis (2/2)

- D) Größere Refactorings nicht konsequent durchgezogen
 - Statt einer einheitlichen guten Art, ein Thema zu lösen, gibt es dann die schlechte alte Art und die gute neue Art, die beide parallel zu pflegen sind (zusätzliche Komplexität!)
- E) Qualität von Anforderungen nicht hoch genug
 - Oft nicht so rigoros erarbeitet wie agile Entwicklung mit testgetriebener Entwicklung, Code-Reviews, Continuous Integration & Delivery
 - Oft nur Hypothesen, die halt oft auch nicht stimmen (→ Lean Startup-Methoden)
 - Fachlichkeit bzw. Kontext der Fachlichkeit (Verbindung zum Code) geht verloren
- F) Gemeinsames, funktionierendes, effizientes Tooling
 - Wiki, Issue Management, Version Control, CI/CD, ...



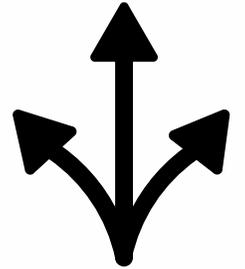
3



Software-Architektur „richtig“

Warum sind gute Architekturen gerade jetzt so wichtig?

- Gute Architektur: erzeugt Optionen zur Veränderung
- Digitalisierung & Technik-Entwicklung & neue Geschäftsmodelle & neue Architekturen (Outside-In) → volatiler Markt
 - Siehe auch https://youtu.be/qiMzkgCI_tU (Prof. Matthes / <http://swa-muc.de> am 21.3.2017)
 - Siehe auch <https://www.infoq.com/articles/real-options-enhance-agility>
- In einem volatilen Markt haben Optionen einen höheren Wert als Stabilität
 - → Optionen haben ist wertvoll!
- Entscheidungen treffen: Last Responsible Moment
 - Das kann auch schon vor Projektbeginn sein...
 - Das Ziel: relevante Entscheidungen möglichst billig wieder ändern zu können



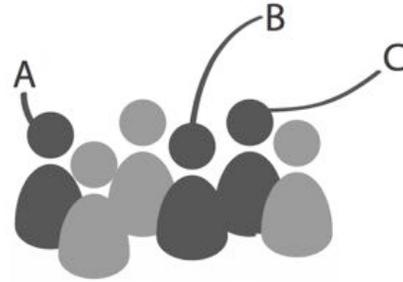
Die Rolle des Architekten

Kein benannter Architekt



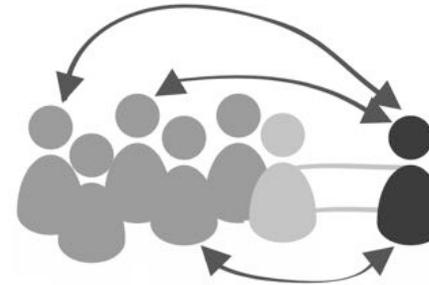
Entwickler stimmen sich selbstständig zu Architekturfragen ab. Jeder Entwickler trifft (und kommuniziert) potenziell auch Architekturentscheidungen.

Architekturagenten



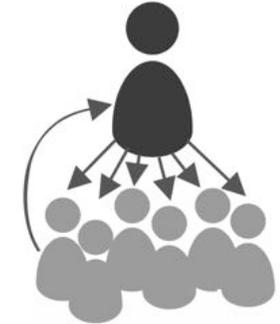
Einige Architekturthemen sind **explizit** von Entwicklern mit Spezialwissen besetzt. Sie kümmern sich um das Thema und unterstützen andere bei Entscheidungen.

unterstützender Architekt

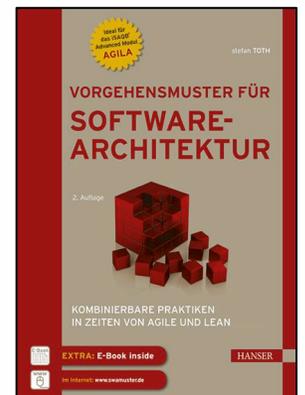


Ein oder mehrere Entwickler übernehmen die Architektenrolle als Teilzeitjob. Fokus liegt auf Unterstützung und Mentoring, nicht auf dem Treffen von Entscheidungen.

klassischer Architekt



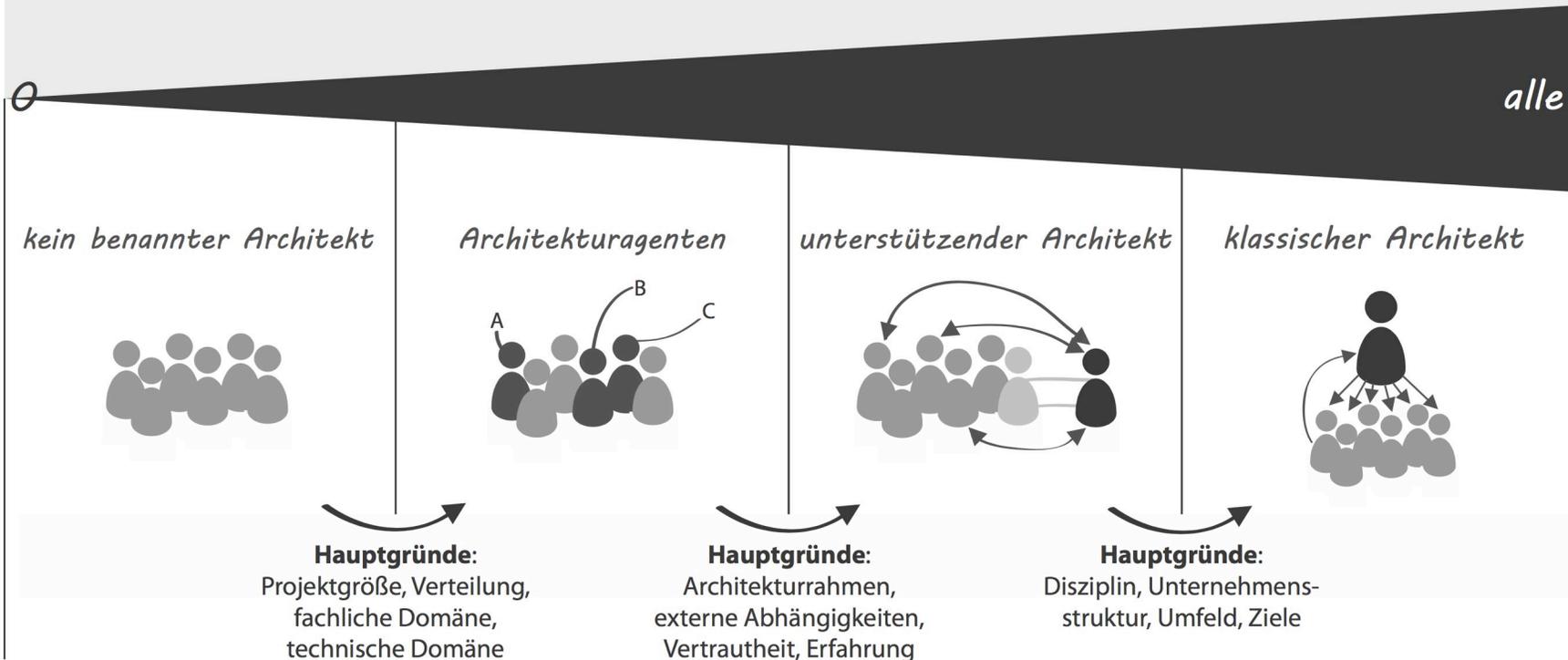
Ein oder mehrere Architekten kümmern sich um Architekturfragen und treffen alle wichtigen Entscheidungen. Entwickler folgen den Plänen und geben Feedback.



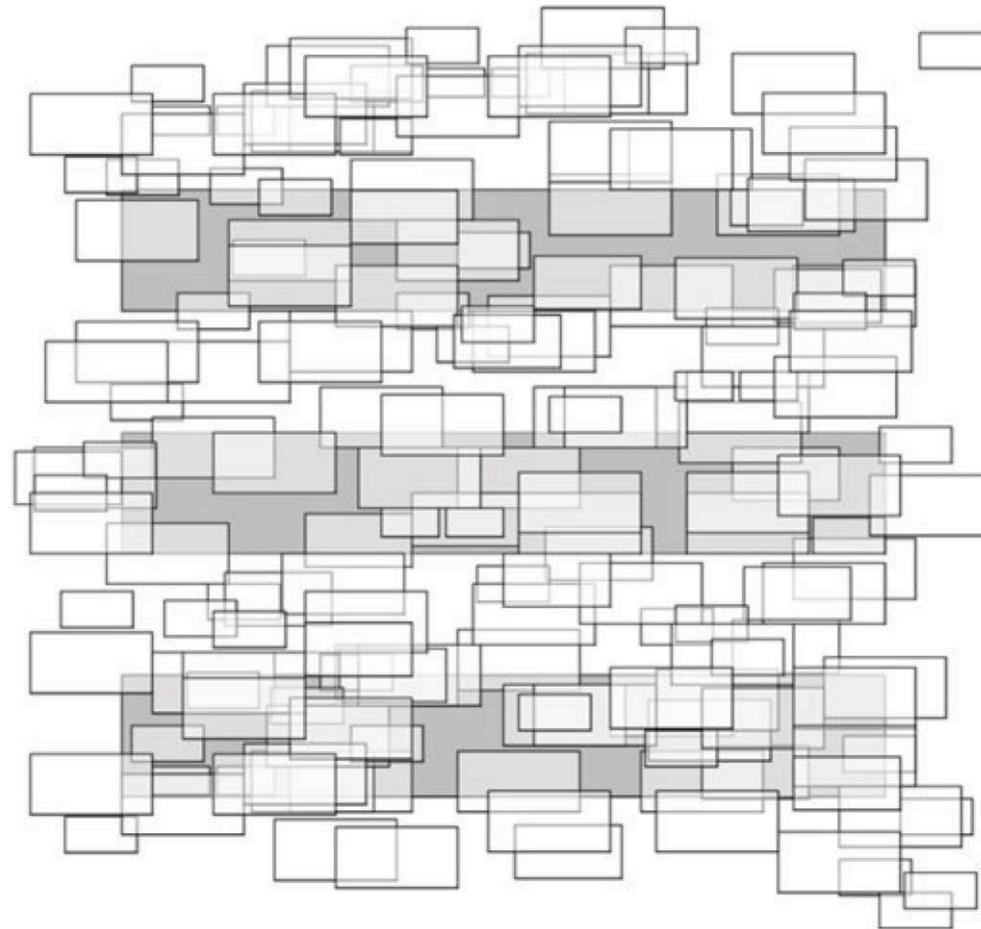
Wann ist welche Rolle angemessen?

“Architektenfaktoren”

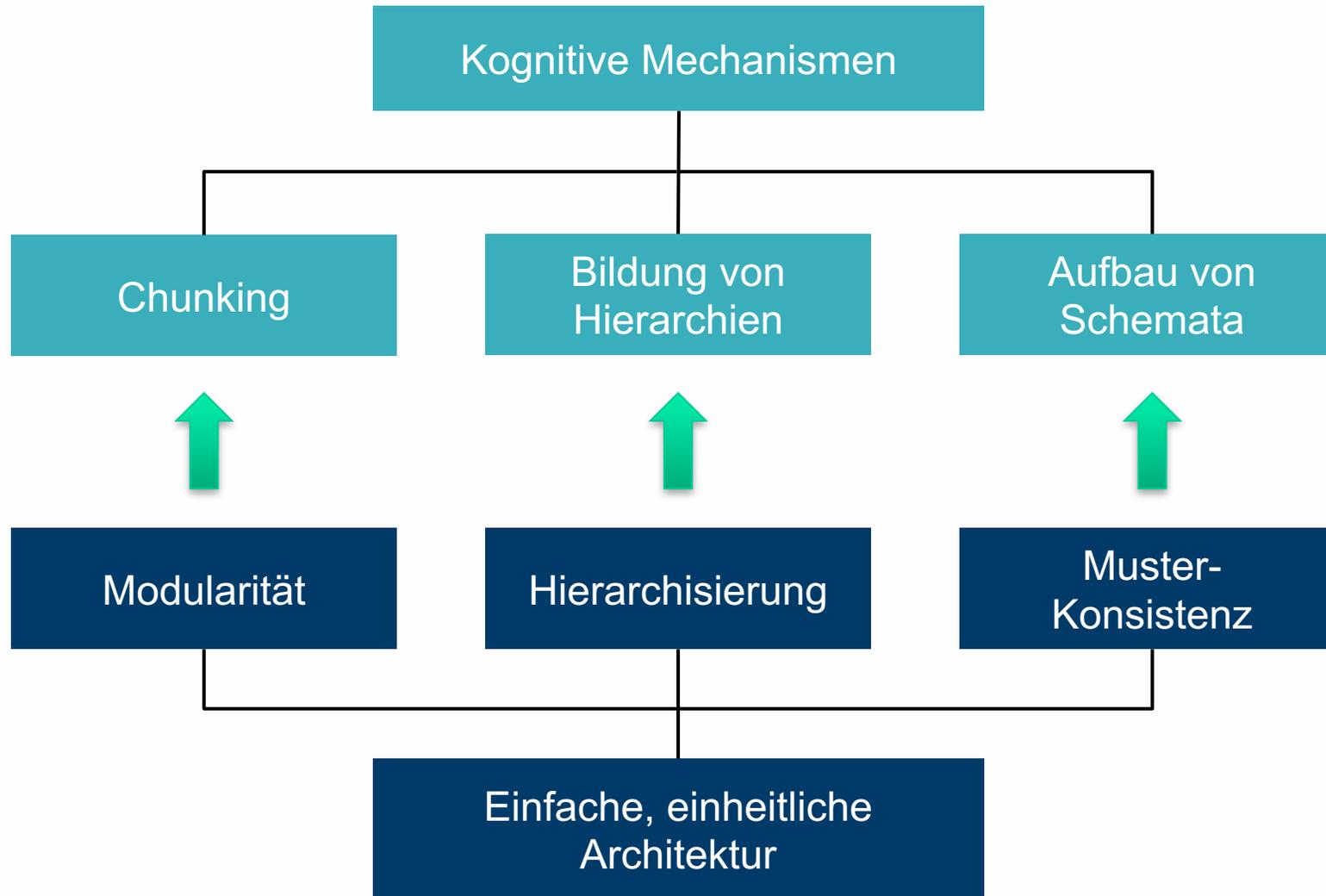
Projektgröße:	mehrere Teams	Vertrautheit:	Erstes Projekt in dieser Zusammensetzung
Verteilung:	geografisch verteilt	Erfahrung:	Viele unerfahrene Entwickler
Fachliche Domäne:	komplex, neu	Disziplin:	Verantwortungsübernahme mangelhaft
Technische Domäne:	schwierig, herausfordernd, neu	Unternehmensstruktur:	stark hierarchisch
Architekturrahmen:	muss erst geschaffen werden	Umfeld:	reguliert oder von Standards bestimmt
Externe Abhängigkeiten:	hoch	Ziele:	Architekturziele in Konflikt (auch zu Projektzielen)



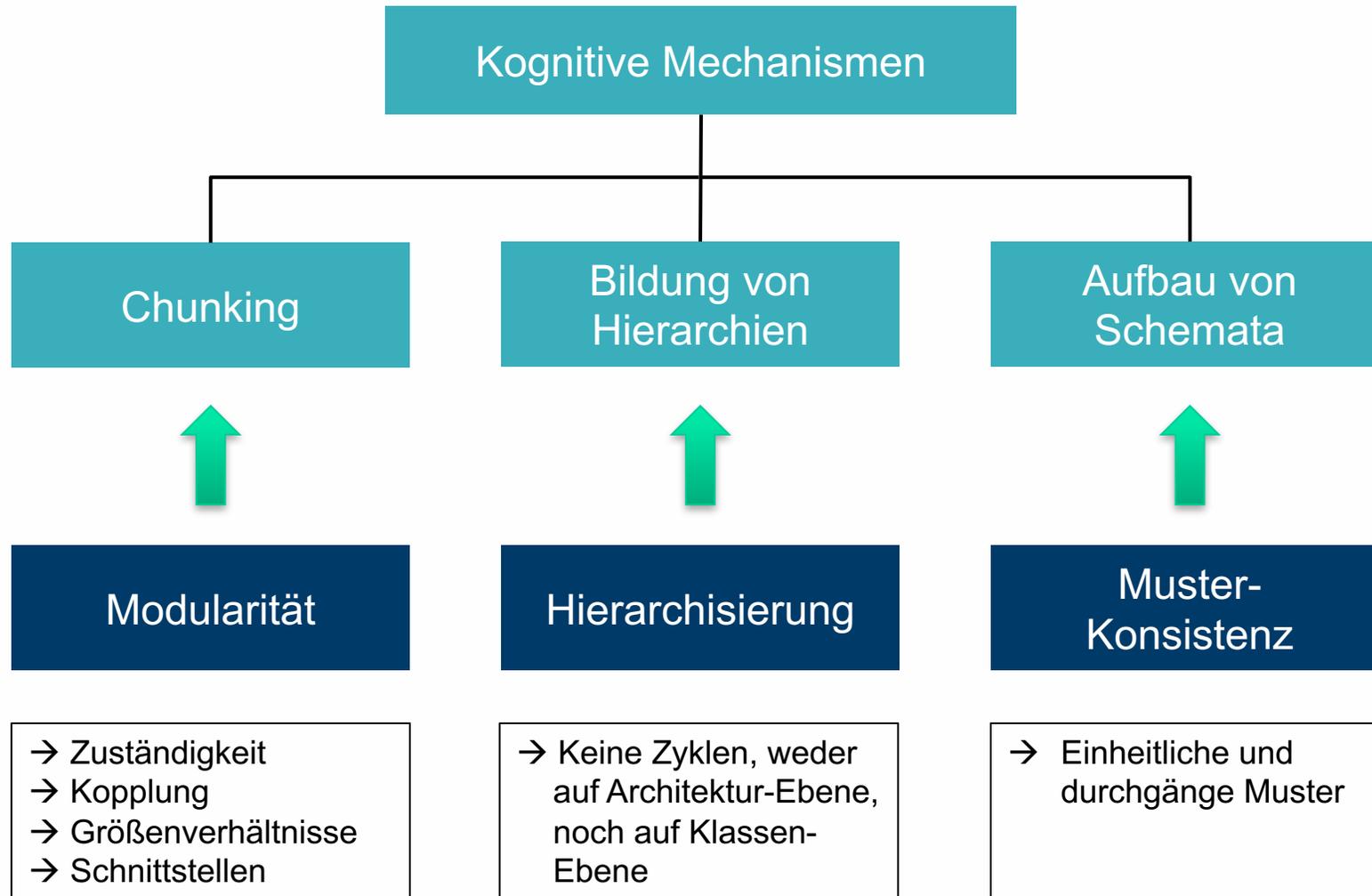
Big Ball of Mud



Komplexität beherrschen



Komplexität beherrschen

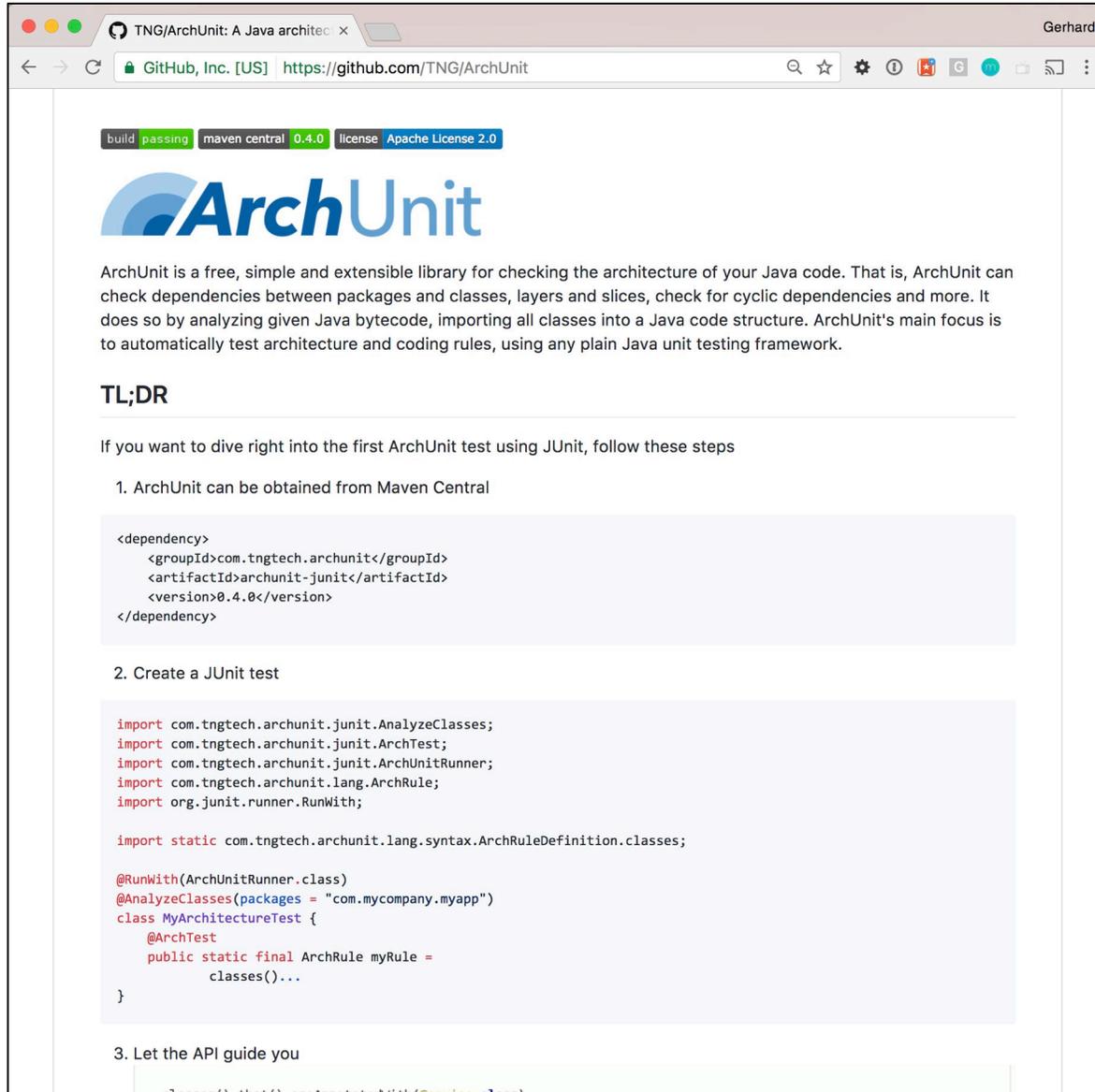


4



Ansätze bei TNG

Testen einer statischen Architektur in einem Produkt



build passing maven central 0.4.0 license Apache License 2.0

ArchUnit

ArchUnit is a free, simple and extensible library for checking the architecture of your Java code. That is, ArchUnit can check dependencies between packages and classes, layers and slices, check for cyclic dependencies and more. It does so by analyzing given Java bytecode, importing all classes into a Java code structure. ArchUnit's main focus is to automatically test architecture and coding rules, using any plain Java unit testing framework.

TL;DR

If you want to dive right into the first ArchUnit test using JUnit, follow these steps

1. ArchUnit can be obtained from Maven Central

```
<dependency>
  <groupId>com.tngtech.archunit</groupId>
  <artifactId>archunit-junit</artifactId>
  <version>0.4.0</version>
</dependency>
```

2. Create a JUnit test

```
import com.tngtech.archunit.junit.AnalyzeClasses;
import com.tngtech.archunit.junit.ArchTest;
import com.tngtech.archunit.junit.ArchUnitRunner;
import com.tngtech.archunit.lang.ArchRule;
import org.junit.runner.RunWith;

import static com.tngtech.archunit.lang.syntax.ArchRuleDefinition.classes;

@RunWith(ArchUnitRunner.class)
@AnalyzeClasses(packages = "com.mycompany.myapp")
class MyArchitectureTest {
    @ArchTest
    public static final ArchRule myRule =
        classes()...
}
```

3. Let the API guide you

- <http://ArchUnit.org>
- Open Source, Apache License 2.0
- Java, ab JDK 1.7
- Aus TNG-Projekten entstanden und von TNG unterstützt
- Refactoring-Safe
- Einführung inkl. Live-Coding: <https://youtu.be/nDWgIvc4zuM>

Testen einer statischen Architektur

```
package com.tngtech.archunit.exampletest;

import javax.persistence.Entity;

import com.tngtech.archunit.core.domain.JavaClasses;
import com.tngtech.archunit.example.persistence.first.InWrongPackageDao;
import com.tngtech.archunit.example.persistence.second.dao.OtherDao;
import com.tngtech.archunit.example.service.ServiceViolatingDaoRules;
import org.junit.Before;
import org.junit.Test;

import static com.tngtech.archunit.lang.syntax.ArchRuleDefinition.classes;

public class DaoRulesTest {
    private JavaClasses classes;

    @Before
    public void setUp() throws Exception {
        classes = new ClassFileImportHelper().importTreesOf(InWrongPackageDao.class, OtherDao.class, ServiceViolatingDaoRules.class);
    }

    @Test
    public void DAOs_must_reside_in_a_dao_package() {
        classes().that().haveNameMatching(regex: ".*Dao").should().resideInAPackage(packageIdentifier: "..dao..")
            .as(newDescription: "DAOs should reside in a package '..dao..'").check(classes);
    }

    @Test
    public void entities_must_reside_in_a_domain_package() {
        classes().that().areAnnotatedWith(Entity.class).should().resideInAPackage(packageIdentifier: "..domain..")
            .as(newDescription: "Entities should reside in a package '..domain..'").check(classes);
    }
}
```

Testen einer statischen Architektur

```
package com.tngtech.archunit.exampletest.junit;

import com.tngtech.archunit.junit.AnalyzeClasses;
import com.tngtech.archunit.junit.ArchTest;
import com.tngtech.archunit.junit.ArchUnitRunner;
import com.tngtech.archunit.lang.ArchRule;
import org.junit.runner.RunWith;

import static com.tngtech.archunit.library.Architectures.layeredArchitecture;

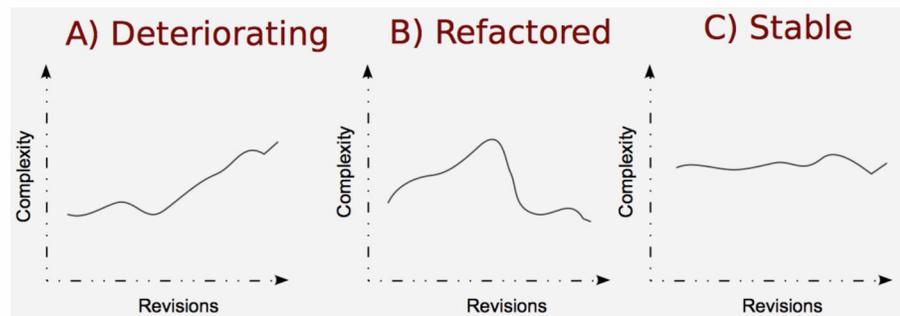
@RunWith(ArchUnitRunner.class)
@AnalyzeClasses(packages = "com.tngtech.archunit.example")
public class LayeredArchitectureTest {
    @ArchTest
    public static final ArchRule layer_dependencies_are_respected = layeredArchitecture()

        .layer( name: "Controllers").definedBy( ...packageIdentifiers: "com.tngtech.archunit.example.controller..")
        .layer( name: "Services").definedBy( ...packageIdentifiers: "com.tngtech.archunit.example.service..")
        .layer( name: "Persistence").definedBy( ...packageIdentifiers: "com.tngtech.archunit.example.persistence..")

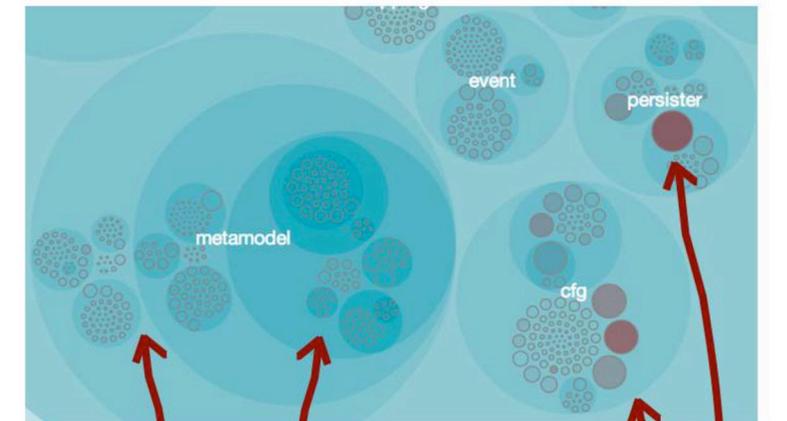
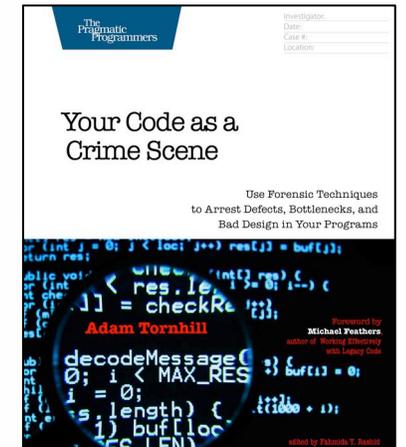
        .whereLayer( name: "Controllers").mayNotBeAccessedByAnyLayer()
        .whereLayer( name: "Services").mayOnlyBeAccessedByLayers( ...layerNames: "Controllers")
        .whereLayer( name: "Persistence").mayOnlyBeAccessedByLayers( ...layerNames: "Services");
}
```

Statische Sicht ist nur eine Teilsicht...

- Veränderungen über die Zeit beobachten
- Methoden aus „Your Code as a Crime Scene“ spannend:
 - Es gibt typischerweise Hotspots bei Veränderungen & Fehlern in wenigen Teilen des Sourcecodes
 - Diese speziell zu betrachten und zu visualisieren macht Sinn!
 - Quelle: Versionsverwaltungssystem & ggf. Issue Management-System



Quelle: Seite 64



Stable parts

Hotspots

Quelle: Seite 43

Architektur-Visualisierung ist wichtig!

Architektur-Visualisierung als Code

- Problem: Diagramme mit separatem Tooling
→ veraltet, falsch, oder automatisch aus Elementen generiert → unübersichtlich
- Daher: Architektur als Klassen/Objekte explizit modellieren, mit verschiedenen Möglichkeiten, Komponenten automatisch zu finden
- <https://github.com/structurizr/java> (Open Source)
- Beispiel: <https://github.com/structurizr/java/blob/master/docs/spring-petclinic.md>
- Ergebnis: <https://structurizr.com/public/1>

The screenshot shows a web browser displaying the GitHub repository for 'structurizr/java'. The page title is 'Structurizr for Java' and the subtitle is 'Visualise, document and explore your software architecture'. The README content includes a list of tools used for visualization: Structurizr (web-based service), PlantUML (UML diagrams), and graphviz (directed graphs). It also provides a code example for creating a software architecture model in Java. The code defines a workspace, adds a person and a software system, and creates a context view. Below the code, a diagram is shown illustrating the relationship between a 'User' and a 'Software System'. The 'User' is represented by a blue circle with a person icon, and the 'Software System' is represented by a blue rectangle. A dashed arrow labeled 'Uses' points from the 'User' to the 'Software System'.

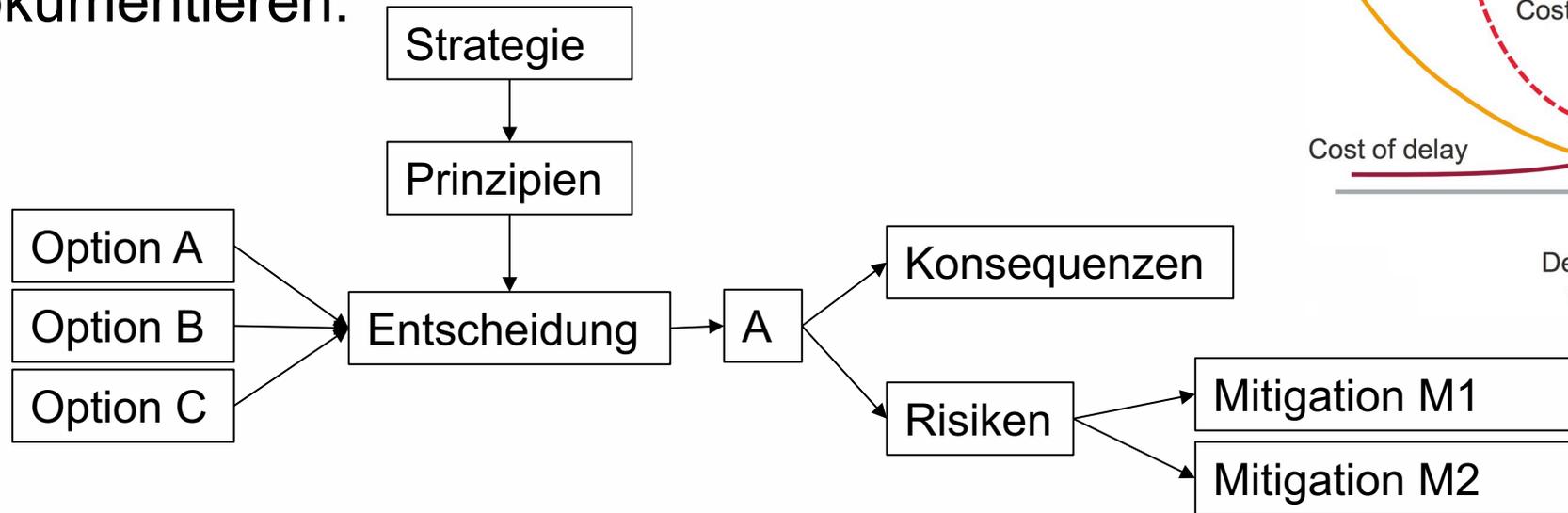
```
Workspace workspace = new Workspace("My model", "This is a model of my software system.");
Model model = workspace.getModel();

Person user = model.addPerson("User", "A user of my software system.");
SoftwareSystem softwareSystem = model.addSoftwareSystem("Software System", "My software system.");
user.uses(softwareSystem, "Uses");

ViewSet viewSet = workspace.getViews();
SystemContextView contextView = viewSet.createSystemContextView(softwareSystem, "context", "A simple example of a Sys
contextView.addAllSoftwareSystems();
contextView.addAllPeople();
```

Entscheidungen

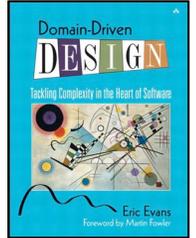
- Architektur-relevante Entscheidungen sorgfältig treffen und dokumentieren:



- Wie? „Architecture Decision Log“, z.B. in einem Wiki, mit Template, mit Architecture Decision Records, z.B.:

- Nummer, Titel, Kontext, Entscheidung, Status, Konsequenzen
- Siehe auch <http://thinkrelevance.com/blog/2011/11/15/documenting-architecture-decisions>

Domain Driven Design



- Einheitliches Vokabular
- Bounded Context als Verständnis dafür, dass z.B. ein Kunde in einem Kontext nicht zwangsweise die gleichen Attribute hat wie ein Kunde in einem anderen Kontext

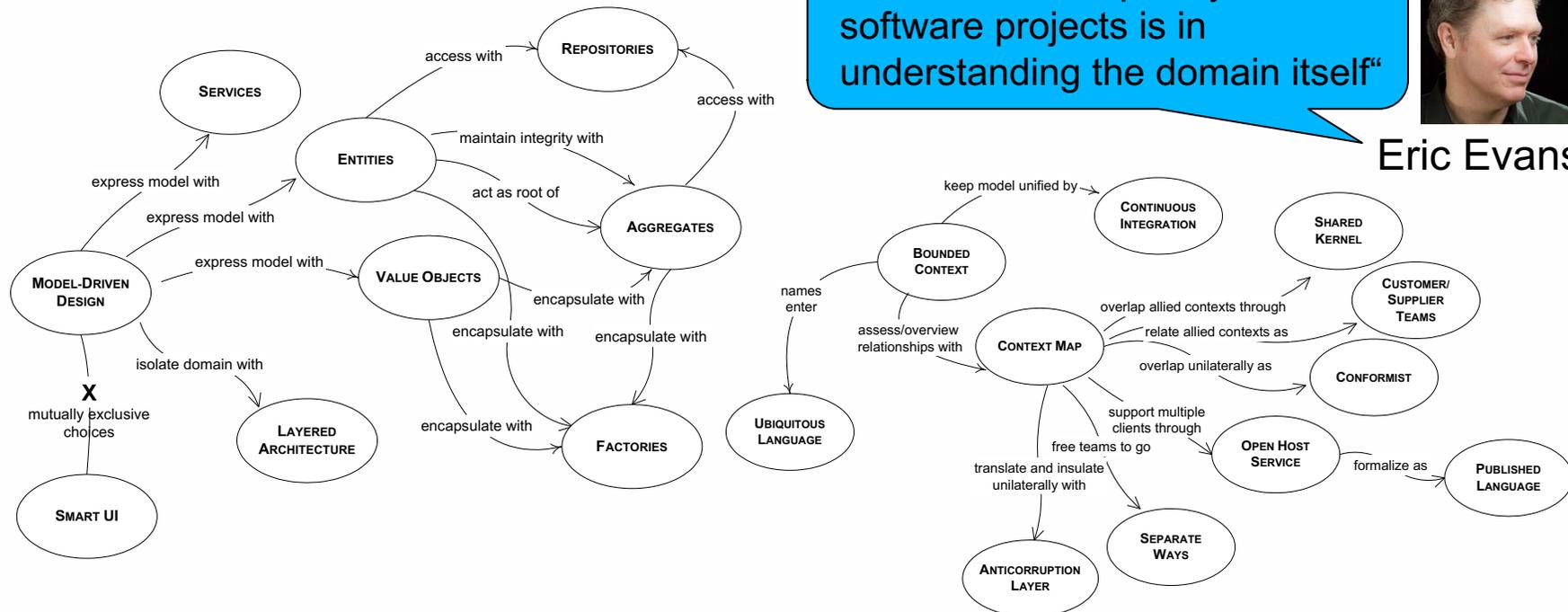
- Entkopplung!

- Bei TNG: Investition in entsprechende Schulungen & Konferenzen & interne Workshops

„The critical complexity of most software projects is in understanding the domain itself“



Eric Evans



Neue Architekturen: Domänenmodell als Kern

- Hexagonale Architektur

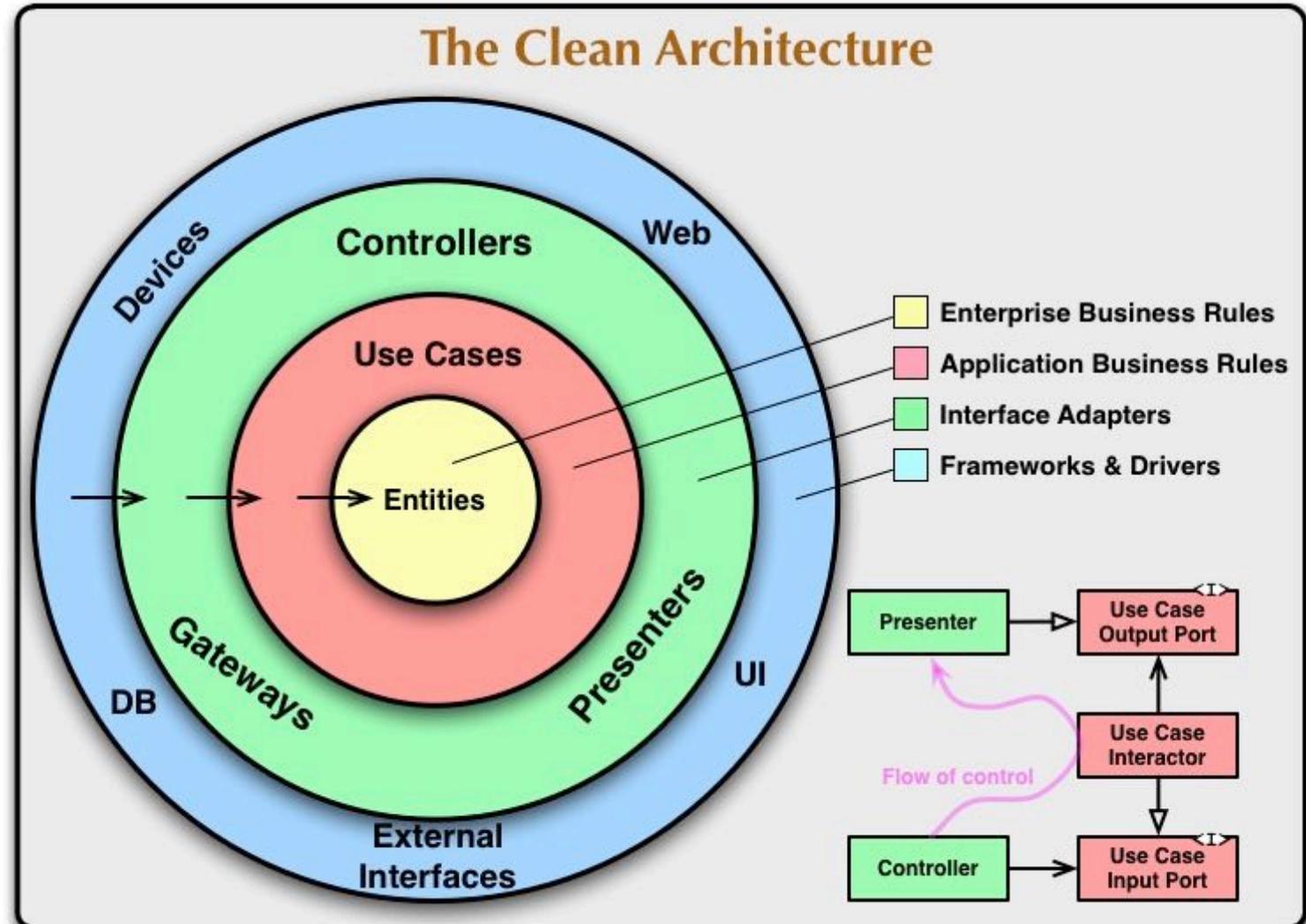
- <http://alistair.cockburn.us/Hexagonal+architecture> (2005)

- Zwiebel-Architektur

- <http://jeffreypalermo.com/blog/the-onion-architecture-part-1/> (2008)

- Saubere Architektur

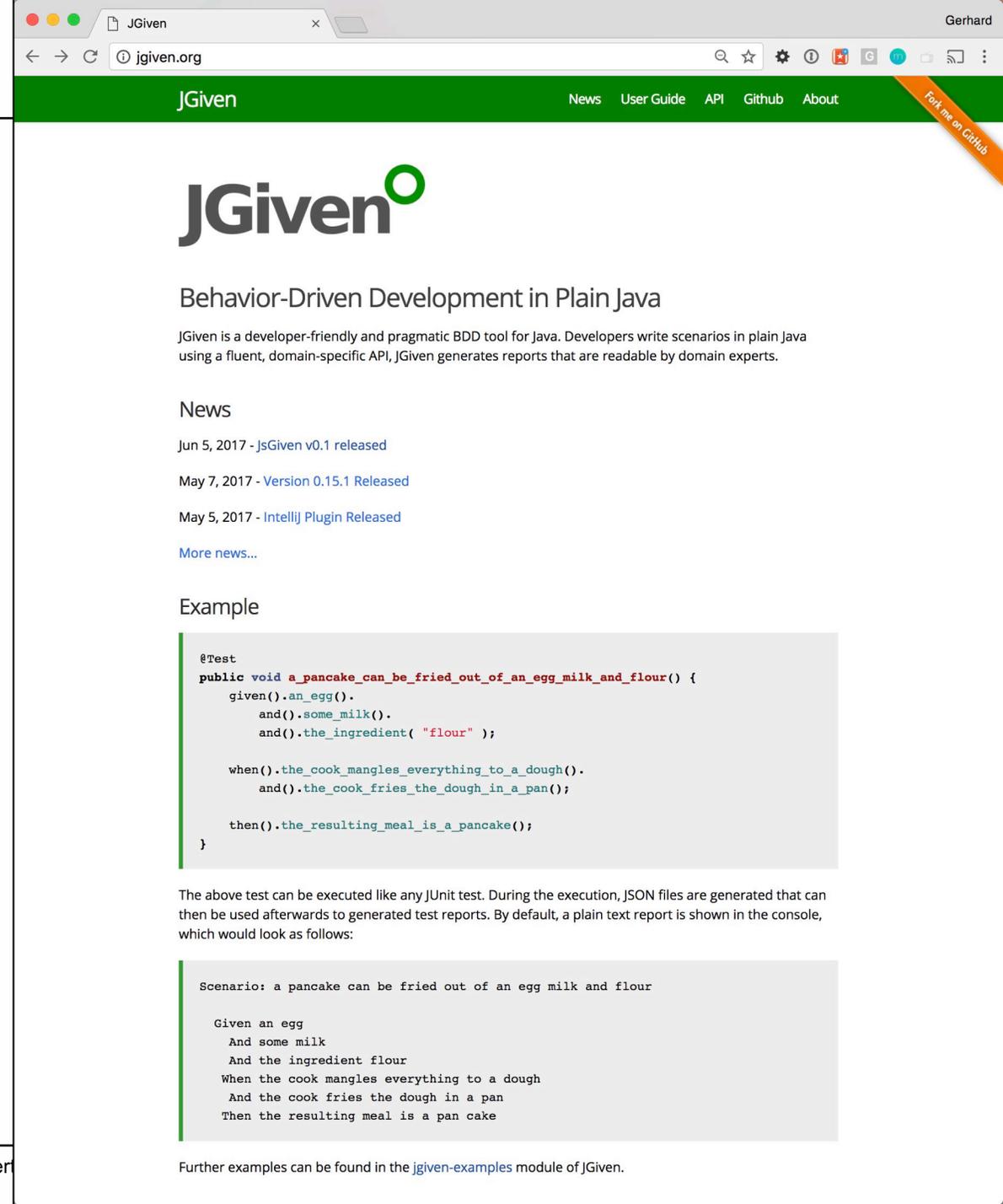
- <https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html> (2012)



Fachlichkeit ist wichtig!

JGiven.org

- BDD-Framework
- Besonders für Akzeptanztests geeignet
- Code ist Dokumentation
- Dokumentation ist mit Code verbunden
- Steht unter Continuous Integration
- Open Source
- Bisher nur für Java, erste Version für JS
- Einführungsvortrag:
 - <https://youtu.be/t-jvr3XGuaA> (deutsch) oder https://youtu.be/x-6bT_0dTWI (englisch)



The screenshot shows the JGiven website homepage. The browser address bar displays 'jgiven.org'. The navigation bar includes 'JGiven', 'News', 'User Guide', 'API', 'Github', and 'About'. A 'Fork me on GitHub' button is visible in the top right corner. The main content area features the JGiven logo, the tagline 'Behavior-Driven Development in Plain Java', and a brief description of the tool. Below this, there is a 'News' section with three entries: 'Jun 5, 2017 - JsGiven v0.1 released', 'May 7, 2017 - Version 0.15.1 Released', and 'May 5, 2017 - IntelliJ Plugin Released'. An 'Example' section follows, showing a Java code snippet for a BDD test. The code defines a test method 'a_pancake_can_be_fried_out_of_an_egg_milk_and_flour()' with 'given', 'when', and 'then' blocks. Below the code, a paragraph explains that the test can be executed like any JUnit test and that JSON files are generated for reports. Finally, a 'Scenario' section shows the human-readable output of the test, which reads like a story: 'Scenario: a pancake can be fried out of an egg milk and flour', followed by 'Given an egg', 'And some milk', 'And the ingredient flour', 'When the cook mangles everything to a dough', 'And the cook fries the dough in a pan', and 'Then the resulting meal is a pan cake'.

JGiven

News User Guide API Github About

JGiven

Behavior-Driven Development in Plain Java

JGiven is a developer-friendly and pragmatic BDD tool for Java. Developers write scenarios in plain Java using a fluent, domain-specific API, JGiven generates reports that are readable by domain experts.

News

Jun 5, 2017 - JsGiven v0.1 released

May 7, 2017 - Version 0.15.1 Released

May 5, 2017 - IntelliJ Plugin Released

[More news...](#)

Example

```
@Test
public void a_pancake_can_be_fried_out_of_an_egg_milk_and_flour() {
    given().an_egg().
        and().some_milk().
        and().the_ingredient( "flour" );

    when().the_cook_mangles_everything_to_a_dough().
        and().the_cook_fries_the_dough_in_a_pan();

    then().the_resulting_meal_is_a_pancake();
}
```

The above test can be executed like any JUnit test. During the execution, JSON files are generated that can then be used afterwards to generated test reports. By default, a plain text report is shown in the console, which would look as follows:

```
Scenario: a pancake can be fried out of an egg milk and flour

Given an egg
And some milk
And the ingredient flour
When the cook mangles everything to a dough
And the cook fries the dough in a pan
Then the resulting meal is a pan cake
```

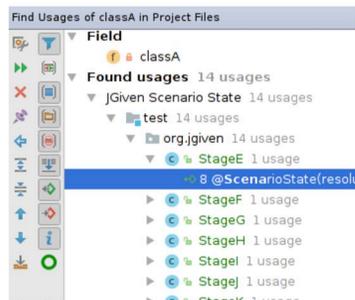
Further examples can be found in the [jgiven-examples](#) module of JGiven.

JGiven

Compatible with IntelliJ IDEA

05.05.2017 40 ☆☆☆☆☆

Provides support for navigation between JGiven scenario states.



✉ matthias.klass@tngtech.com

<> [Source code](#)

📄 [License](#)

🔍 [Issue tracker](#)

📖 [Documentation](#)

Vendor: [TNG Technology Consulting GmbH](#)

Authors: matthias.klass@tngtech.com
pavel.fischer@tngtech.com

Download plugin

VERSION	COMPATIBLE BUILDS	UPDATE DATE	
0.0.4	123+	05.05.2017	DOWNLOAD
0.0.3	123+	27.04.2017	DOWNLOAD

General usage instructions

All scenario states with usages are annotated with a line marker. When looking for usages of a scenario state field, IntelliJ will also show all other fields referencing this scenario state. To make those lists more readable, an additional usage type for JGiven scenario states is available. Within this list, all non scenario state usages can be filtered out by clicking the JGiven icon.

Please [sign in](#) to leave a comment.

No comments so far.

README.md

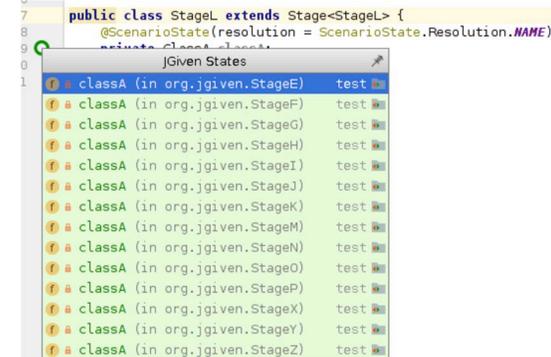
build passing license Apache License 2.0

JGiven Plugin for IntelliJ IDEA

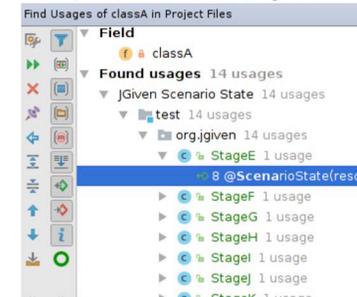
IntelliJ IDEA plugin to support both navigating between scenario states and finding usages of scenario states within JGiven test stages.

Features

- All scenario states with usages are annotated



- When looking for usages of a scenario state field, IntelliJ will also show all other fields referencing this scenario state. To make those lists more readable, an additional usage type for JGiven scenario states is available. Within this list, all non scenario state usages can be filtered out by clicking the JGiven icon.



Building

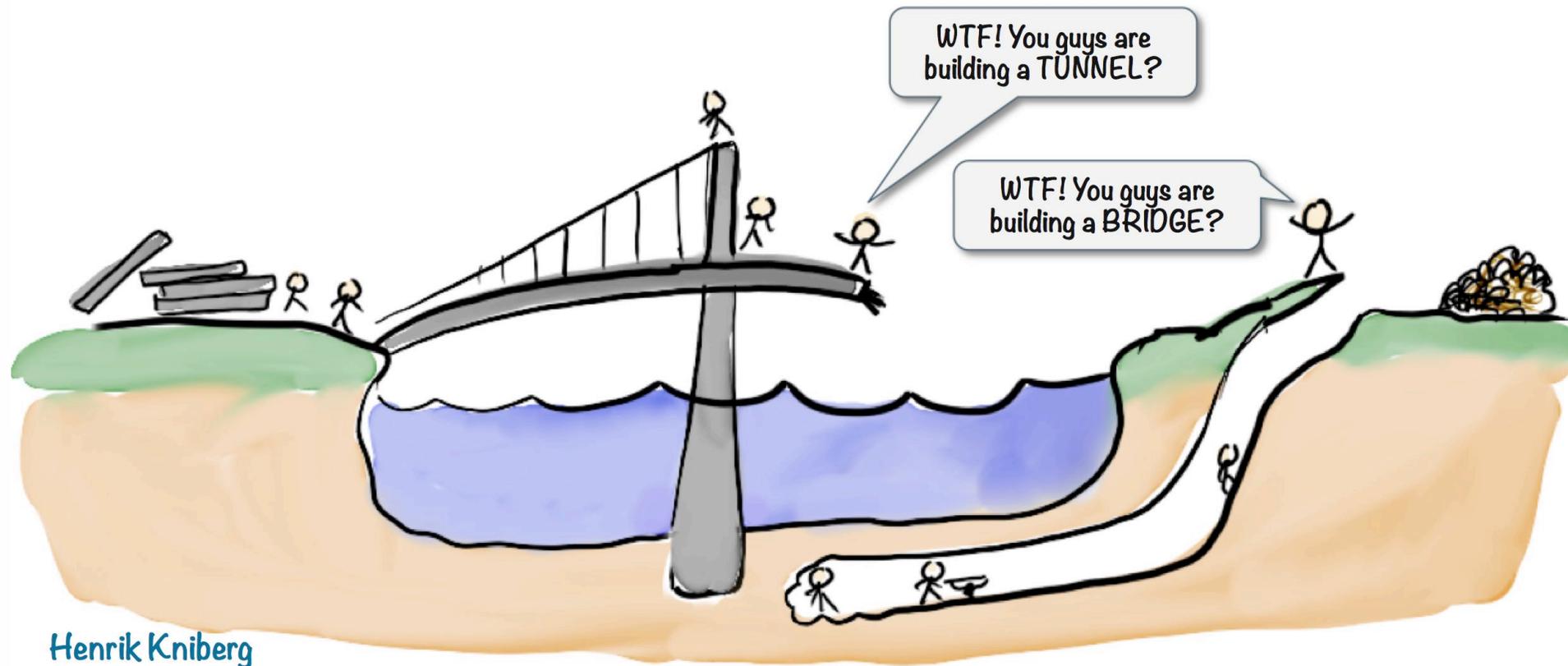
Checkout all source files and run `./gradlew buildPlugin` from within the checkout directory.

The installable artifact can be found at `build/distributions/jgiven-intellij-plugin.zip`.

Installation

Größere Systeme: Achtung!

Misalignment

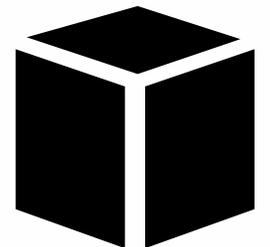
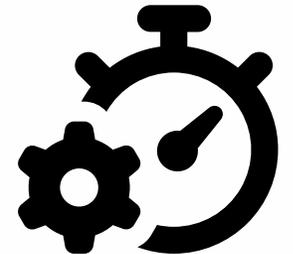
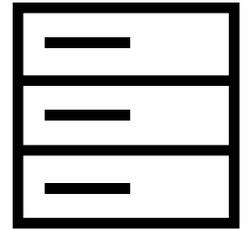


Mehr als ein Team? Mehr Arbeit...

- Emergente Architekturen über mehr als ein Team sind eine Herausforderung
 - Kontext pro Team oft zu speziell, Blick für das Ganze
- DDD zur Entkopplung von Teams
 - Nur die Dinge festzurren, die man unbedingt braucht (→ hängen andere Teams davon ab?)
 - Ist menschlich eine Herausforderung
- Gemeinsame GUI als zentrale Integration?
 - Gemeinsame Oberflächentechnologie notwendig
- Architekten schauen, dass so wenig wie möglich Entscheidungen getroffen werden, die nicht mehr reversibel sind

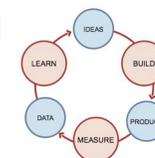
Infrastruktur

- Entwicklungsteams sollten möglichst hohe Kontrolle über ihre Infrastruktur haben
 - Gerade klassische Betriebsteams und Prozesse können sehr bremsen
 - Mit Cloud-Umgebungen ist die Entwicklungsgeschwindigkeit von „Infrastructure As Code“ sehr viel höher
 - Potential der Optimierung von Monaten zu Minuten (!)
 - End-to-End-Verantwortung hilft sehr (You Write It – You Change It – You Run It)
 - Klassische Betriebsorganisationen sind mit Microservices, Docker & Co. häufig überfordert
- Bei TNG: jeder Kollege hat Zugang zur vollen AWS Infrastruktur
 - jeder kann sich entsprechende Credentials selbst erzeugen



Was kann man noch tun?

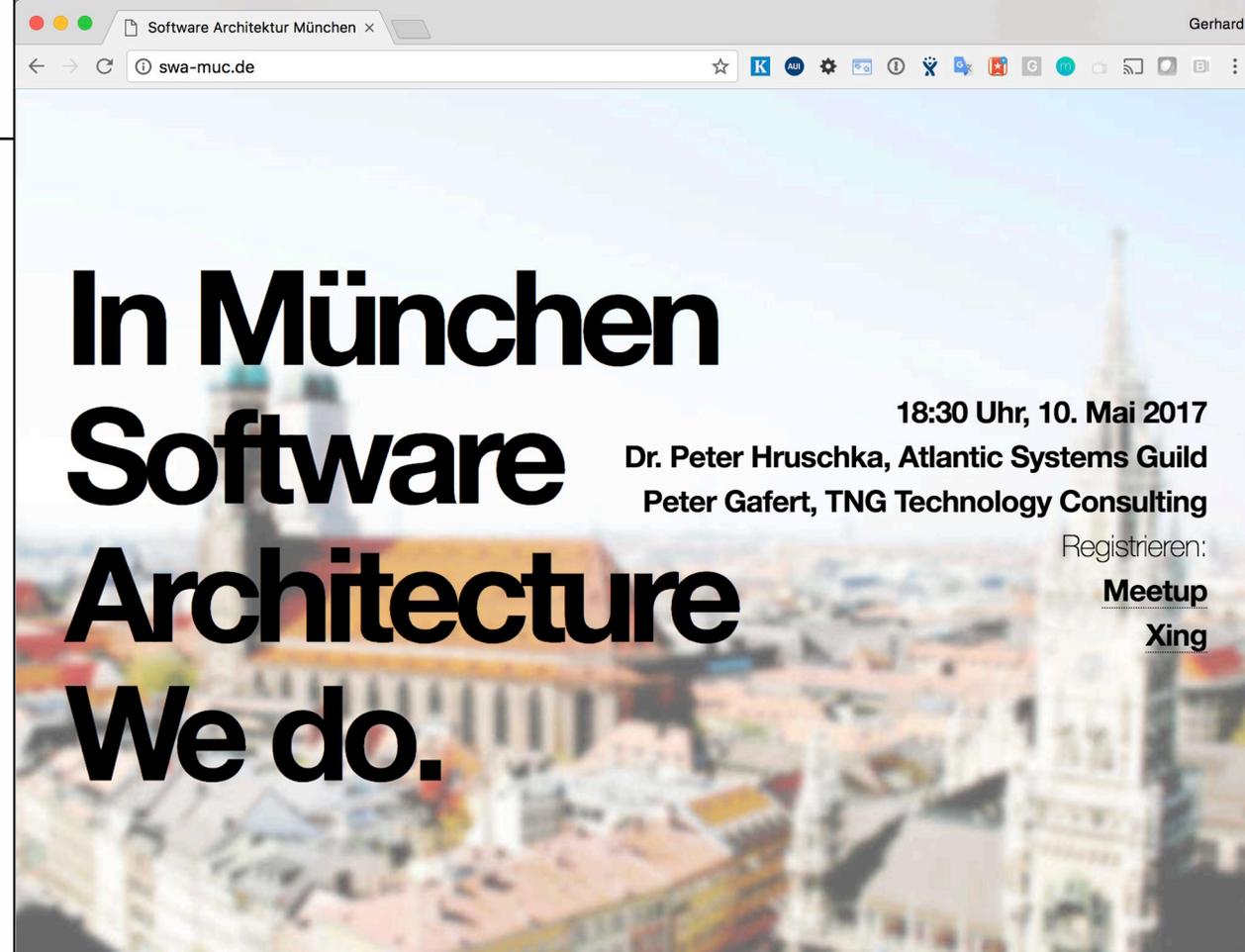
- Zeit geben für Weiterbildung: Techdays, Reading Groups, regelmäßige interne Schulungen
- Awareness für Software-Architektur-Themen erzeugen
- Zertifizierungen / <http://www.isaqb.org/> nach CPSA-F und CPSA-A
- Durchführung von Architektur-Katas
 - wie Code-Katas, nur für Architektur-Themen
- Tooling bei Kunden verbessern
 - [Atlassian Consulting Team von TNG](#): Einführung von Confluence, JIRA & Co.
- Lean-Startup-Methoden erklären & Ansätze zum Einsatz suchen (Requirements vs. Hypothesen, Messen & Analysieren)



Es gibt eine Community zu SW-Architektur-Themen

Software Architektur München

- Gegründet im September 2016, bisher 6 Treffen, letztes heute 😊
- Regelmäßige Veranstaltungen in München zum Themen im Umfeld Software-Architektur
- <http://swa-muc.de>
- Videos der letzten Veranstaltungen sind verfügbar, siehe Wiki
- Über 970 Mitglieder in der Meetup-Gruppe <https://www.meetup.com/de-DE/Software-Architektur-Muenchen/>



Software Architektur München x

swa-muc.de

In München Software Architecture We do.

18:30 Uhr, 10. Mai 2017
Dr. Peter Hruschka, Atlantic Systems Guild
Peter Gafert, TNG Technology Consulting

Registrieren:
[Meetup](#)
[Xing](#)

IT-Architekten - Ihre Aufgaben und Verantwortungen

Speaker: [Dr. Peter Hruschka, Atlantic Systems Guild](#)

Je nach Projektgrößenordnung tragen Architekten verschiedene Verantwortungen - manche offensichtlich und andere weniger offensichtlich. Die Übernahme dieser Verantwortungen macht den Reiz dieses Berufsbildes aus. In dem Vortrag gibt Peter Hruschka Denkanstöße dazu. Angefangen von Architekten, die innerhalb eines Produktes die Verantwortung für ein Subsystem übernehmen, über Gesamtproduktverantwortung, bis hin zu Architekten für

Es gibt eine Community zu SW-Architektur-Themen

SwaCamp.org

- Un-Conferenz (Open Space / Barcamp) speziell zum Thema Software-Architektur und –Design
- Organisiert von einer übergreifenden Community, insbesondere iteratec, Scandio und TNG
- Von Freitag, 8. September (abends) bis 10. September, bis 150 Personen
- In den Räumen von TNG Technology Consulting GmbH, Unterföhring
- Tickets zum Selbstkostenpreis von 50€

SwaCamp Munich 2017

swacamp.org/#

HOME ÜBER ZEITPLAN ANMMMLDUNG VERHALTENSKODEX LOCATION ENGLISH

SwaCamp 2017

... ist eine Un-Konferenz zu den Themen Software Architektur und Design, die vom **8. - 10. September 2017** in München stattfindet.

Unsere Sponsoren

 **iteratec**
KOMPETENZ, DIE ENTLASTET

 **scandio**
SOFTWARE & CONSULTING

 **TNG** TECHNOLOGY CONSULTING

Über die Un-Konferenz

Was ist Un-Conf / Barcamp?

Das SwaCamp ist eine Un-Conf oder ein Barcamp. Dem zugrunde liegt das *Open Space* Format.

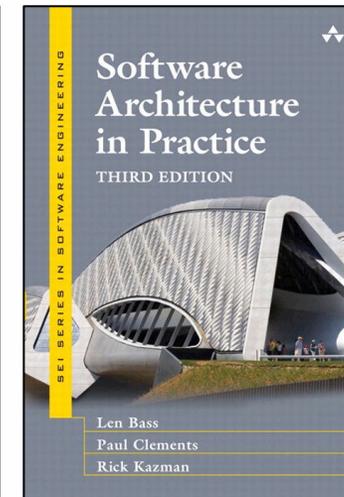
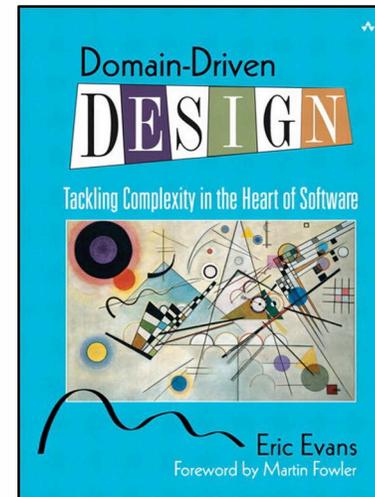
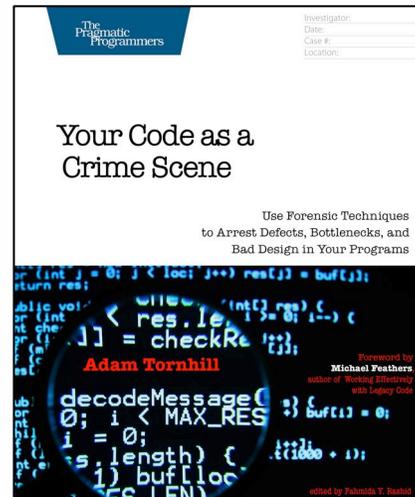
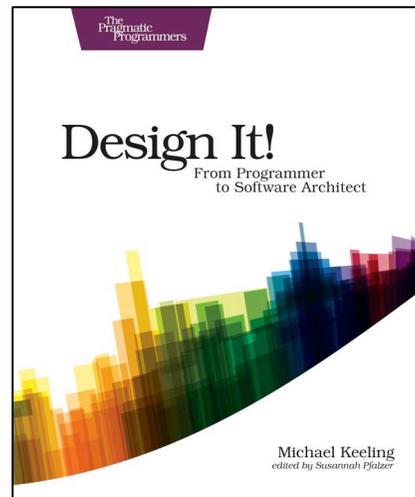
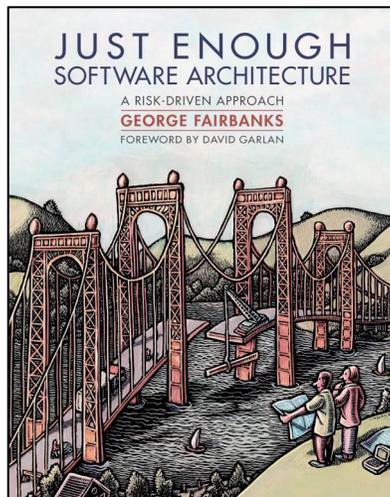
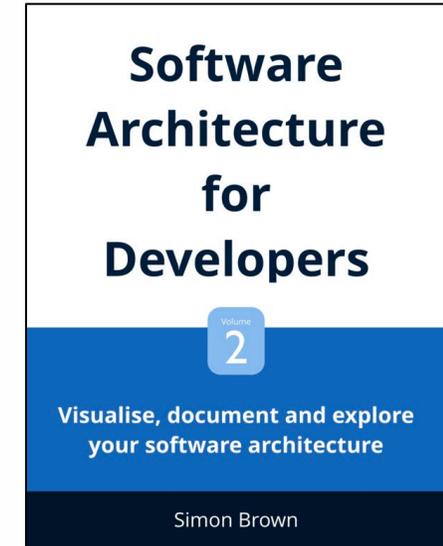
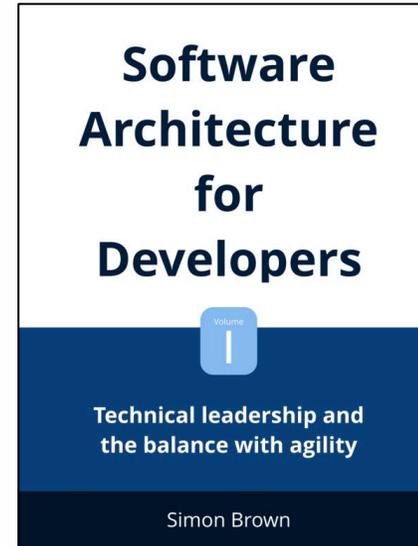
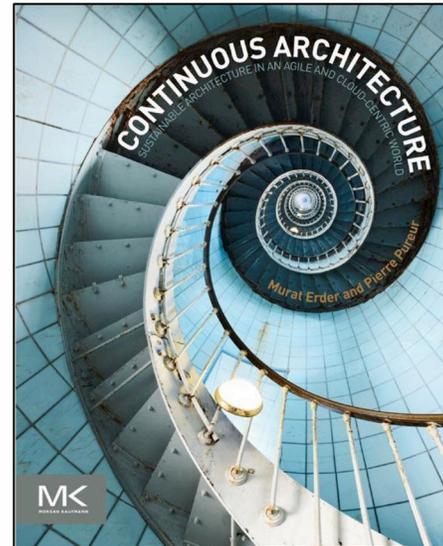


Zusammenfassung

Zusammenfassung

- Gute Software-Architektur ist wertvoll!
- Gute Software-Architektur entsteht auch im agilen Kontext nicht von alleine, sondern muss aktiv erarbeitet werden
- Dafür braucht es geeignete Personen, Organisation, **Disziplin**
- **Und damit Stabilität für Agilität!**

Literatur



Ende

#NoQuestions

Lieber Gespräche und Diskussionen 😊



Gerhard Müller
Dipl. Inf. (Univ)
Partner



TNG Technology Consulting GmbH
Betastr. 13a
85774 Unterföhring

Tel. +49 89 2158 9960

Fax +49 89 2158 9969

Mobil +49 179 133 8060

Gerhard.Mueller@tngtech.com