22.01.2019
MÜNCHEN, OOP 2019

Mit Werkzeugen von morgen Software von gestern systematisch verbessern

Markus Harrer
Senior Consultant

Twitter: @feststelltaste
Folien: speakerdeck.com/feststelltaste

INNOQ

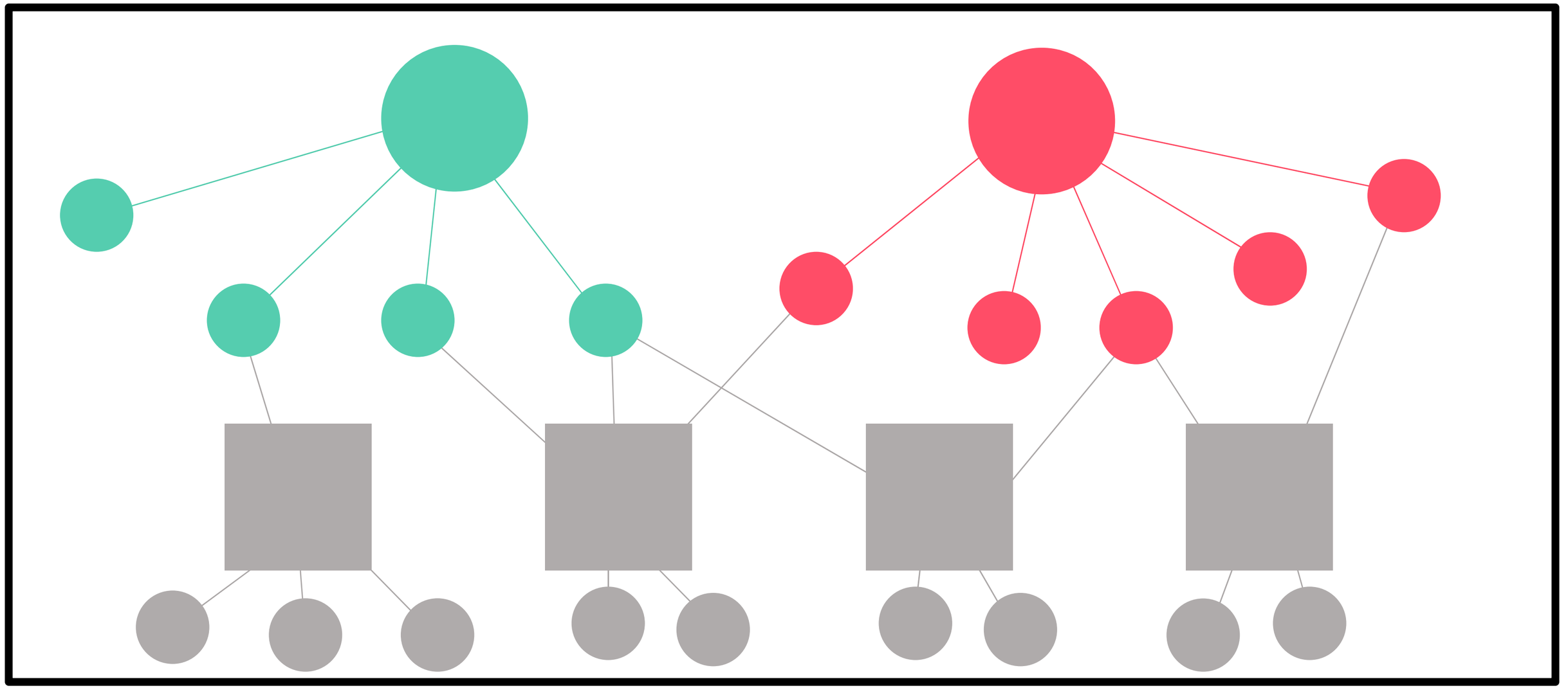
oop
software meets business

Disclaimer

„Werkzeuge von morgen“

- Es gibt sie schon heute
- Auch „alte“ Werkzeuge im Vortrag!

Agenda



Motivation

Bank

Touristik

LOGISTIK

https://www.redbubble.com/de/people/gungable/works/23615911-legacy-code (promotional)

Herausforderung

https://www.flickr.com/photos/spongebabyalwaysfull/20782138502/in/photolist-xErTJN-bhpoWe-wJgkR5

Softwarearchäologie

https://www.flickr.com/photos/jimoftheday/2537608919/

Arbeit, die zählt

https://www.flickr.com/photos/149368236@N06/33882279845/

Unmittelbares Feedback

https://www.flickr.com/photos/kwirth/9596521368/

Moderne Werkzeuge

https://www.flickr.com/photos/charles_hudson/5815634885/

Legacy Code

Software von gestern

Was ist überhaupt Legacy Code?

"Code without tests."

Michael Feathers

"Code that you are scared to change."

Jeff Foster

"Code without communication artifacts."

Andrea Goulet

Was ist überhaupt Legacy Code?

"Code without communication artifacts."

Andrea Goulet

Kommunikationsartefakte

Code

Continuous Integration

Musterkatalog

Tests

Nutzungsstatistiken

Build-Skripte

Bug-Tracker

Log-Dateien

Versionshistorie

Qualitätsberichte

Architekturdokumentation

Protokolle

Anforderungen

Betriebshandbuch

E-Mails

Foren

Gespräche

Meetings

Telefonate

Admins

Entwickler

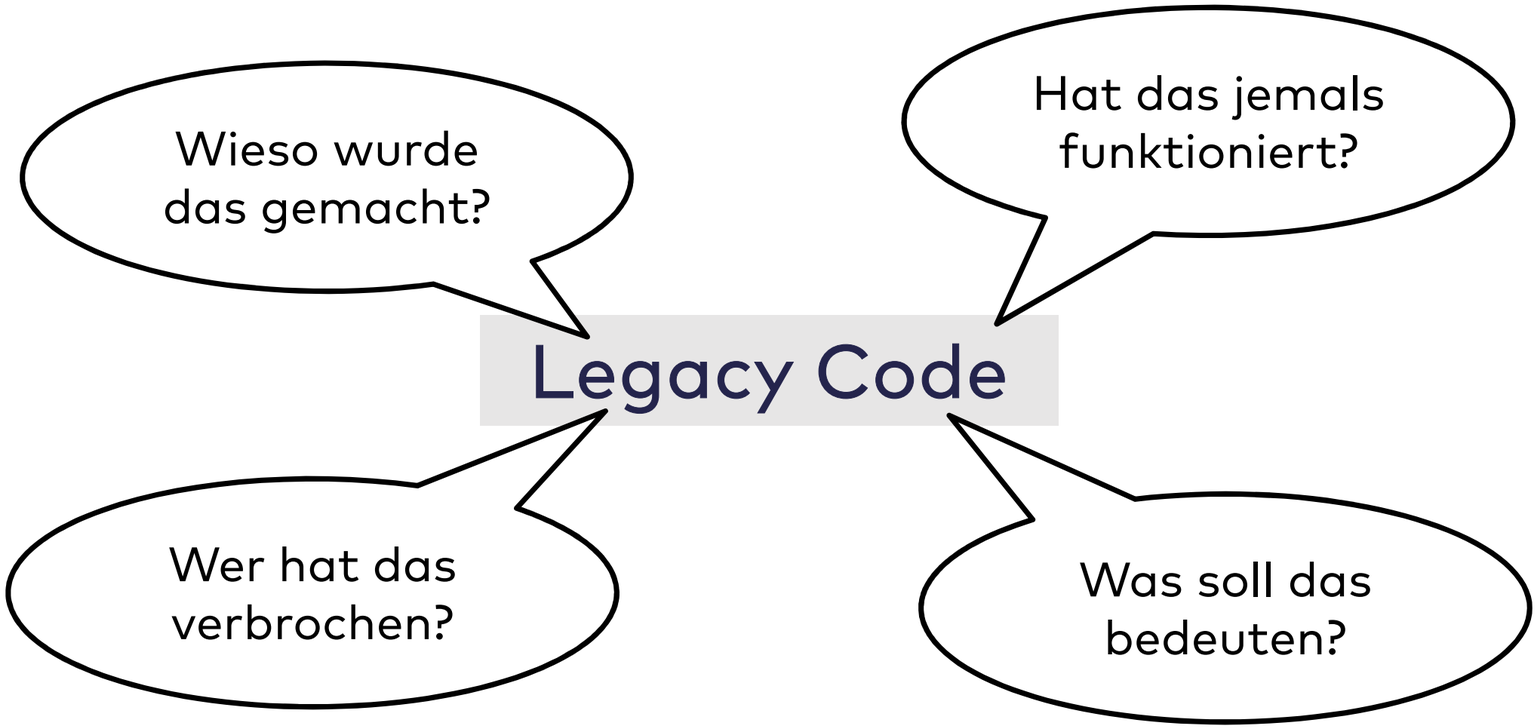
Fachexperten

Kunden

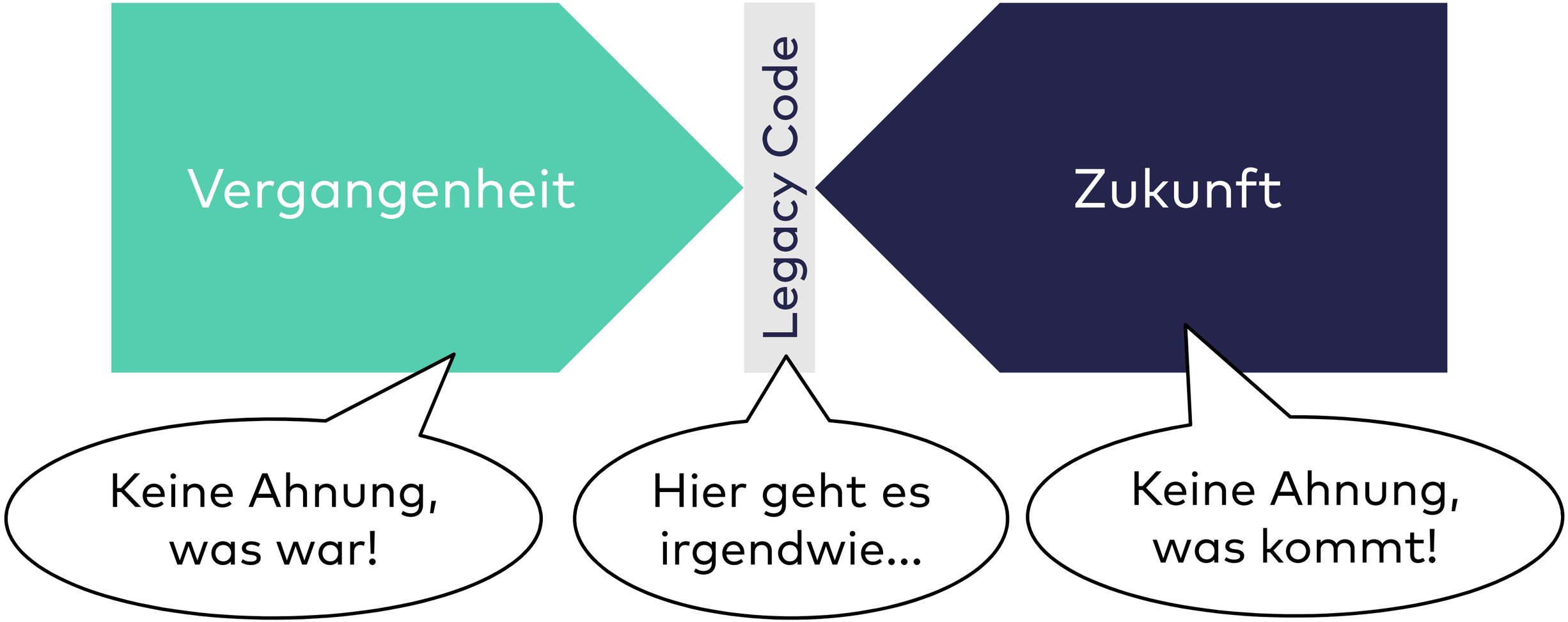
Keine Kommunikationsartefakte

Legacy Code

Keine Kommunikationsartefakte



Gefangen im Jetzt



RIP Legacy Code

Legacy Code wieder lebendig machen

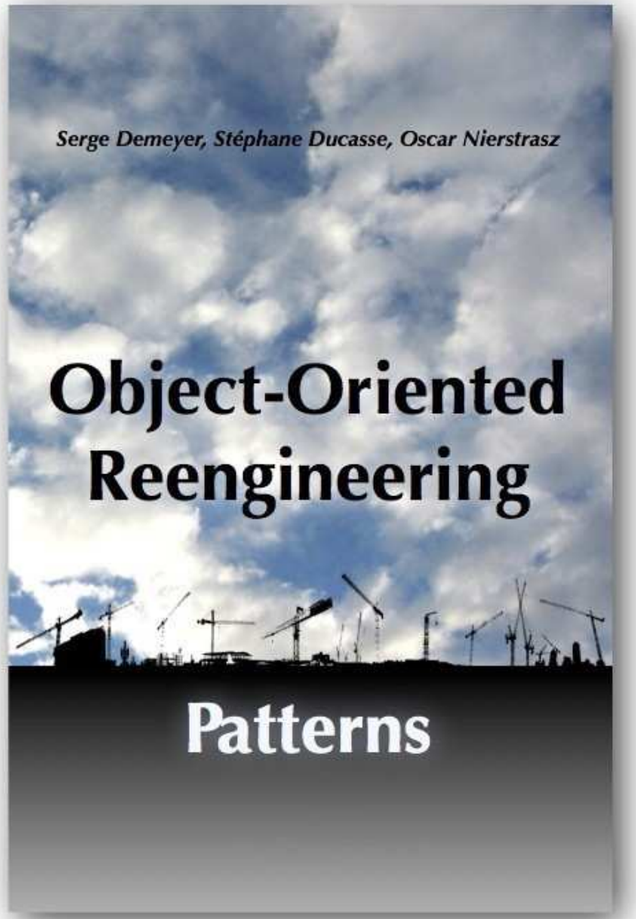
Legacy Code wieder lebendig machen

Moderne Werkzeuge

Methodenkataloge

zum Umgang mit Legacy Code

OORP & aim42



aim42 Method Reference

Fork me on GitHub

About aim42

aim 42 supports software evolution, maintenance, migration and improvement - in a systematic and pragmatic way.

aim42 is a collection of practices and patterns to support software evolution, modernization, maintenance, migration and improvement of software systems:

1. helps to optimize your software and reduce maintenance cost,
2. identifies critical issues, technical debt and risks,
3. supports both business and technical stakeholders,
4. grounded in practice and experience, backed by serious research,
5. free and open-source, contributions welcome.

aim42 seamlessly integrates with your day-to-day development work.

Authored by the aim42 community, lead by Dr. Gernot Starke <gernot.starke@innoq.com>

Version 0.7.7 Date Nov 12, 2018 build passing issues 87 open stars 113 contributors 18 Followers 3k

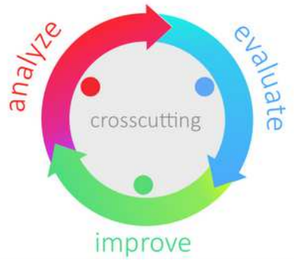
About this documentation

This document serves as the *method reference* - it collects practices and patterns. Please don't expect a *user manual* or step-by-step guidebook here - we're currently busy working on the latter.

1. Introduction

1.1. Overview

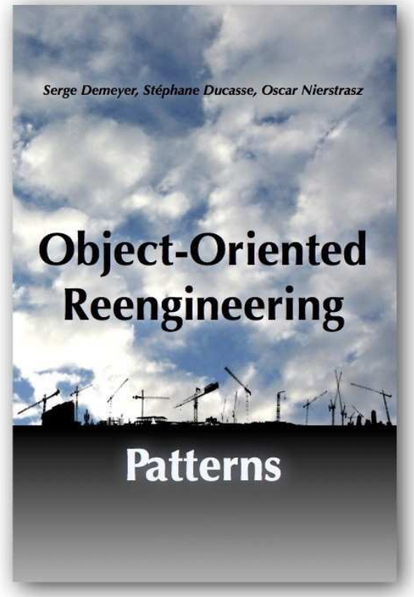
aim42 organizes software improvement in three major phases (Chapter 2, *Analyze*, Chapter 3, *Evaluate* and Chapter 4, *Improve*), build around some crosscutting activities.



OORP

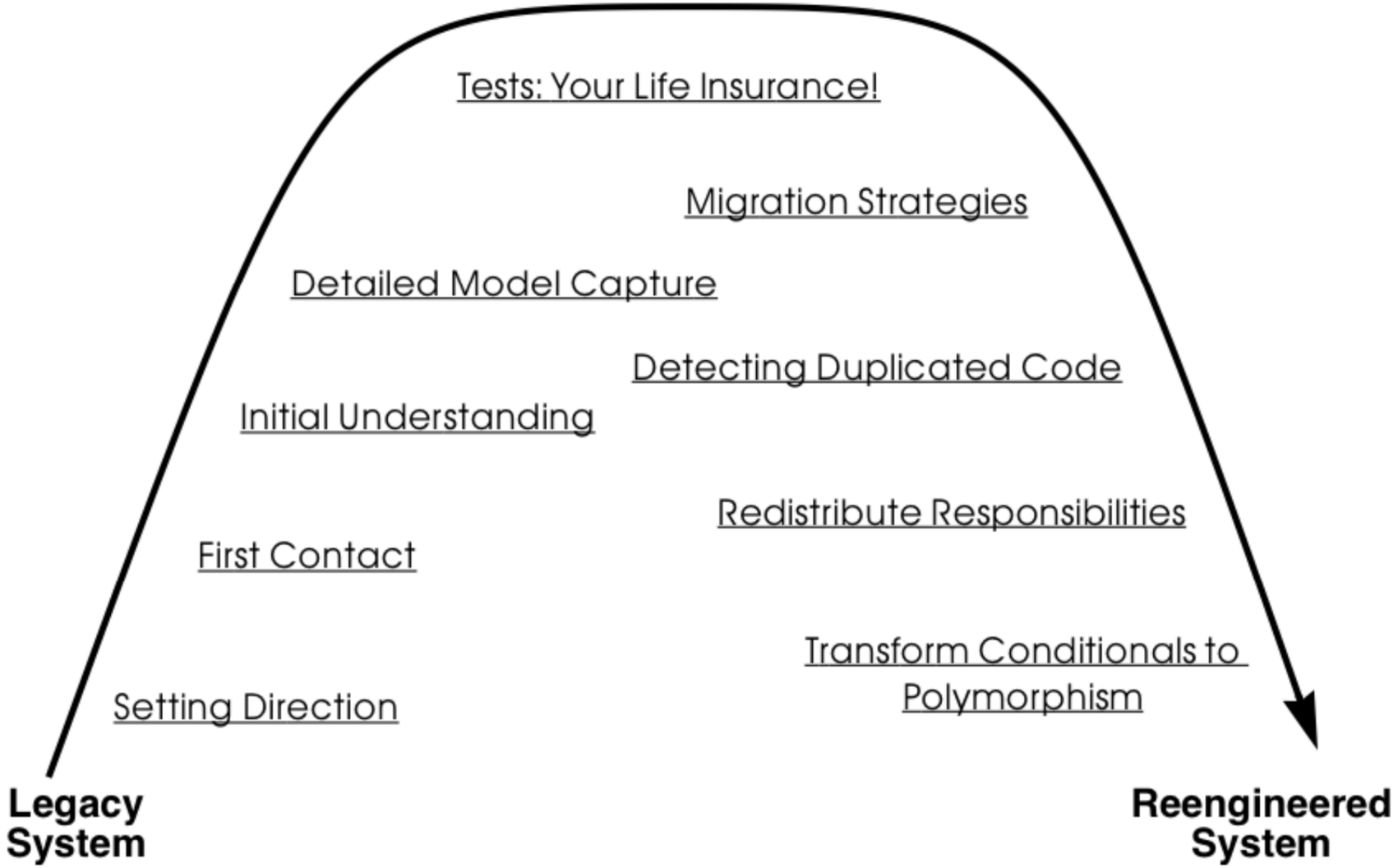
OORP

http://scg.unibe.ch/download/oorp/



- Vorgehensmodell & 49 Muster
  - Reverse Engineering & Reengineering
- Inhalte entstanden um 2000
  - Autoren Serge Demeyer, Stéphane Ducasse, Oscar Nierstrasz
  - Vorworte von Michael Fowler und Ralph E. Johnson
  - Empfehlungen von Frank Buschmann, Kent Beck und Linda Rising
- Buch kostenfrei im Internet
  - Leider älter und nicht mehr weitergepflegt
  - Immer noch sehr guter Einstieg

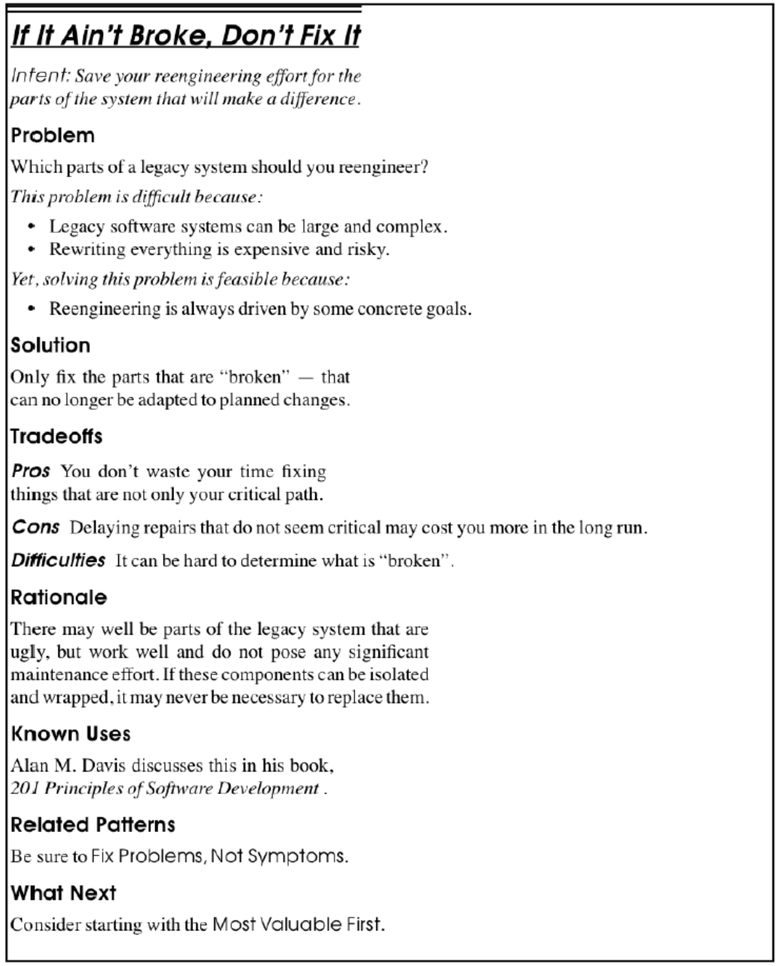
OORP-Übersicht



Aufbau der OORP-Muster

<Name des Musters>

- Also Known As
- Intent
- Problem
- Solution
  - Inkl. der relevanten Schritte
- Tradeoffs
  - Pros
  - Cons
- Difficulties
- Rationale
- Related Patterns



If It Ain't Broke, Don't Fix It

Intent: Save your reengineering effort for the parts of the system that will make a difference.

Problem

Which parts of a legacy system should you reengineer?

This problem is difficult because:

- Legacy software systems can be large and complex.
- Rewriting everything is expensive and risky.

Yet, solving this problem is feasible because:

- Reengineering is always driven by some concrete goals.

Solution

Only fix the parts that are "broken" — that can no longer be adapted to planned changes.

Tradeoffs

Pros You don't waste your time fixing things that are not only your critical path.

Cons Delaying repairs that do not seem critical may cost you more in the long run.

Difficulties It can be hard to determine what is "broken".

Rationale

There may well be parts of the legacy system that are ugly, but work well and do not pose any significant maintenance effort. If these components can be isolated and wrapped, it may never be necessary to replace them.

Known Uses

Alan M. Davis discusses this in his book, *201 Principles of Software Development*.

Related Patterns

Be sure to Fix Problems, Not Symptoms.

What Next

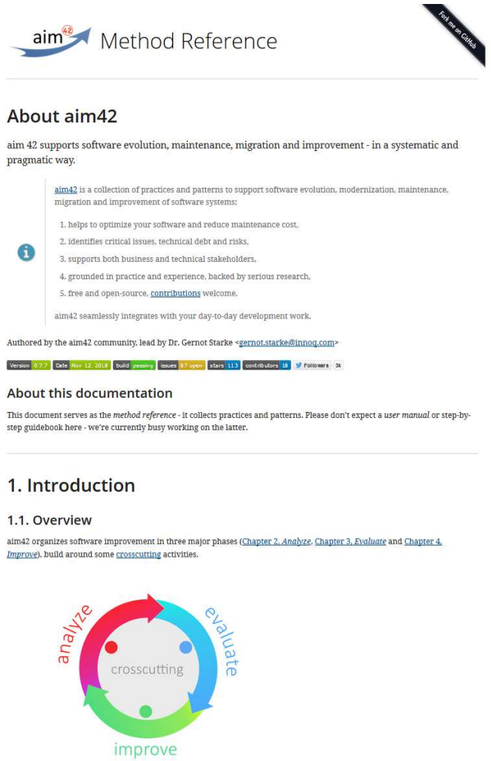
Consider starting with the Most Valuable First.

aim42

aim42

https://aim42.github.io/

- Besteht aus jahrelang gesammelten Projekterfahrungen
  - + wissenschaftlichen Belegen
- Entsteht seit 2014
  - Initiiert von Gernot Starke
- Wird aktiv weitergepflegt
  - Bisher 67 Praktiken niedergeschrieben
- Open-Source auf GitHub
  - Jede(r) kann mitmachen



aim42 Method Reference

Fork me on GitHub

About aim42

aim 42 supports software evolution, maintenance, migration and improvement - in a systematic and pragmatic way.

aim42 is a collection of practices and patterns to support software evolution, modernization, maintenance, migration and improvement of software systems:

1. helps to optimize your software and reduce maintenance cost,
2. identifies critical issues, technical debt and risks,
3. supports both business and technical stakeholders,
4. grounded in practice and experience, backed by serious research,
5. free and open-source, contributions welcome.

aim42 seamlessly integrates with your day-to-day development work.

Authored by the aim42 community, lead by Dr. Gernot Starke <gernot.starke@innoq.com>

About this documentation

This document serves as the *method reference* - it collects practices and patterns. Please don't expect a *user manual* or step-by-step guidebook here - we're currently busy working on the latter.

1. Introduction

1.1. Overview

aim42 organizes software improvement in three major phases (Chapter 2, *Analyze*, Chapter 3, *Evaluate* and Chapter 4, *Improve*), build around some crosscutting activities.

aim42-Vorgehen

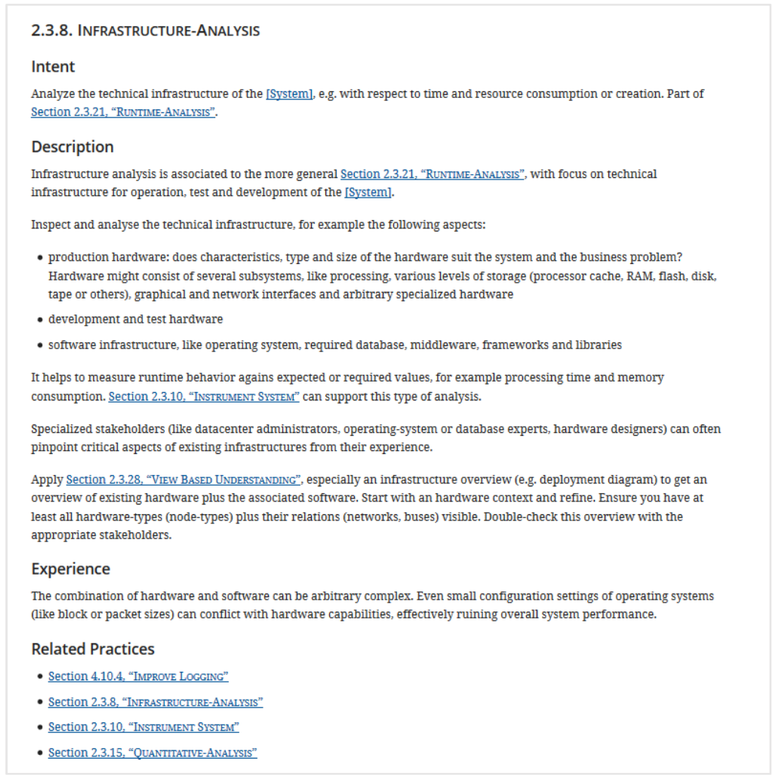
Architecture Improvement Method

https://aim42.github.io/

Aufbau der aim42-Muster

<Name der Praktik>

- Description
- Experiences
- Applicability
- Related Practices
- References



2.3.8. Infrastructure-Analysis

Intent

Analyze the technical infrastructure of the [System], e.g. with respect to time and resource consumption or creation. Part of Section 2.3.21, "Runtime-Analysis".

Description

Infrastructure analysis is associated to the more general Section 2.3.21, "Runtime-Analysis", with focus on technical infrastructure for operation, test and development of the [System].

Inspect and analyse the technical infrastructure, for example the following aspects:

- production hardware: does characteristics, type and size of the hardware suit the system and the business problem? Hardware might consist of several subsystems, like processing, various levels of storage (processor cache, RAM, flash, disk, tape or others), graphical and network interfaces and arbitrary specialized hardware
- development and test hardware
- software infrastructure, like operating system, required database, middleware, frameworks and libraries

It helps to measure runtime behavior agains expected or required values, for example processing time and memory consumption. Section 2.3.10, "Instrument System" can support this type of analysis.

Specialized stakeholders (like datacenter administrators, operating-system or database experts, hardware designers) can often pinpoint critical aspects of existing infrastructures from their experience.

Apply Section 2.3.28, "View Based Understanding", especially an infrastructure overview (e.g. deployment diagram) to get an overview of existing hardware plus the associated software. Start with an hardware context and refine. Ensure you have at least all hardware-types (node-types) plus their relations (networks, buses) visible. Double-check this overview with the appropriate stakeholders.

Experience

The combination of hardware and software can be arbitrary complex. Even small configuration settings of operating systems (like block or packet sizes) can conflict with hardware capabilities, effectively ruining overall system performance.

Related Practices

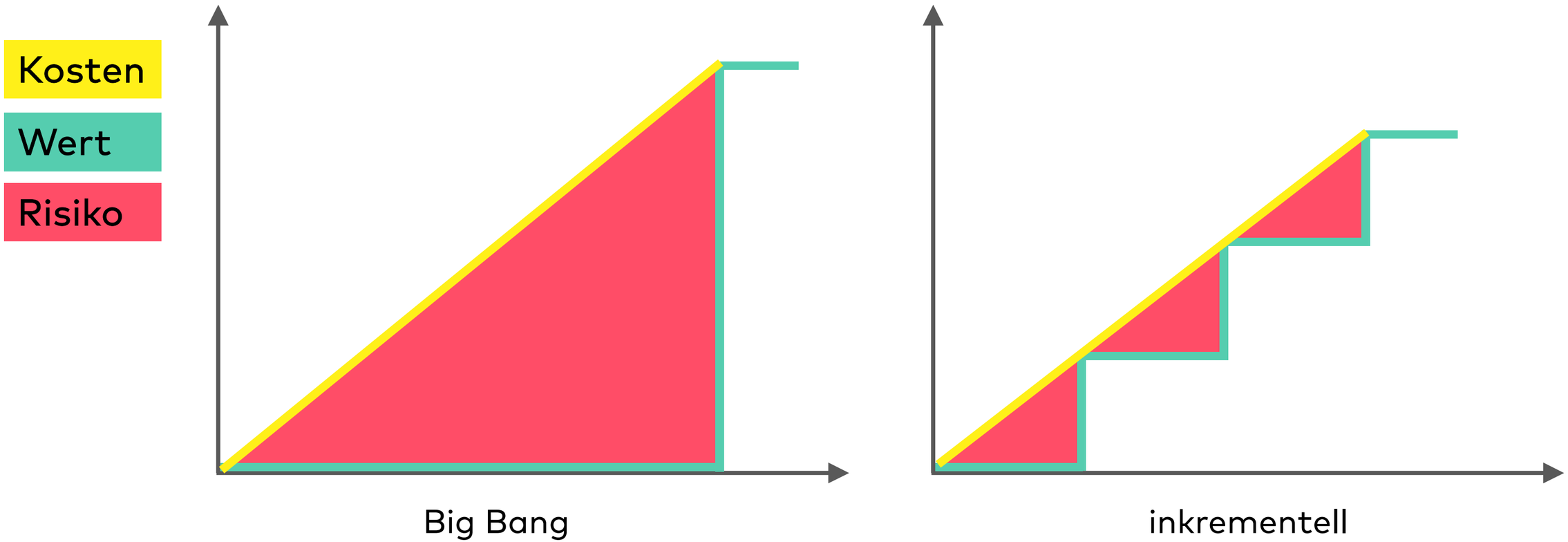
- Section 4.10.4, "Improve Logging"
- Section 2.3.8, "Infrastructure-Analysis"
- Section 2.3.10, "Instrument System"
- Section 2.3.15, "Quantitative-Analysis"

Systematisches Verbessern

am Beispiel von aim42

Inkrementell statt Big Bang

- Weniger riskant, aber anstrengender



Idee von https://blog.crisp.se/wp-content/uploads/2013/08/20130820-What-is-Agile.pdf

Moderne Werkzeuge

zur Unterstützung von Modernisierungsvorhaben

Auswahlkriterien

- Open-Source (kostenlos)
- Praxistauglich

- Java-lastig
- Analyse-orientiert

Jäger des verlorenen Schatzes

Data-Science-Werkzeugkasten

Muster und Praktiken

OORP

- Learn from the Past

aim42

- Bus Factor
- Software Archeology

Softwarearchäologie?

„Archäologen versuchen, die Hinterlassenschaften der Leute zu finden, die vor uns waren, und den Sinn dahinter zu verstehen.“

übersetzt aus „Archaeology at work“,
English Heritage Education Service

Live-Demo

Einfache Historienanalyse mit Git

- bash
- Git

Search for commit activity per author

Command

```
git shortlog -ns -- **Owner**.java
```

Example output

```
    45  michaelisvy
    12  Antoine Rey
     6  Keith Donald
     2  Tomas Repel
     1  Colin But
```

https://gist.github.com/feststelltaste/044799f484847e7d31d43999b86e2642

Live-Demo

Verlorenes Wissen im Linux-Kernel

- Git
- Jupyter Notebook
- Python
- pandas
- matplotlib

https://www.feststelltaste.de/identifying-lost-knowledge-in-the-linux-kernel-source-code/

Live-Demo (Daten)

```
static void rb532_mask_and_ack_irq(struct irq_data *d)
{
  rb532_disable_irq(d);
  ack_local_irq(group_to_ip(irq_to_group(d->irq)));
}

static int rb532_set_type(struct irq_data *d, unsigned type)
{
  int gpio = d->irq - GPIO_MAPPED_IRQ_BASE;
  int group = irq_to_group(d->irq);

  if (group != GPIO_MAPPED_IRQ_GROUP)
```

Quellcode

Live-Demo (Daten)

```
164) static void rb532_mask_and_ack_irq(struct irq_data *d)
165) {
166)    rb532_disable_irq(d);
167)    ack_local_irq(group_to_ip(irq_to_group(d->irq)));
168) }
169)
170) static int rb532_set_type(struct irq_data *d, unsigned type)
171) {
172)    int gpio = d->irq - GPIO_MAPPED_IRQ_BASE;
173)    int group = irq_to_group(d->irq);
174)
175)    if (group != GPIO_MAPPED_IRQ_GROUP)
```

Änderung pro Zeile

Live-Demo (Daten)

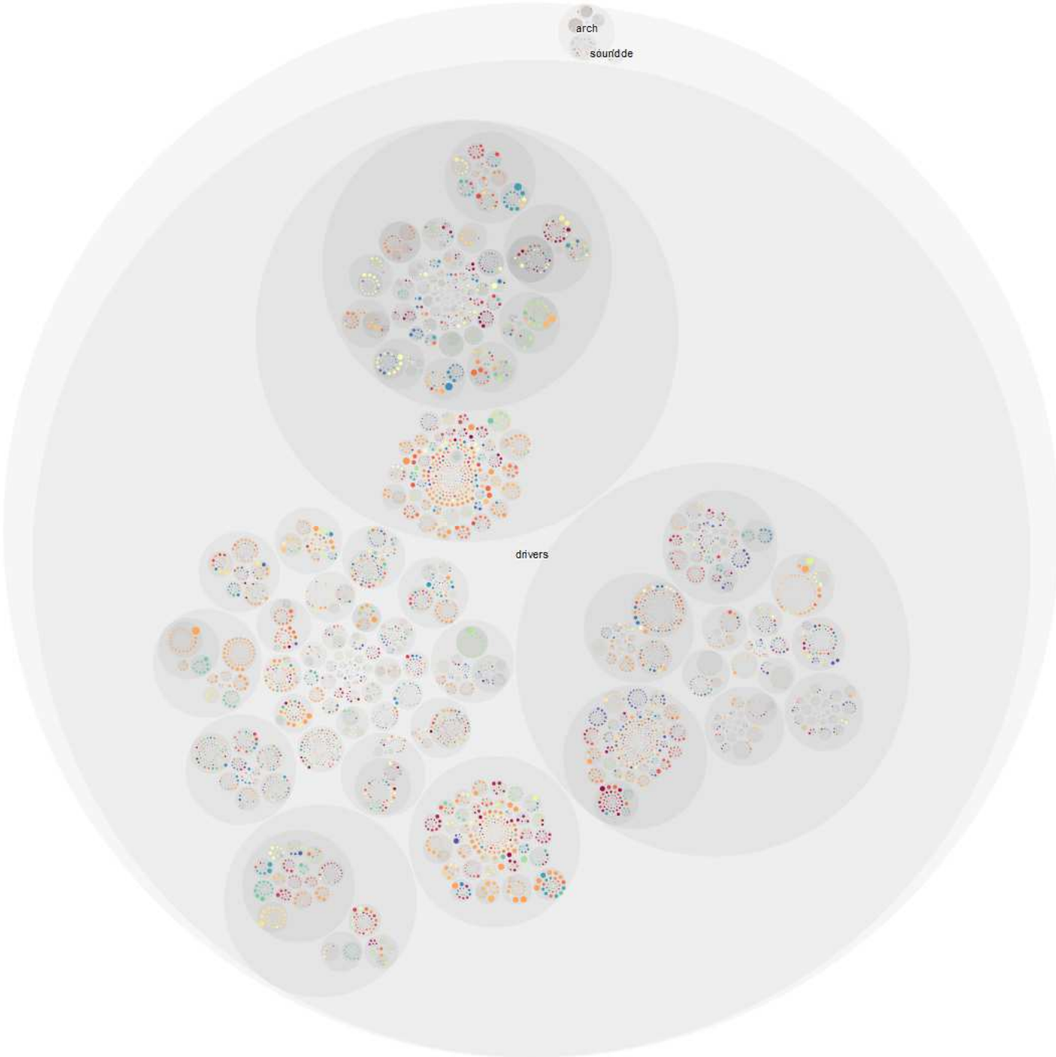
```
efb02d arch/irq.c (Thomas Gleixner  2011-03-23 21:09:10 +0000 164) s
73b439 arch/irq.c (Ralf Baechle     2008-07-16 16:12:25 +0100 165) {
efb02d arch/irq.c (Thomas Gleixner  2011-03-23 21:09:10 +0000 166)
efb02d arch/irq.c (Thomas Gleixner  2011-03-23 21:09:10 +0000 167)
73b439 arch/irq.c (Ralf Baechle     2008-07-16 16:12:25 +0100 168) }
73b439 arch/irq.c (Ralf Baechle     2008-07-16 16:12:25 +0100 169)
efb02d arch/irq.c (Thomas Gleixner  2011-03-23 21:09:10 +0000 170) s
4aa0f4 arch/irq.c (Phil Sutter      2008-11-28 20:45:10 +0100 171) {
efb02d arch/irq.c (Thomas Gleixner  2011-03-23 21:09:10 +0000 172)
efb02d arch/irq.c (Thomas Gleixner  2011-03-23 21:09:10 +0000 173)
4aa0f4 arch/irq.c (Phil Sutter      2008-11-28 20:45:10 +0100 174)
efb02d arch/irq.c (Thomas Gleixner  2011-03-23 21:09:10 +0000 175)
```

Letzter Entwickler pro Zeile

Live-Demo

Wissensinseln finden

- Git
- Jupyter Notebook
- Python
- pandas
- D3



https://www.feststelltaste.de/knowledge-islands/

Der Herr der Dinge

Integrierte Anwendungs-Dashboards

Muster und Praktiken

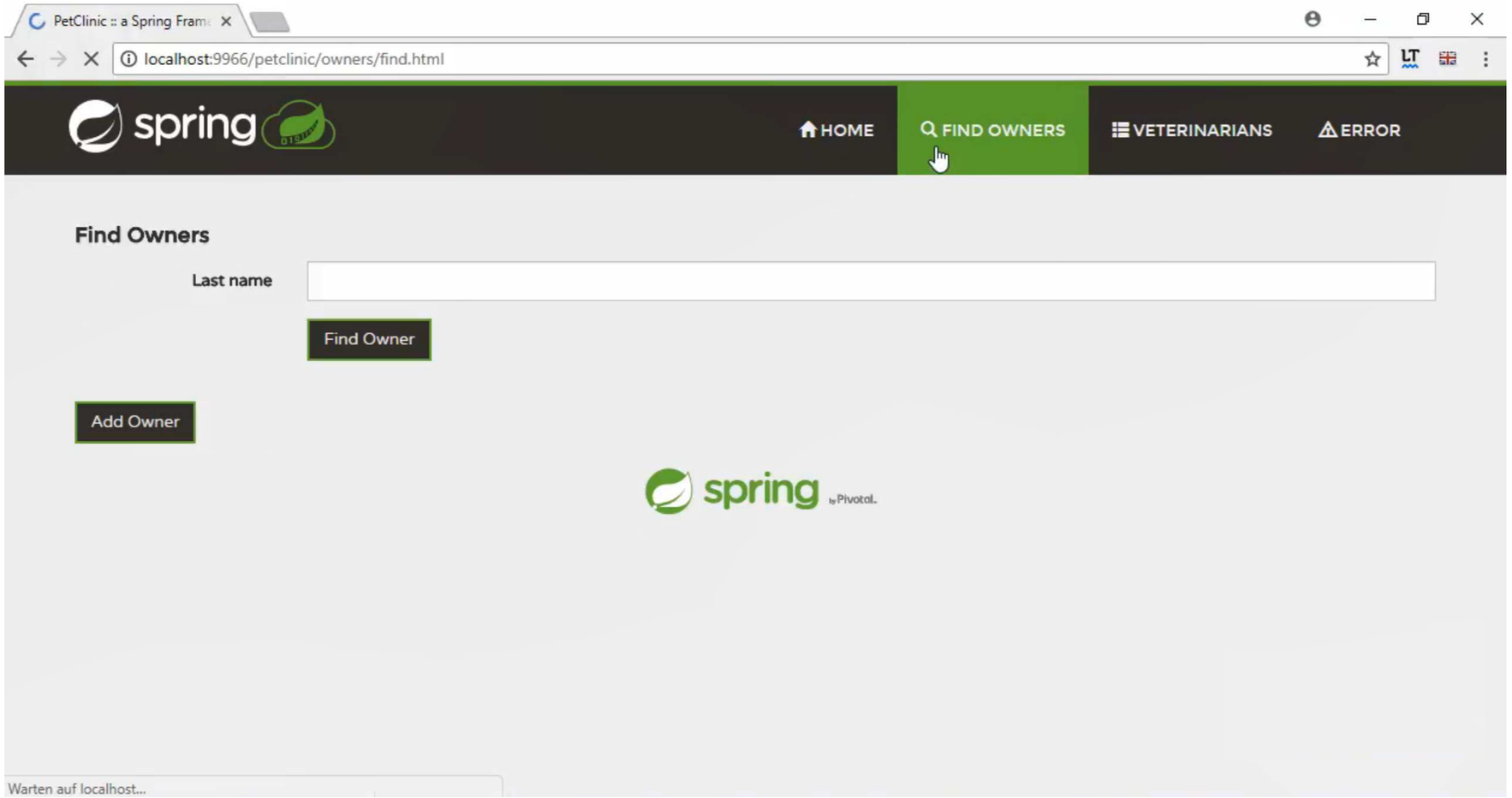
OORP

- Build Confidence
- Involve the Users

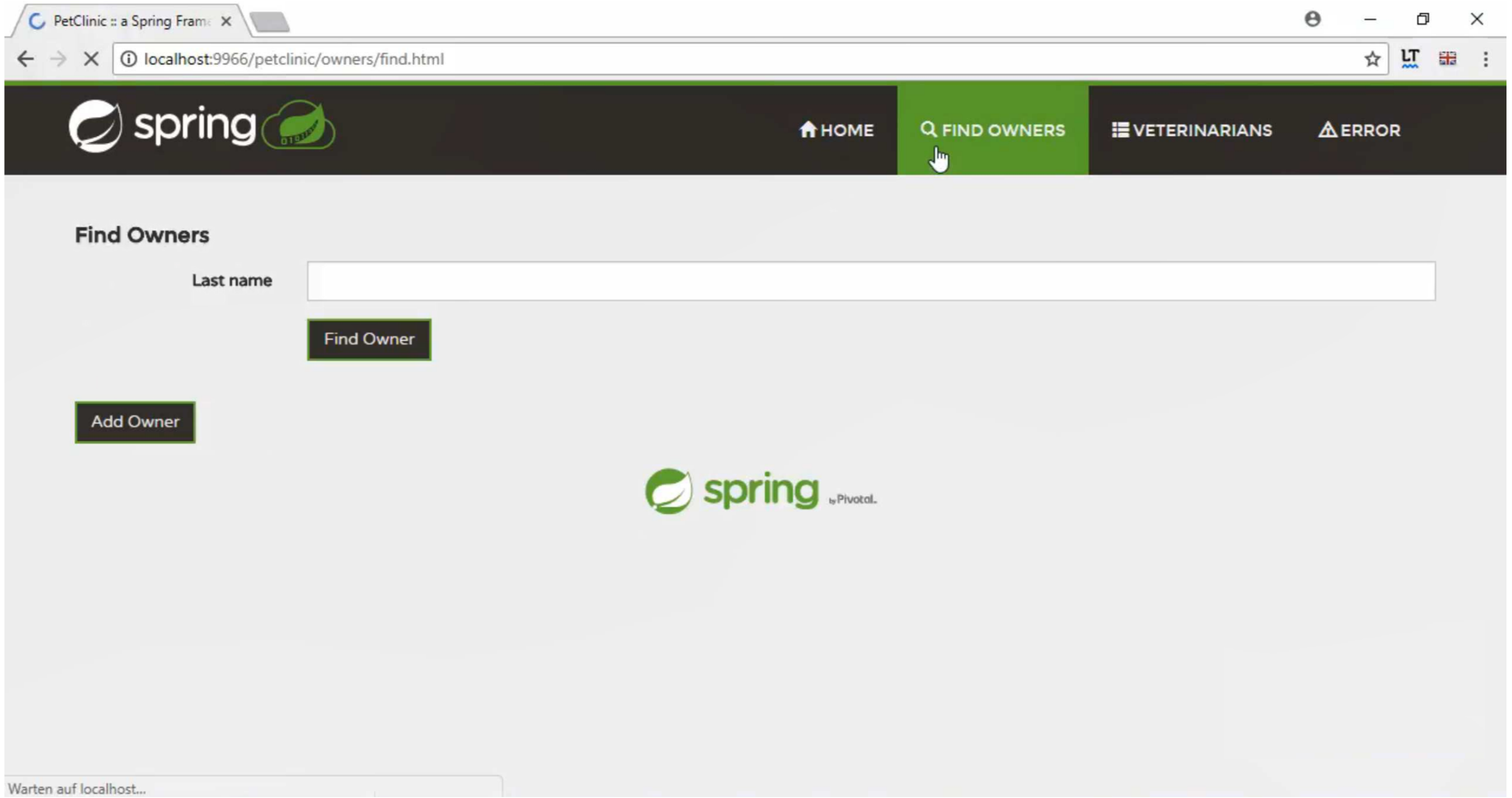
aim42

- Fast Feedback
- Expect Denial

Die Software vor dem Reengineering



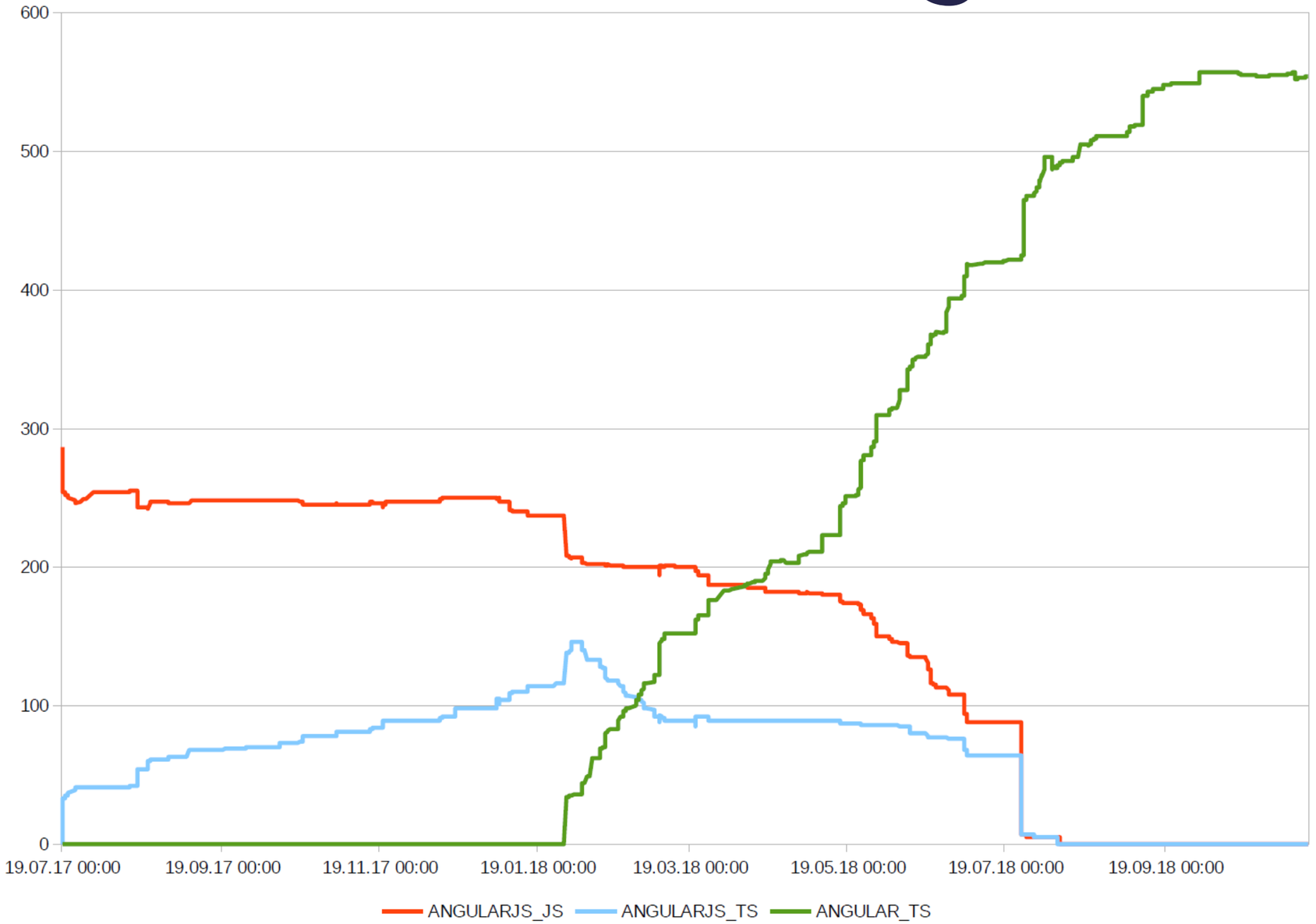
Die Software nach dem Reengineering



Technische Arbeiten offenlegen

Beispiel:

Technologie-austausch



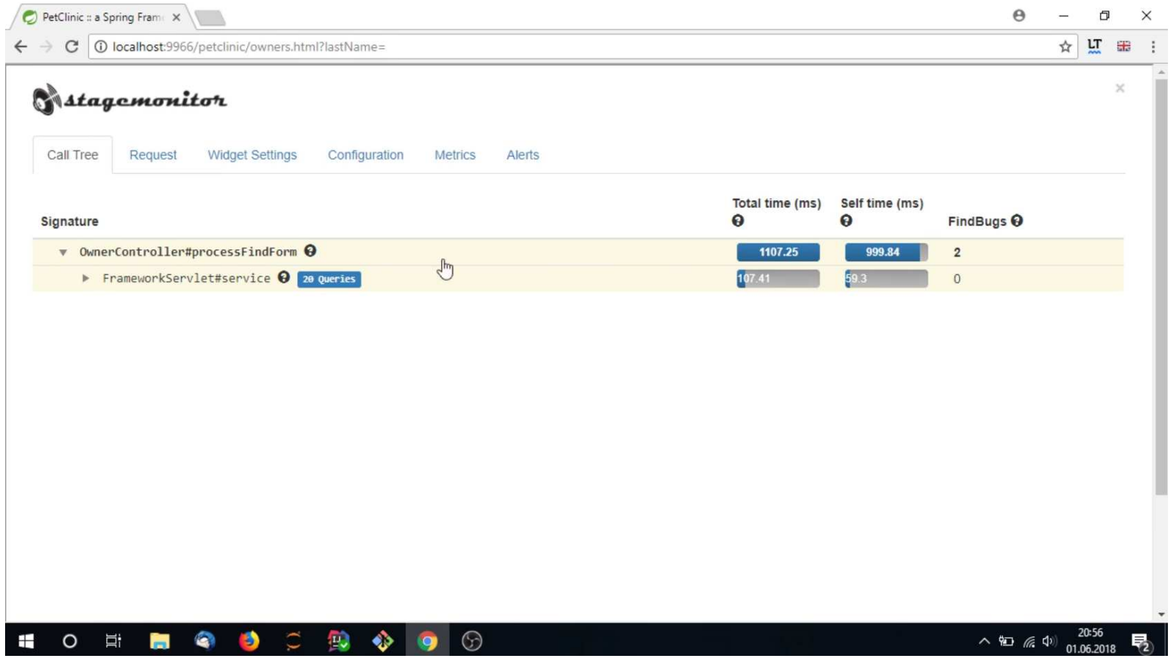
Y: Dateianzahl

X: Zeit

Demo: stagemonitor

Technische Schulden

- in der Anwendung über Application Performance Monitoring sichtbar machen



Den Wert der Arbeit visualisieren

Systemüberwachung

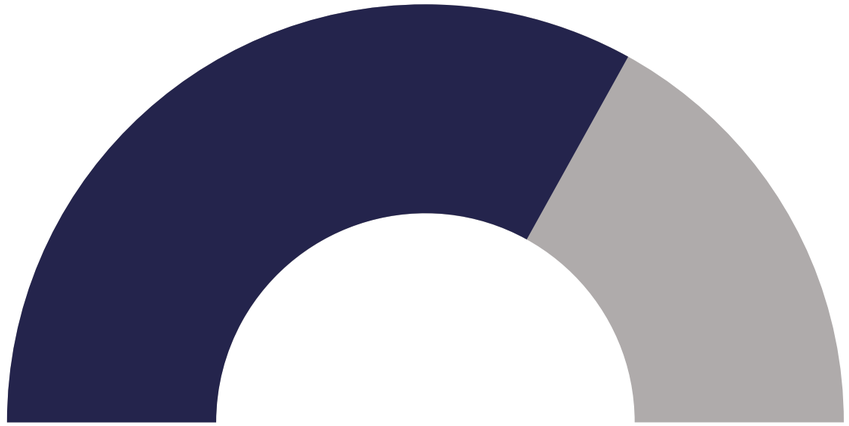
Tage seit letztem Ausfall

143

Ticketsystem

Erledigte Stories in diesem Sprint

4/15



Aktive User

14

Fertigstellung DB-Zugriff

78%

Heutige Bestellungen

41

Ø Bestellungen

54

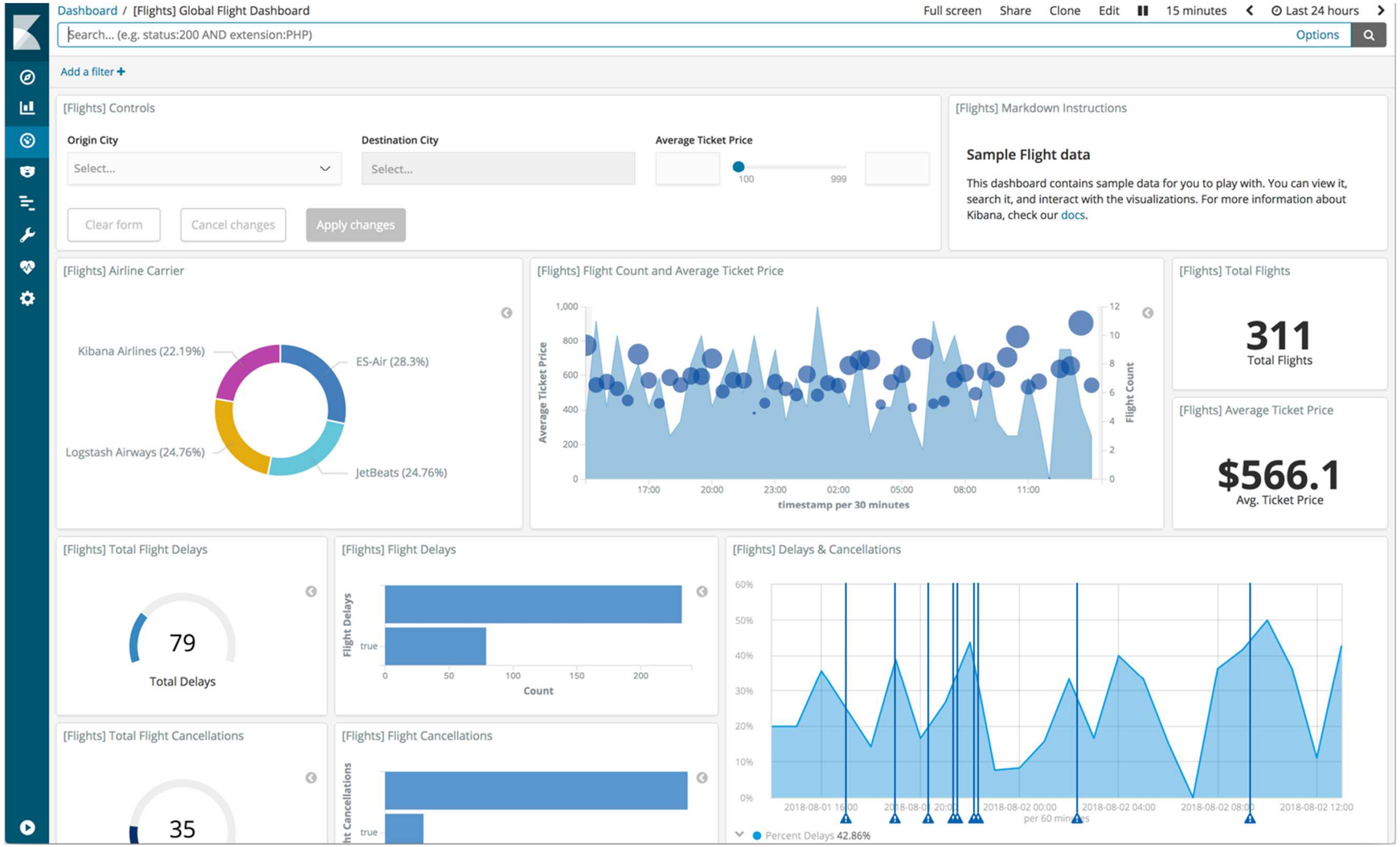
Log-Dateien

Versionskontrolle

Business Monitoring

Data Warehouse

Beispiel: Kibana



Quelle: https://www.elastic.co/guide/en/kibana/current/tutorial-sample-data.html

Zurück in die Zukunft

Code-Suchmaschinen

Muster und Praktiken

OORP

- Look for the Contracts
- Prototype to Target Solution

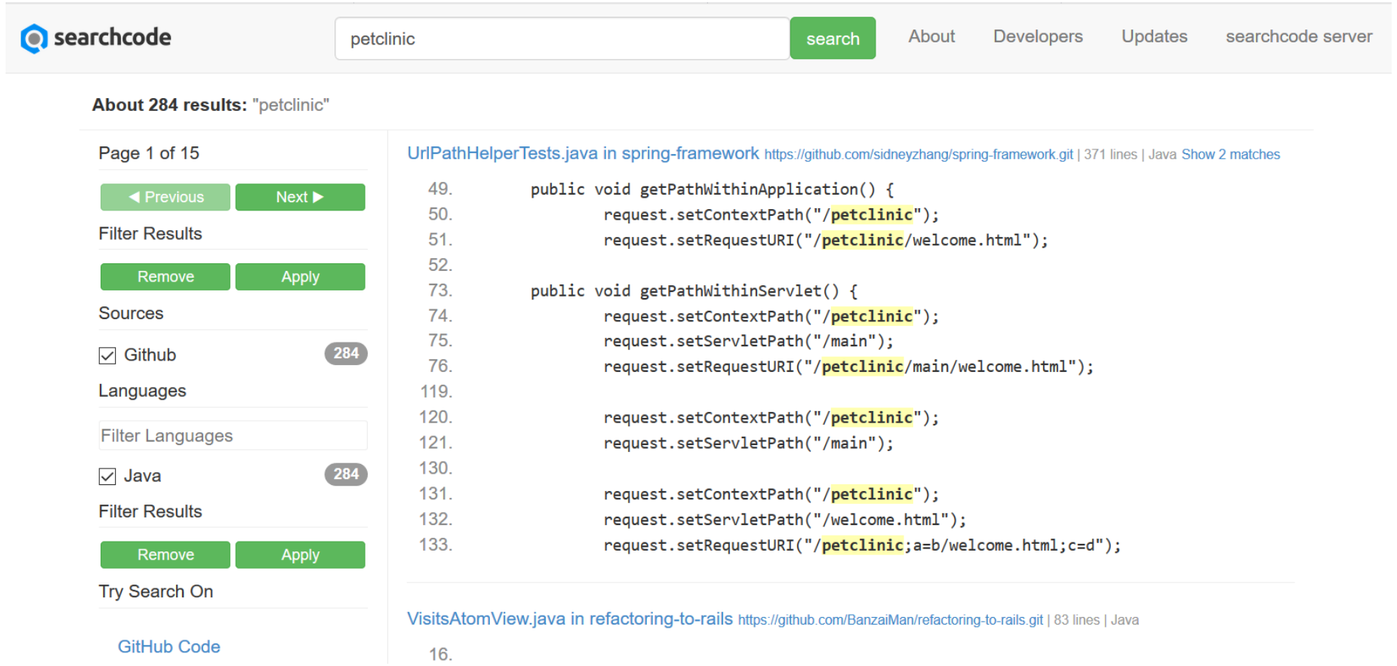
aim42

- Documentation-Analysis
- Static-Code Analysis
- Estimate Feature Value
- Estimate Improvement Cost
- Estimate Issue Cost

Live-Demo

Übersicht bekommen

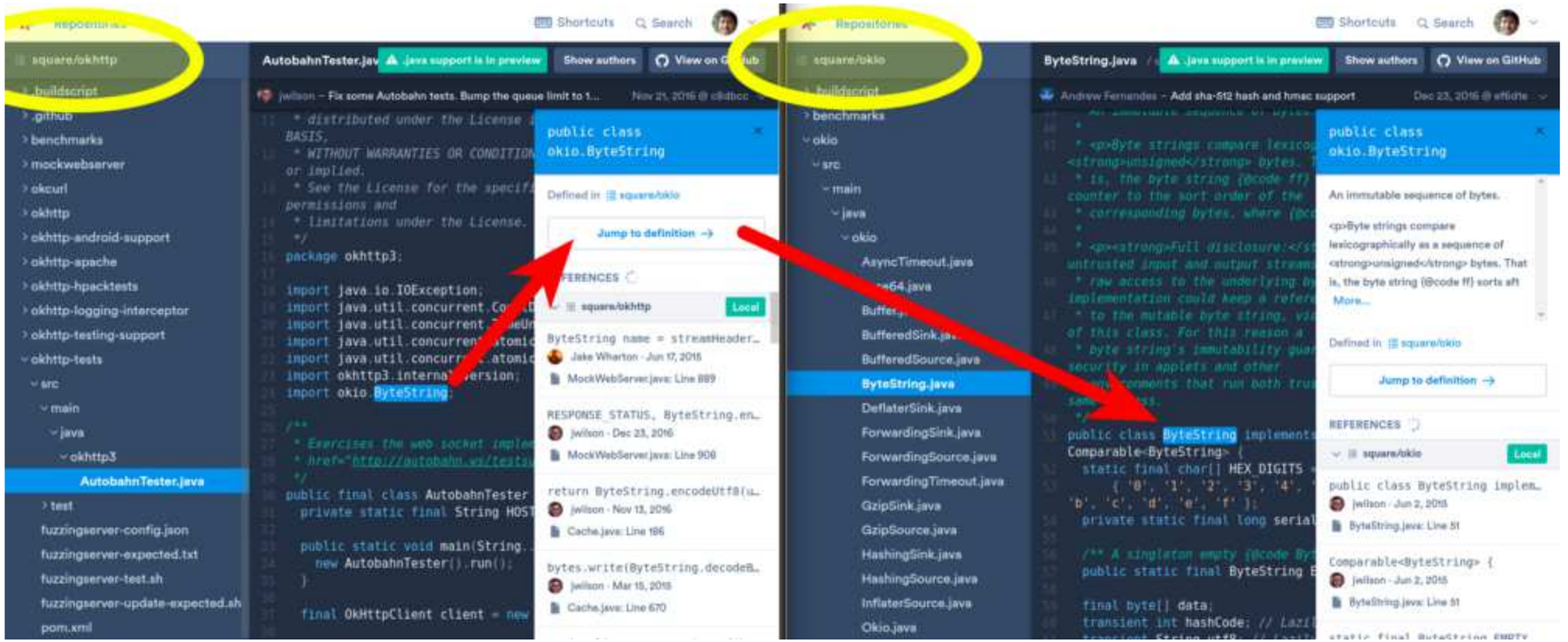
- searchcode



~~Live~~-Demo

Impact-Analysen mit Cross-Reference-Suche*

- sourcegraph



*Seit November 2018 / Version 2.12.0 Cross-Reference-Funktion kostenpflichtig

https://about.sourcegraph.com/blog/code-intelligence-now-available-for-java-on-sourcegraph-com

Code Intelligence / Software Analytics

Idee

Analysen auf Daten aus der Entwicklung oder dem Betrieb von Software

Zielgruppe

Softwareentwickler und Management

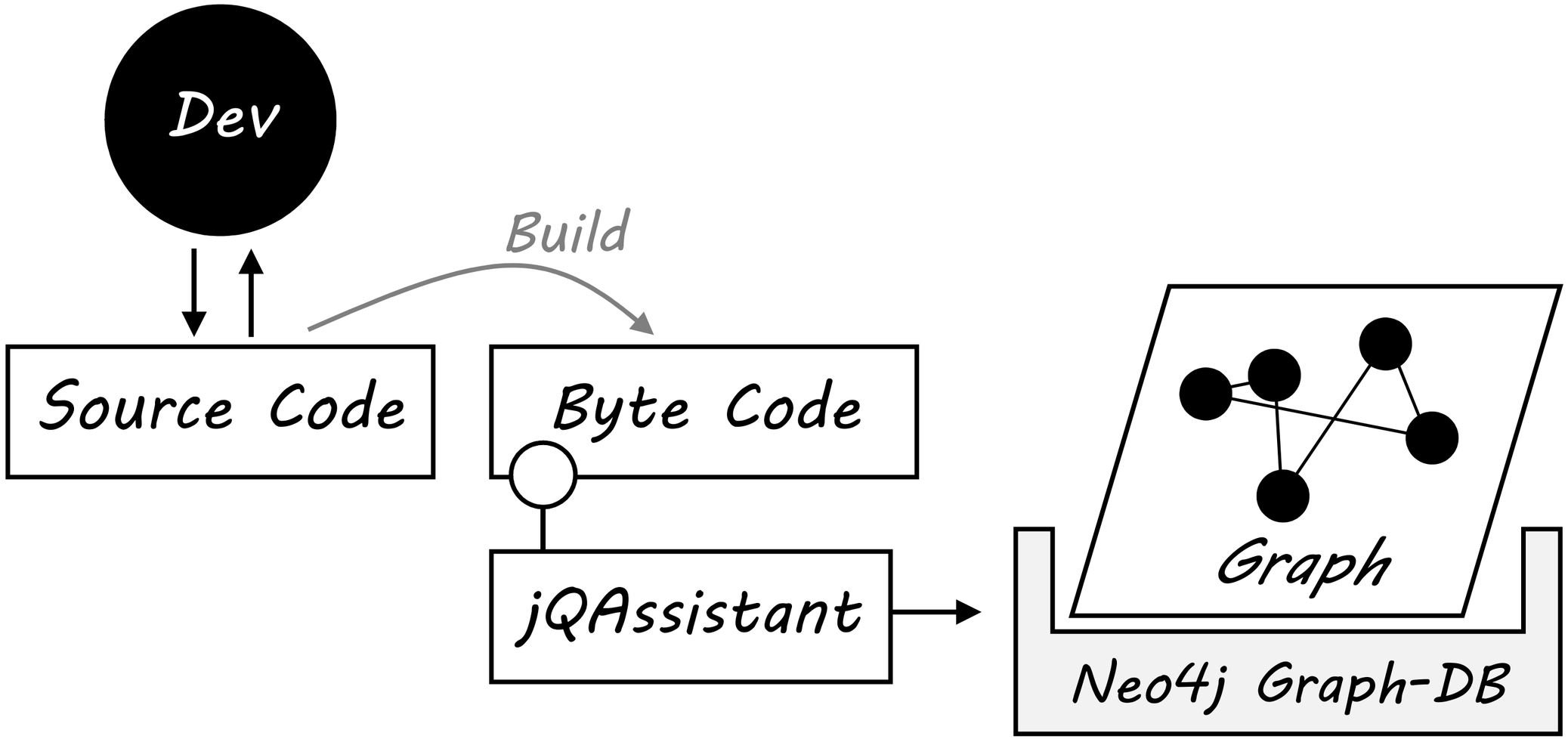
Ziel

Neue Einsichten aus den Softwaredaten gewinnen

Zweck

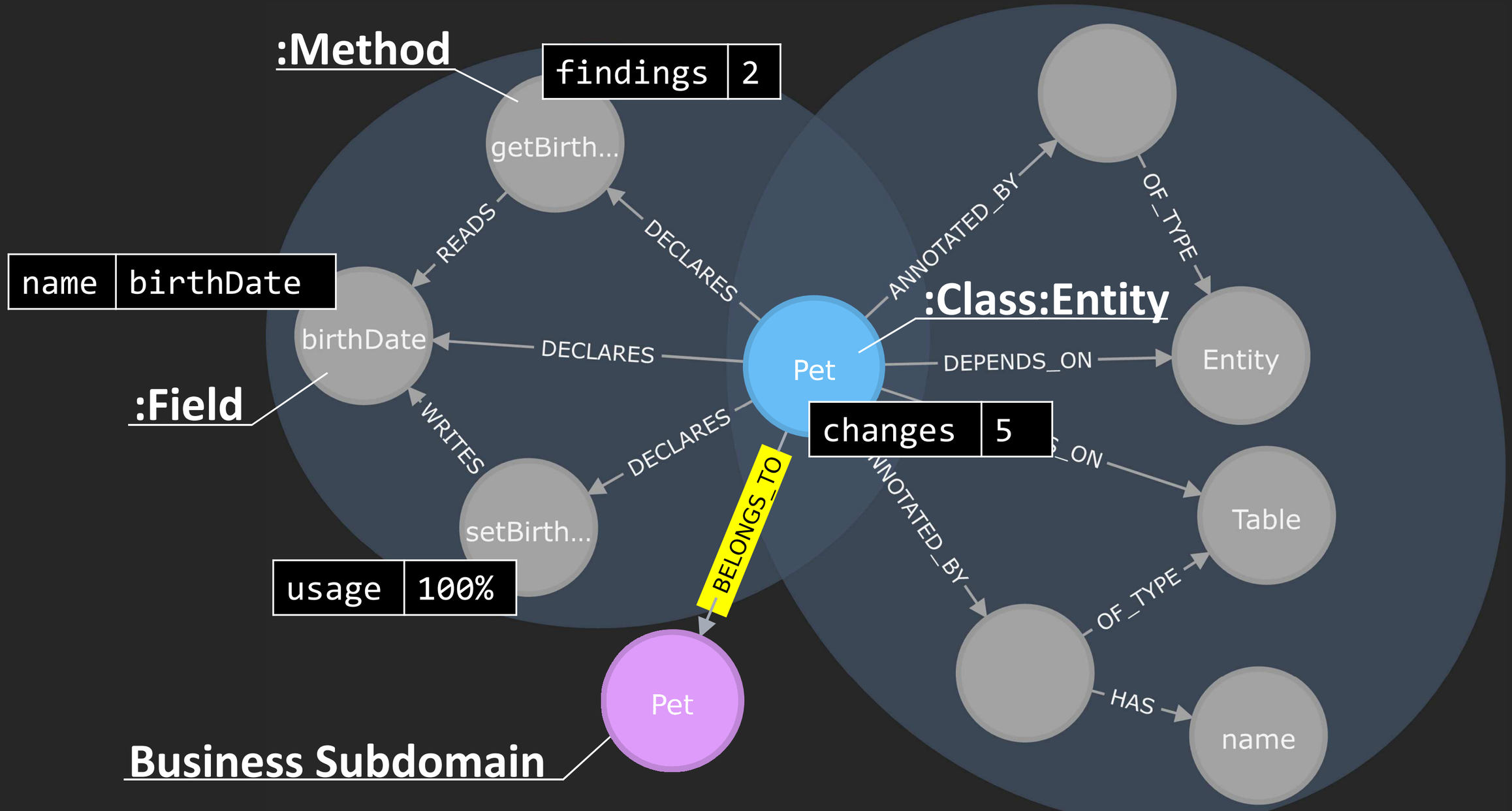
Bessere Entscheidungen treffen

Software Analytics mit Graphen

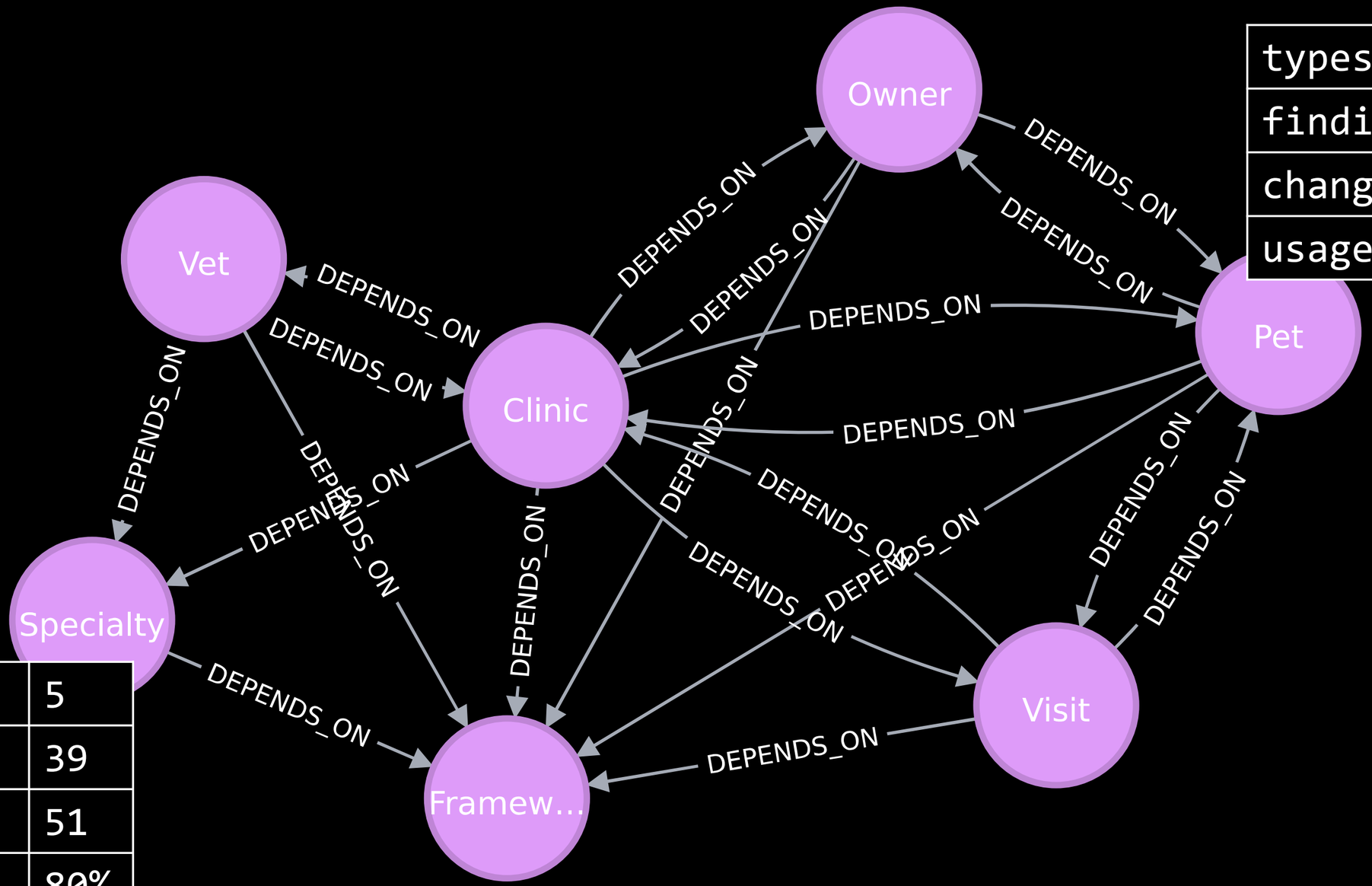


https://git.io/f49KO

Softwarelandschaft als Graph



Softwarelandschaft als Graph



types	16
findings	17
changes	15
usage	70%

types	5
findings	39
changes	51
usage	80%

Live-Demo

Reengineerings simulieren

- jQAssistant
- Neo4j

Die zehn Gebote

Selbstvalidierende Architekturdokumentation

Muster und Praktiken

OORP

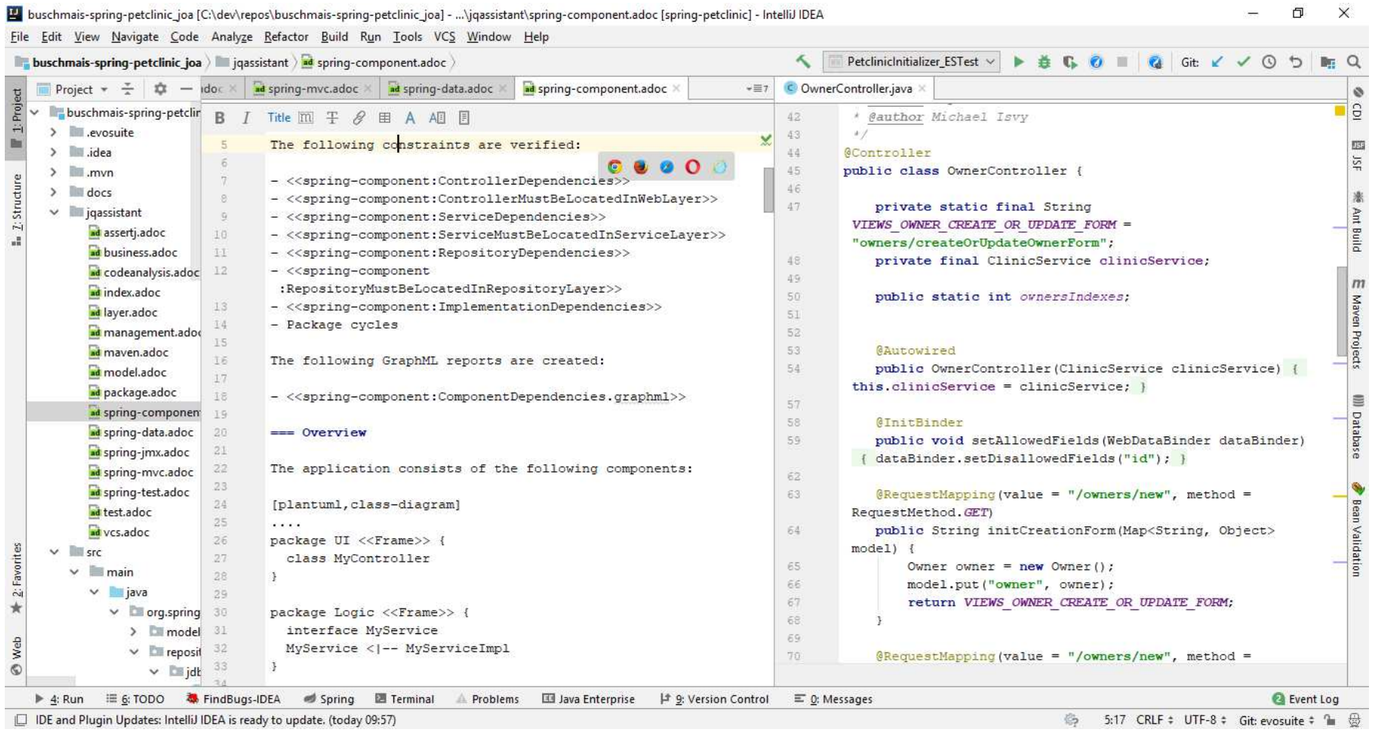
- Speculate about Design
- Record Business Rules as Tests
- Tie Code and Questions

aim42

- Docs-As-Code

Dokumentation als Quellcode

- ✓Versionierbar
- ✓Vergleichbar/Review-fähig
- ✓Branch-fähig
- ✓Revisionssicher
- ✓Auslieferbar
- ✓Kontrollierbar

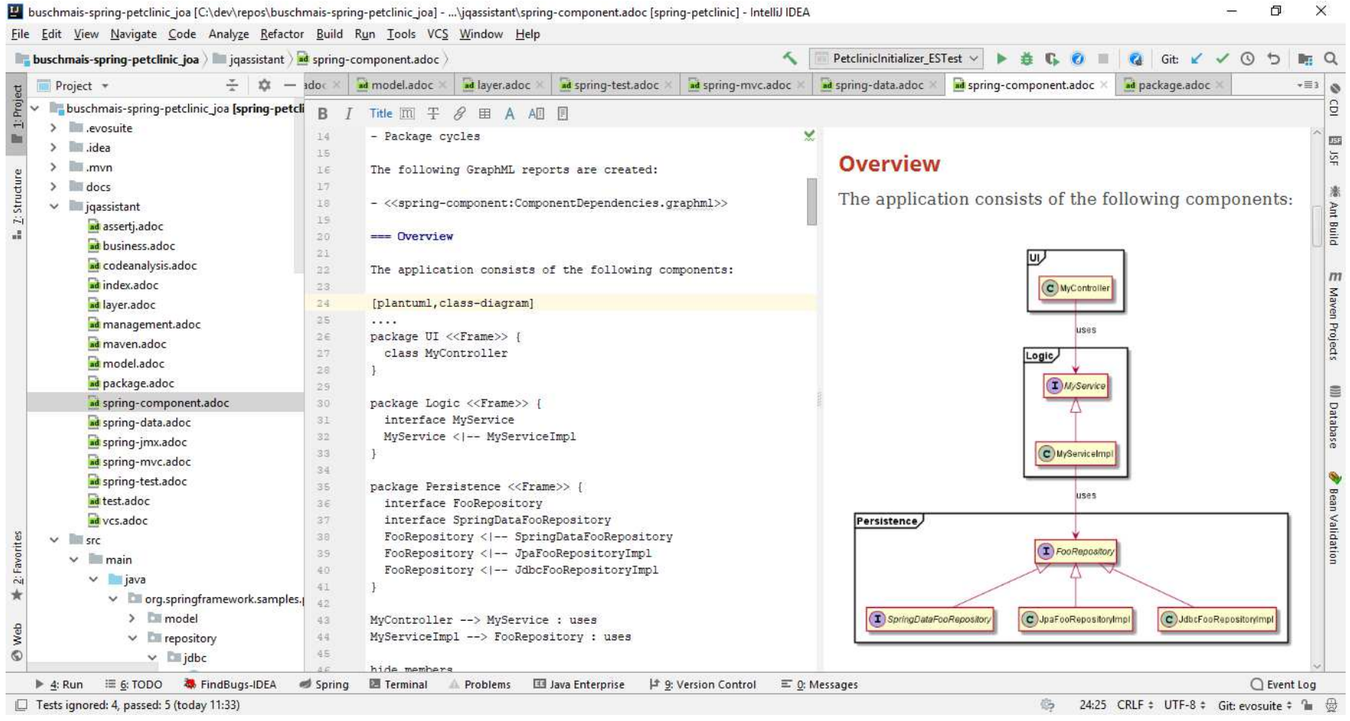


=> **Gleicher Stellenwert**

Der Docs-As-Code-Ansatz

Dokumentation in der Entwicklungsumgebung

- Kein Medienbruch / Werkzeugwechsel
- Direktes Verlinken (Code⇔Doku)
- Modularisierung
- Integriert in Build
- Testbarkeit



Live-Demo

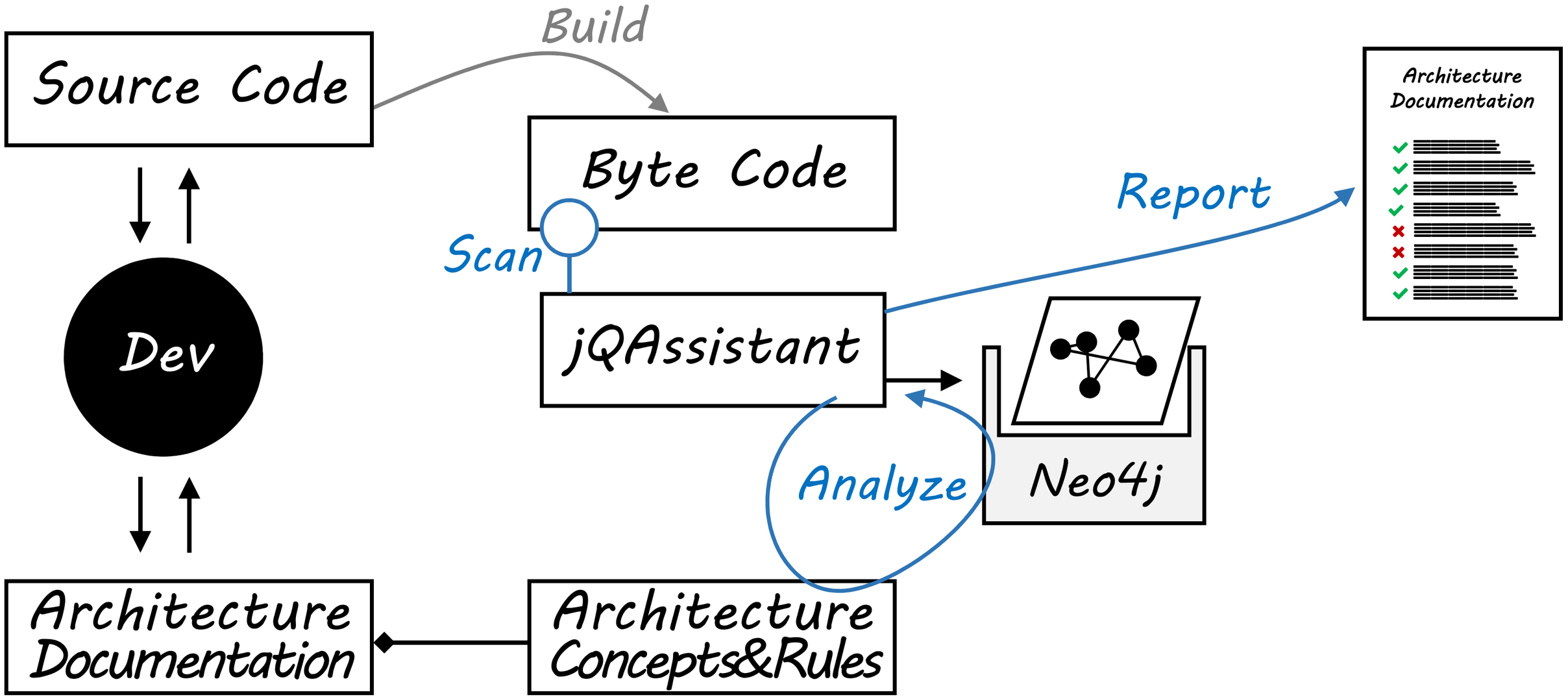
Docs-As-Code

- AsciiDoc
- Asciidoctor

Selbstvalidierende Dokumentation

- Dokumentation beschreibt Architektur als Text
- Wichtige Muster und Konzepte können abstrakt formuliert werden
- Regelvalidierungen können Abweichungen feststellen
- Konzepte und Regelabweichungen werden in Dokumentation angezeigt
- Überprüfung erfolgt automatisiert

Aufbau und Abläufe



Live-Demo

Selbstvalidierende (Re-)Dokumentation

- AsciiDoc
- Asciidoctor
- jQAssistant
- Neo4j

Zusammenfassung

Zusammenfassung

- OORP und aim42 bieten praktische Vorgehen
- Moderne Werkzeuge sind vorhanden
  - Analysen effektiv möglich
  - Reengineering auch im Großen möglich
  - Sichtbarkeit technischer Arbeiten möglich
  - Dokumentation wird frustfrei und (teil-)automatisiert

=> Legacy Code ist ein **spannendes Thema!**

Danke!

INNOQ
www.innoq.com

Markus Harrer
markus.harrer@innoq.com

+49 175 5753640
@feststelltaste
https://feststelltaste.de

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin
Germany

Ludwigstr. 180E
63067 Offenbach
Germany

Kreuzstr. 16
80331 München
Germany

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
+41 41 743 01 11

Albulastr. 55
8048 Zürich
Switzerland

Anhang: Empfehlungen

Empfehlungen

Literatur

- Michael Feathers: Effektives Arbeiten mit Legacy Code
- Adam Tornhill: Software Design X-Ray
- Christian Bird, Tim Menzies, Thomas Zimmermann:
  The Art and Science of Analyzing Software Data
- Tim Menzies, Laurie Williams, Thomas Zimmermann:
  Perspectives on Data Science for Software Engineering
- Wes McKinney: Python For Data Analysis

Software

- Python Data Science Distribution: https://anaconda.com
- jQAssistant: https://github.com/JavaOnAutobahn/spring-petclinic
- GitHub-Repo: https://github.com/feststelltaste/software-analytics
- Tutorial: https://feststelltaste.de/mini-tutorial-git-log-analyse-mit-python-und-pandas

Empfehlungen

Internet

- Architecture Improvement Method aim42 (Website): https://aim42.org
- Architecture Improvement Method aim42 (Guide): http://aim42.github.io
- Object-Oriented Reengineering Patterns OORP (PDF): http://scg.unibe.ch/download/oorp/
- Legacycode.rocks (Website, Podcast und mehr): https://legacycode.rocks
- Dylan Beattie (Video): http://videos.ncrafts.io/video/275529979
- Adam Tornhill (Video): https://www.youtube.com/watch?v=SdUewLCHWvU

- Mein Blog: https://feststelltaste.de
- Mein GitHub-Repository: https://github.com/feststelltaste

INNOQ

"Tools only find, people have to find out!"

Markus Harrer

Software Development Analyst

bei innoQ Deutschland GmbH

- Datenanalysen in der Softwareentwicklung
- Architektur-, Design- und Code-Reviews
- Reverse- und Re-Engineering von Legacy-Code

www.innoq.com

SERVICES

Strategy & technology consulting
Digital business models
Software architecture & development
Digital platforms & infrastructures
Knowledge transfer, coaching & trainings

FACTS

~125 employees
Privately owned
Vendor-independent

OFFICES

Monheim
Berlin
Offenbach
Munich
Zurich

CLIENTS

Finance
Telecommunications
Logistics
E-Commerce
Fortune 500
SMBs
Startups