

**ICH GLAUB ES HAT SCHNUPFEN**

**DER HEALTHCHECK FÜR SOFTWAREPROJEKTE**



**Saxonia Systems**  
So geht Software.

# Der Sprecher



## Hendrik Lösch

Senior Consultant

[Hendrik.Loesch@saxsys.de](mailto:Hendrik.Loesch@saxsys.de)

[@HerrLoesch](#)

[Hendrik-Loesch.de](http://Hendrik-Loesch.de)



LinkedIn Learning



### ReSharper lernen

Effektiver programmieren mit der Erweiterung zu Visual Studio

[Hendrik Lösch](#)



### WPF-Anwendungen mit MVVM und Prism

Modulare Architekturen verstehen und umsetzen

[Hendrik Lösch](#)



### Windows 8 Store Apps mit MVVM und Prism

XAML-Entwurfsmuster, Bootstrapping, Navigation, Messaging

[Hendrik Lösch](#)



### LINQ Grundkurs

Wichtige Spracheigenschaften von C#, Joins, Variablen und andere Operationen, erweiterte Techniken

[Hendrik Lösch](#)



### Grundlagen der Programmierung: Test Driven Development

Business-Applikationen testgetrieben entwickeln

[Hendrik Lösch](#)



### Inversion of Control und Dependency Injection - Grundlagen

Prinzipien der modernen Software-Architektur ...

[Hendrik Lösch](#)



### C#: Test Driven Development

Grundlagen, Frameworks, best Practices

[Hendrik Lösch](#)



### Grundlagen der Programmierung: Codemetriken

Softwarequalität einschätzen, sicherstellen und ...

[Hendrik Lösch](#)



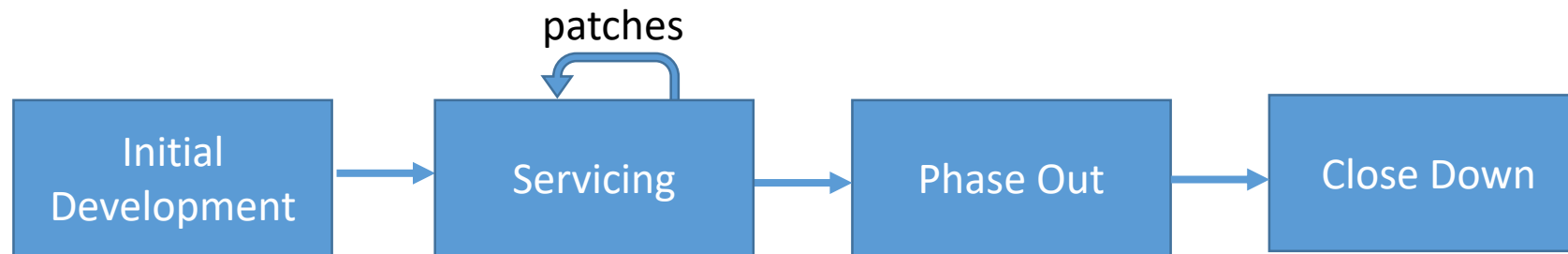
**Saxonia Systems**  
So geht Software.

# AUSGANGSSITUATION

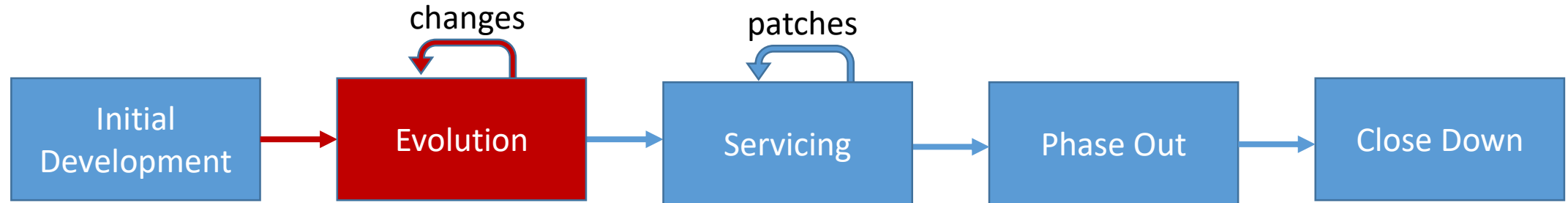


**Saxonia Systems**  
So geht Software.

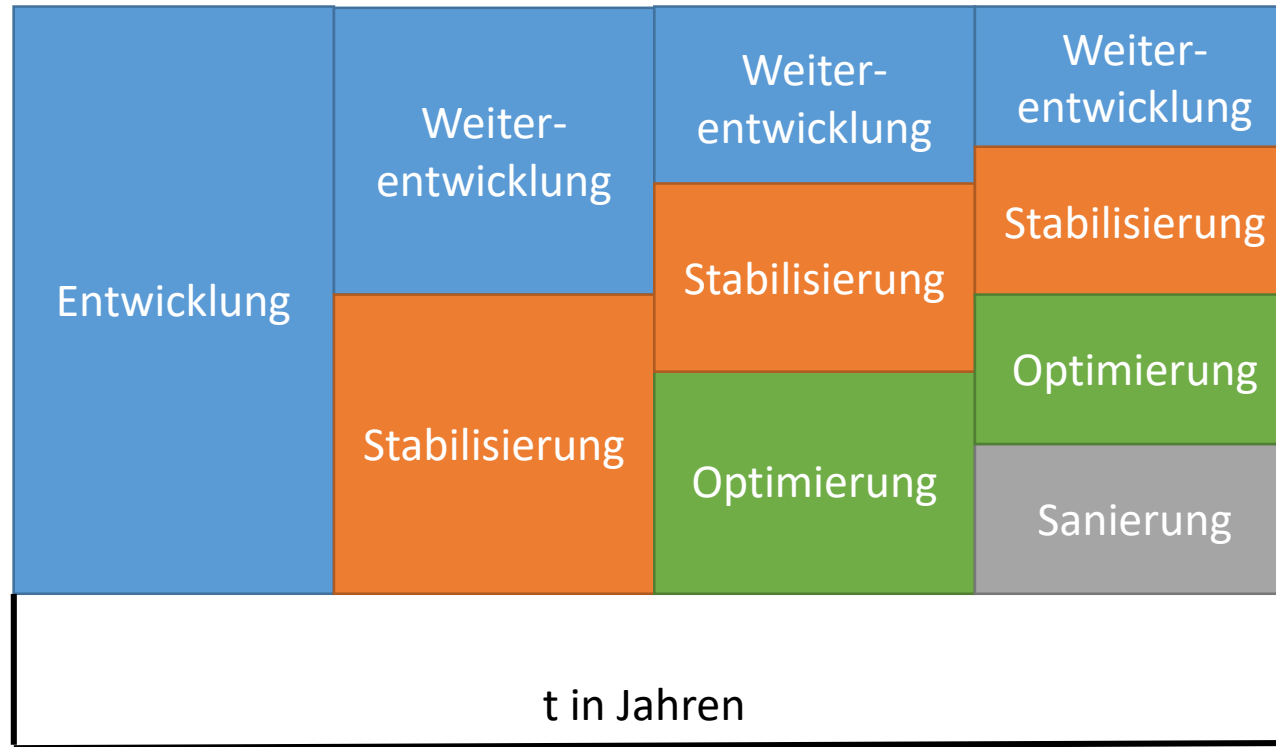
# Der Softwarelebenszyklus



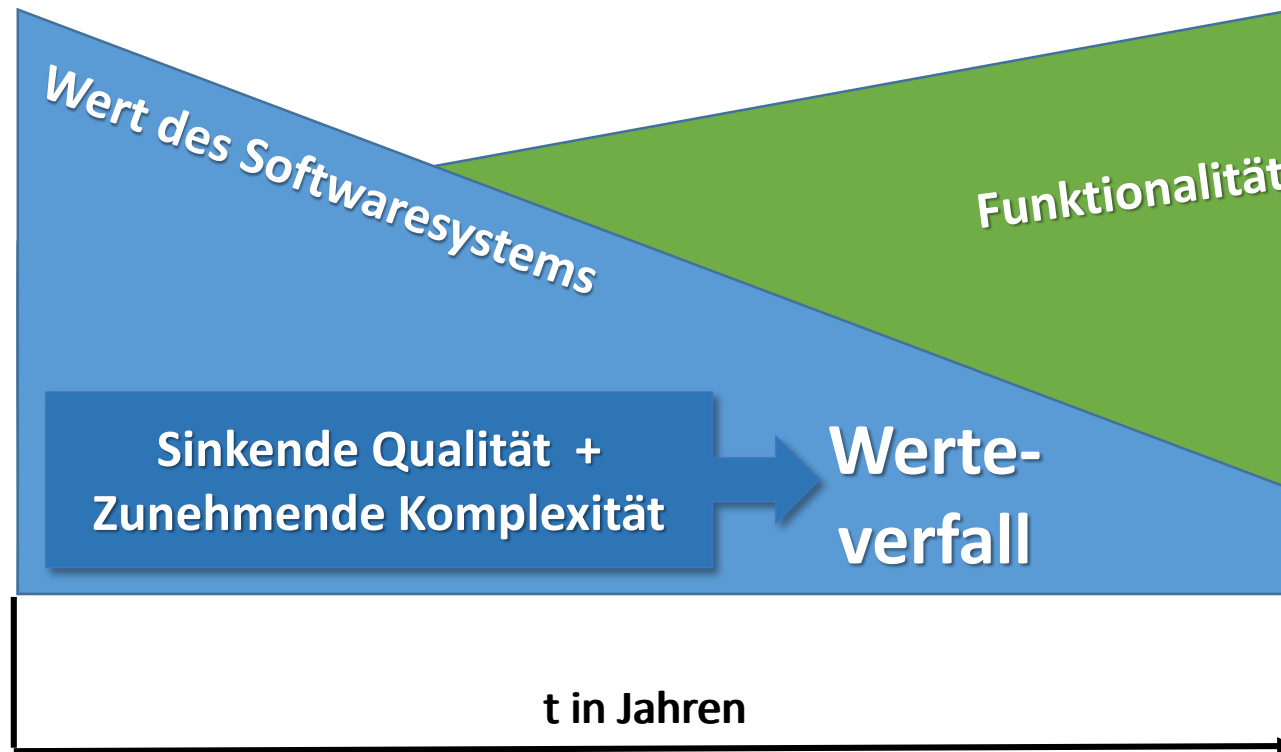
# Der Softwarelebenszyklus



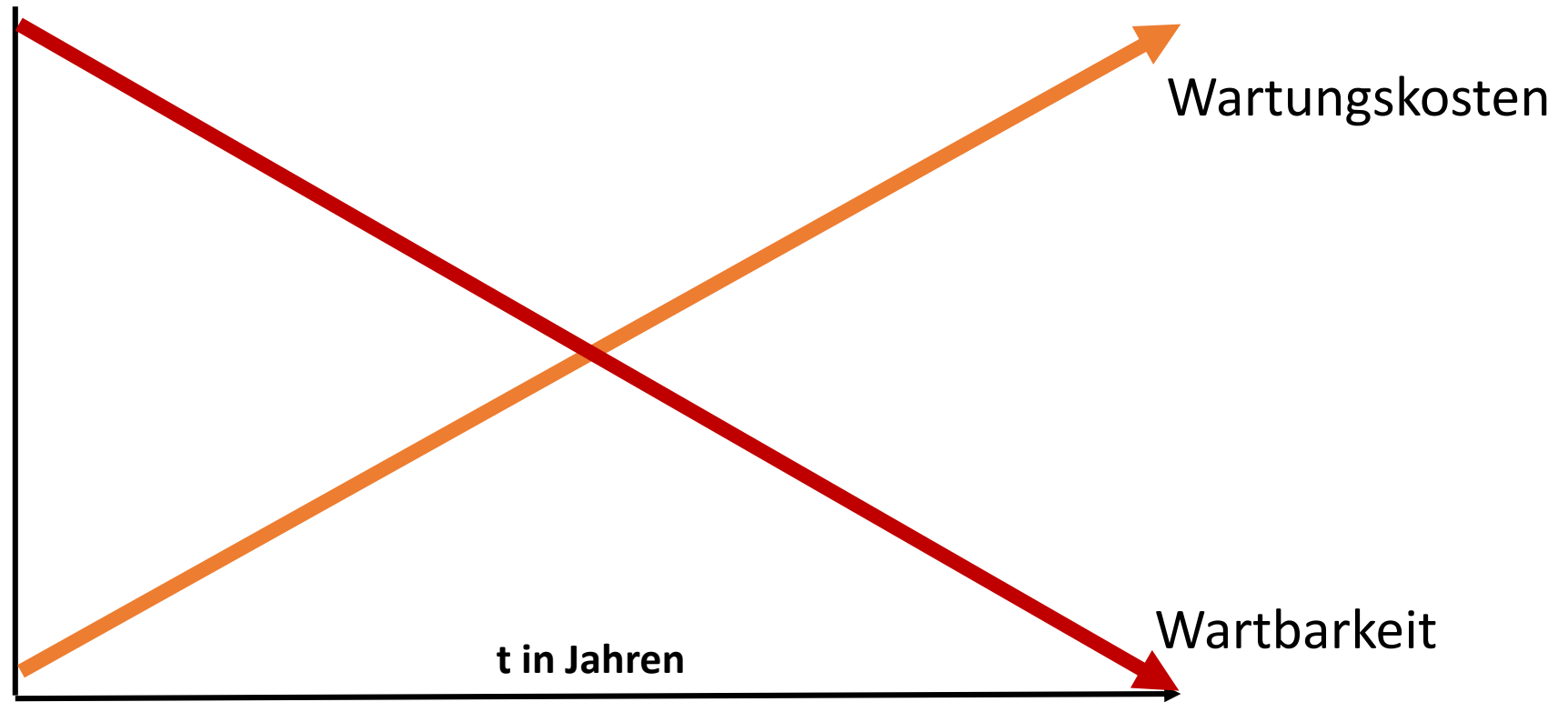
# Evolutionäre Softwareentwicklung



# Evolutionäre Softwareentwicklung



# Technische Schuld



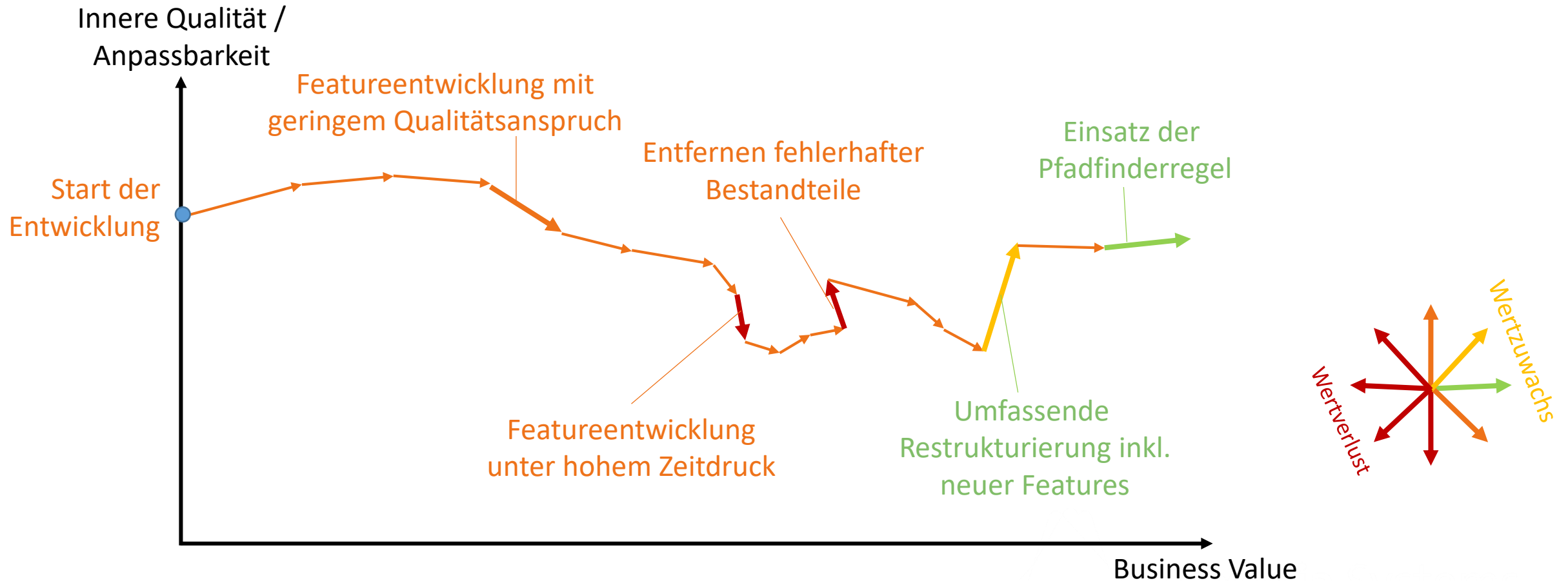


# Technische Schuld



**Saxonia Systems**  
So geht Software.

# Anpassbarkeit vs. Business Value



# DER HEALTHCHECK



**Saxonia Systems**  
So geht Software.

„Ein Ziel ohne Plan ist nur  
ein Wunsch.“

Antoine de Saint Exupéry



# Health Check

**Saxon Systems**  
So geht Software.

ICH SUCHE

SOFTWAREENTWICKLUNG ZUSAMMENARBEIT REFERENZEN KARRIERE UNTERNEHMEN AKTUELLES INITIATIVE HOLDING BLOG

SOFTWAREEVOLUTION WEB SOFTWARE HEALTHCHECK SMART BUSINESS APPLICATIONS SAP BYPASSING ASP.NET CORE

## So geht Software-Healthcheck.

Der Pflegebedarf Ihres Systems steigt und steigt ...

### Verhält sich Ihre Software noch, wie sie soll?

Treten immer wieder unvorhergesehene Fehler auf? Dauert die Implementierung von neuen Features von Jahr zu Jahr länger? Sie können Releasepläne nicht einhalten? Ihnen fehlt der richtige Ansatz für eine kontinuierliche Wartung und Weiterentwicklung?

#### IHRE HERAUSFORDERUNGEN

- › Sie wollen Ihre Software gezielt weiterentwickeln, wissen aber nicht, wo Sie ansetzen sollen
- › Sie möchten die Schwachstellen finden, die Ihre aktuelle Lösung aufweist
- › Sie benötigen eine unabhängige Einschätzung Ihrer Softwarearchitektur, um eine Weiterentwicklung planen zu können
- › Sie müssen sich entscheiden, ob Sie ein bestehendes System weiterentwickeln, oder etwas Neues beginnen sollen

Herausforderungen






Nutzen

Leistung


Ansprechpartner



# Check-Ups beim Arzt



[Leistungen & Services](#) [Mitgliedschaft & Tarife](#) [Gesundes Leben](#) [Medizin & Versorgung](#) [Pflege](#)



## Gespräch über Ihre Krankheitsgeschichte

In einem Gespräch berichten Sie Ihrem Arzt von Ihrem Gesundheitszustand. Anschließend sprechen Sie über Ihre Lebens- und Ernährungsgewohnheiten: Haben Sie berufliche oder familiäre Sorgen? Wie sieht es mit Sport aus, wie viel Alkohol trinken Sie oder sind Sie beispielsweise Raucher? Nutzen Sie dieses ausführliche Gespräch und seien Sie offen, denn es geht um Ihre Gesundheit.

## Die Untersuchungen im Überblick

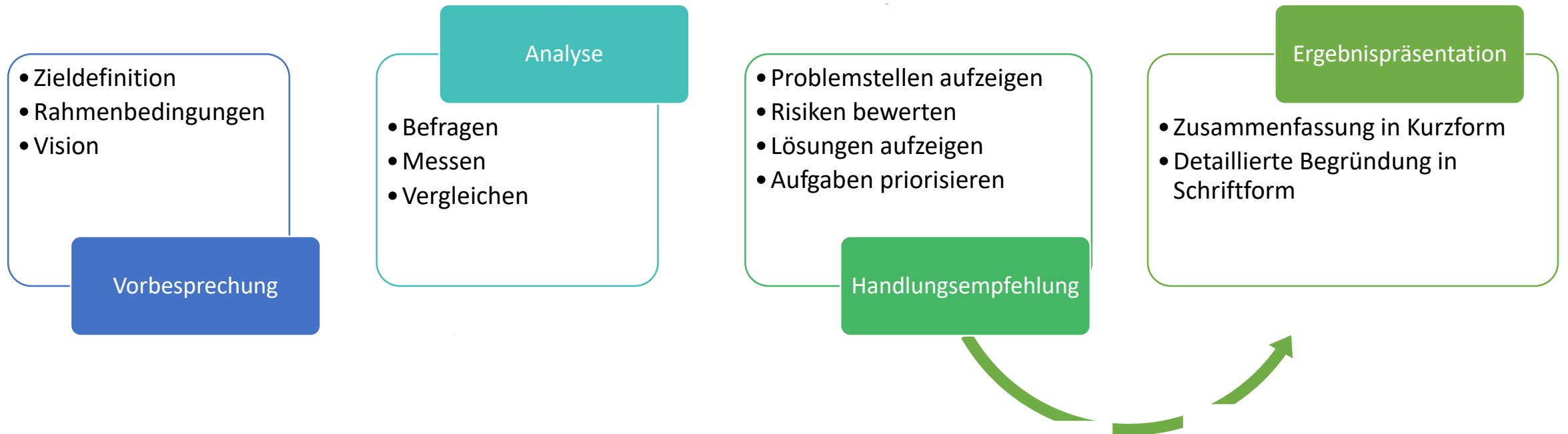
Im nächsten Schritt des Check-ups wird der Arzt Ihren Körper gründlich untersuchen. Dazu gehören:

- Überprüfung von Herz, Lunge, Bauch, Bewegungsapparat, Nervensystem und Sinnesorganen, um abweichende oder krankhafte Befunde feststellen zu können.
- Die Entnahme einer Blutprobe: Diese wird im Labor ausgewertet und gibt Aufschluss über Cholesterin- und Blutzuckerwerte. Anhand dieser Werte kann der Arzt erkennen, ob zum Beispiel der Verdacht auf eine Zuckerkrankheit (Diabetes) besteht.
- Das Messen des Blutdrucks: Zusammen mit dem Cholesterinwert kann der Blutdruck auf Risiken für Herz-Kreislauf-Erkrankungen wie Arteriosklerose (Gefäßverkalkung) oder für einen Herzinfarkt hinweisen.
- Die Untersuchung des Urins: Hinweise auf Nieren- und Blasenerkrankungen, aber auch eine Zuckerkrankheit lassen sich nach einer Urinprobe auf dem Teststreifen ablesen.

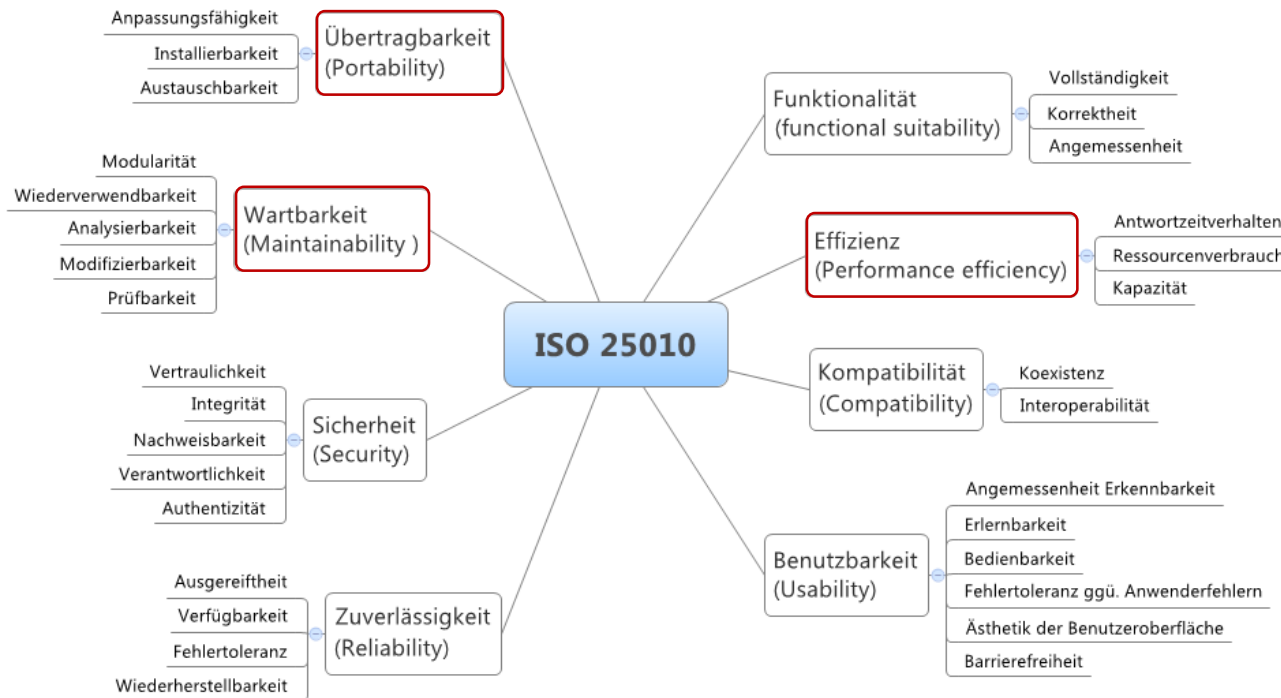
Service für AOK PLUS Versicherte: Für Versicherte der AOK PLUS gehört zusätzlich eine Diabetes-Vorsorge-Untersuchung zum



# Ablauf



# Typische Bestandteile



- Zieldefinition
- Rahmenbedingungen
- Systembetrachtung
- Betrachtung des Entwicklungsprozesses
- Priorisierte Handlungsempfehlungen
- Zusammenfassung



# SYSTEMBETRACHTUNG



**Saxonia Systems**  
So geht Software.

# Laufzeitverhalten

Am Laufzeitverhalten erkennt man wie die Software tatsächlich arbeitet.

Copyright 2007 by Randy Glasbergen.  
www.glasbergen.com



**"The bad news is, our customers hate us. The good news is, we have a lot fewer customers than we used to!"**

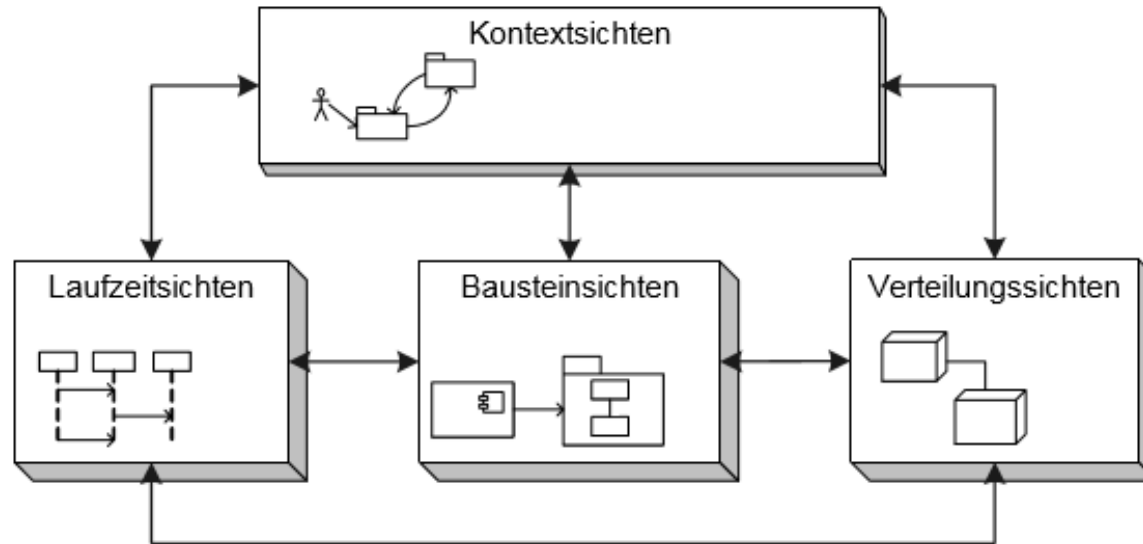
- Welche Workflows sind die wichtigsten?
- Wie ist Laufzeitstabilität?
- Wie ist das generelle Geschwindigkeitsempfinden?
- Was sagen die eigentlichen Nutzer?
- Gibt es Usability & Last Tests?



**Saxonia Systems**  
So geht Software.

# Dokumentation

Noch wichtiger als das Wie, ist das Warum!

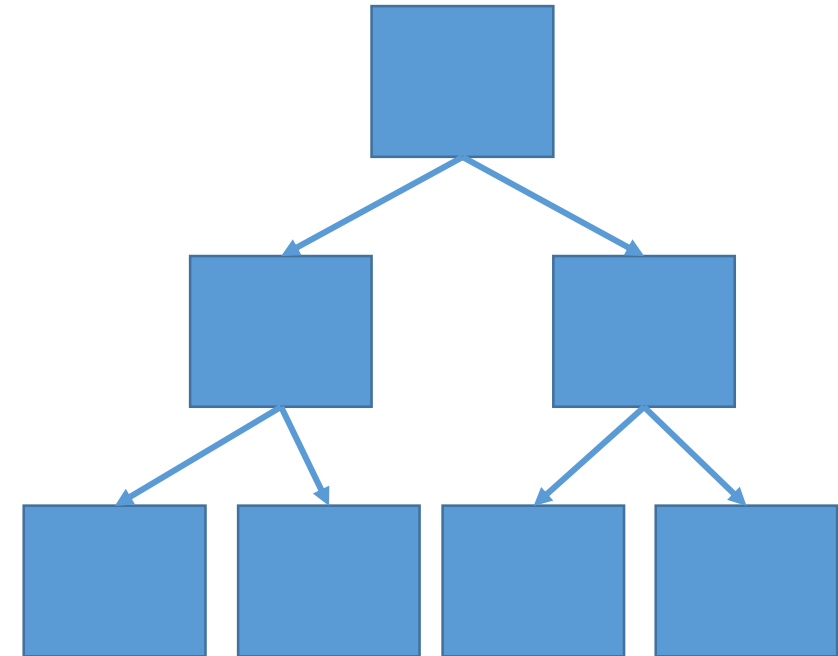


„Mein persönlicher Tipp: Verzichten Sie möglichst auf weitere Sichten. Jede Sicht kostet Erstellungs- und Wartungsaufwand, der Sie eventuell von (wichtigeren) Architekturaufgaben abhält. Die grundlegenden Aspekte der Architektur- und Systementwicklung decken Sie bereits mit den drei Sichten (Bausteinsicht, Laufzeitsicht, Verteilungssicht) ab.“

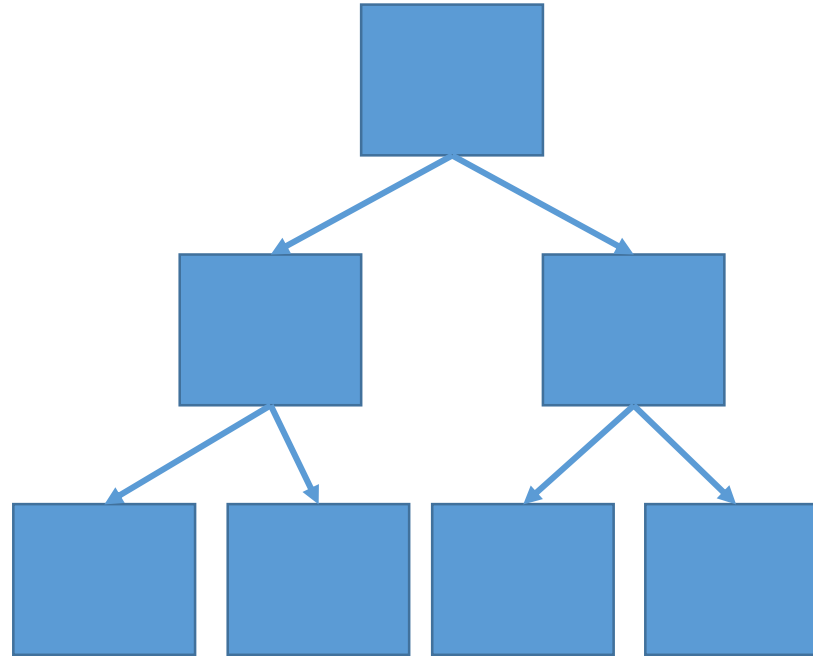
Gernot Starke

# Architektur

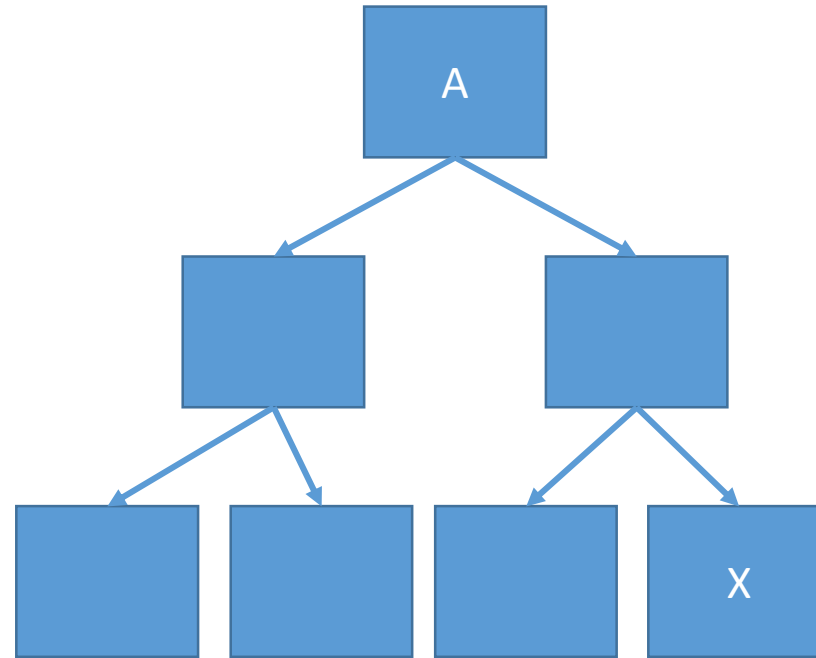
- Sind eindeutige Verantwortlichkeiten zu erkennen?
- Gibt es eine Trennung zwischen fachlicher Domäne und technischer Infrastruktur?
- Wie ist das Verhältnis zwischen abstrakten zu konkreten Datentypen?
- Wird IoC verwendet?
- Entspricht die Architektur der Dokumentation?



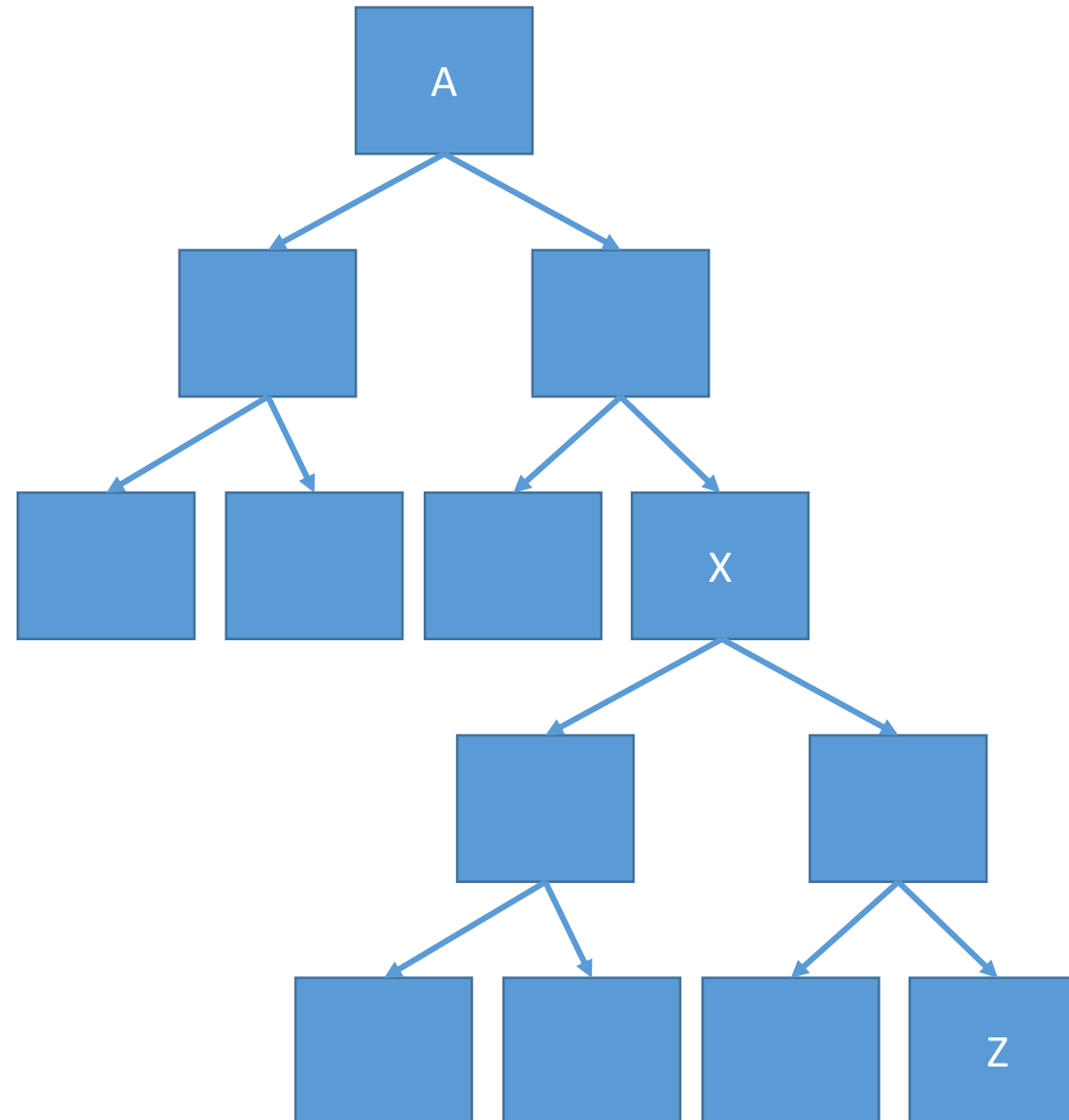
# Abhängigkeiten



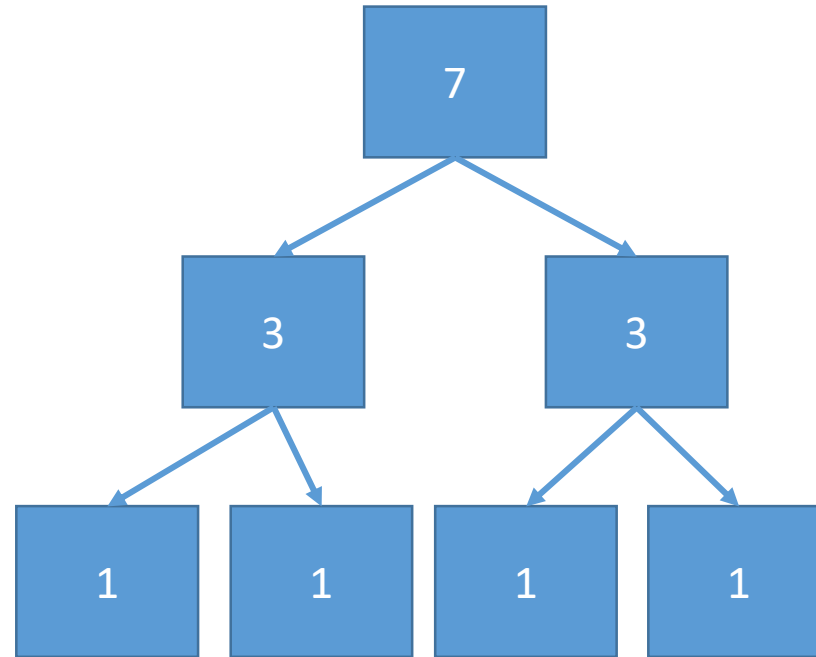
# Abhängigkeiten



# Abhängigkeiten



# Abhängigkeiten

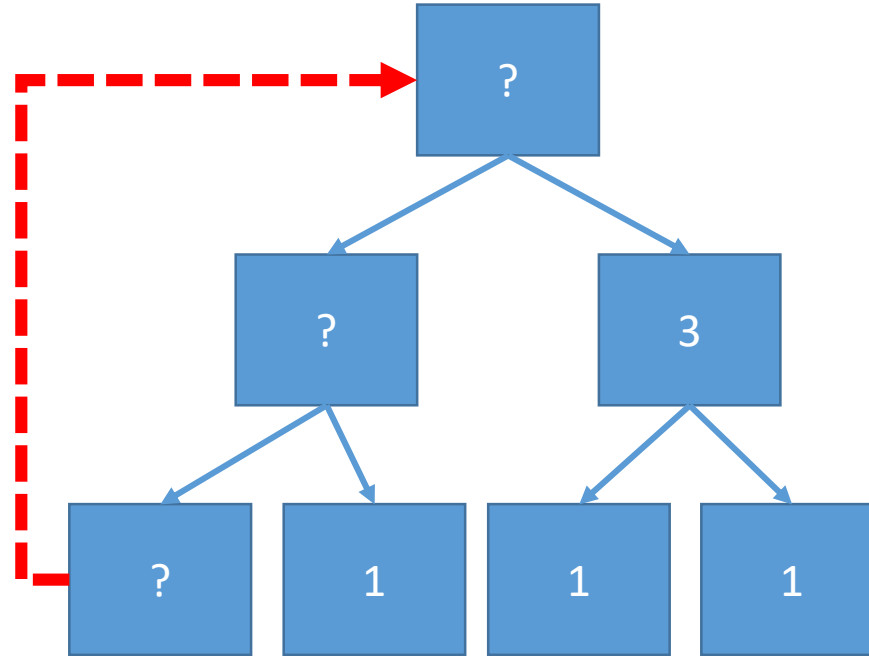


Kumulierte Komponentenabhängigkeit =  $1 \cdot 7 + 2 \cdot 3 + 4 \cdot 1 = 17$   
(cumulated component dependency)



# Abhängigkeiten

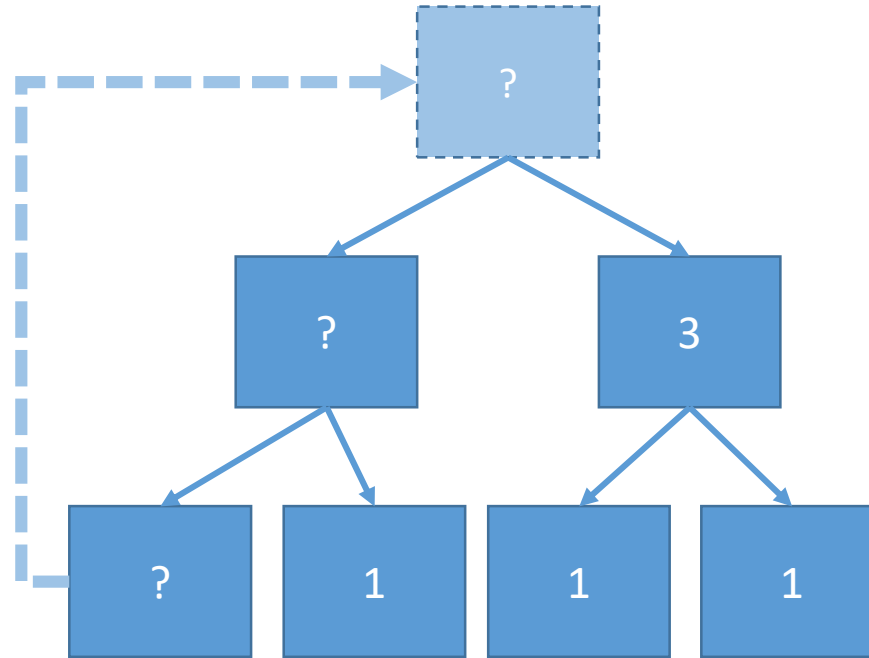
Kreisabhängigkeiten!!!



Kumulierte Komponentenabhängigkeit =  $n^{k+1} = 7^{1+1} = 49$   
(cumulated component dependency)

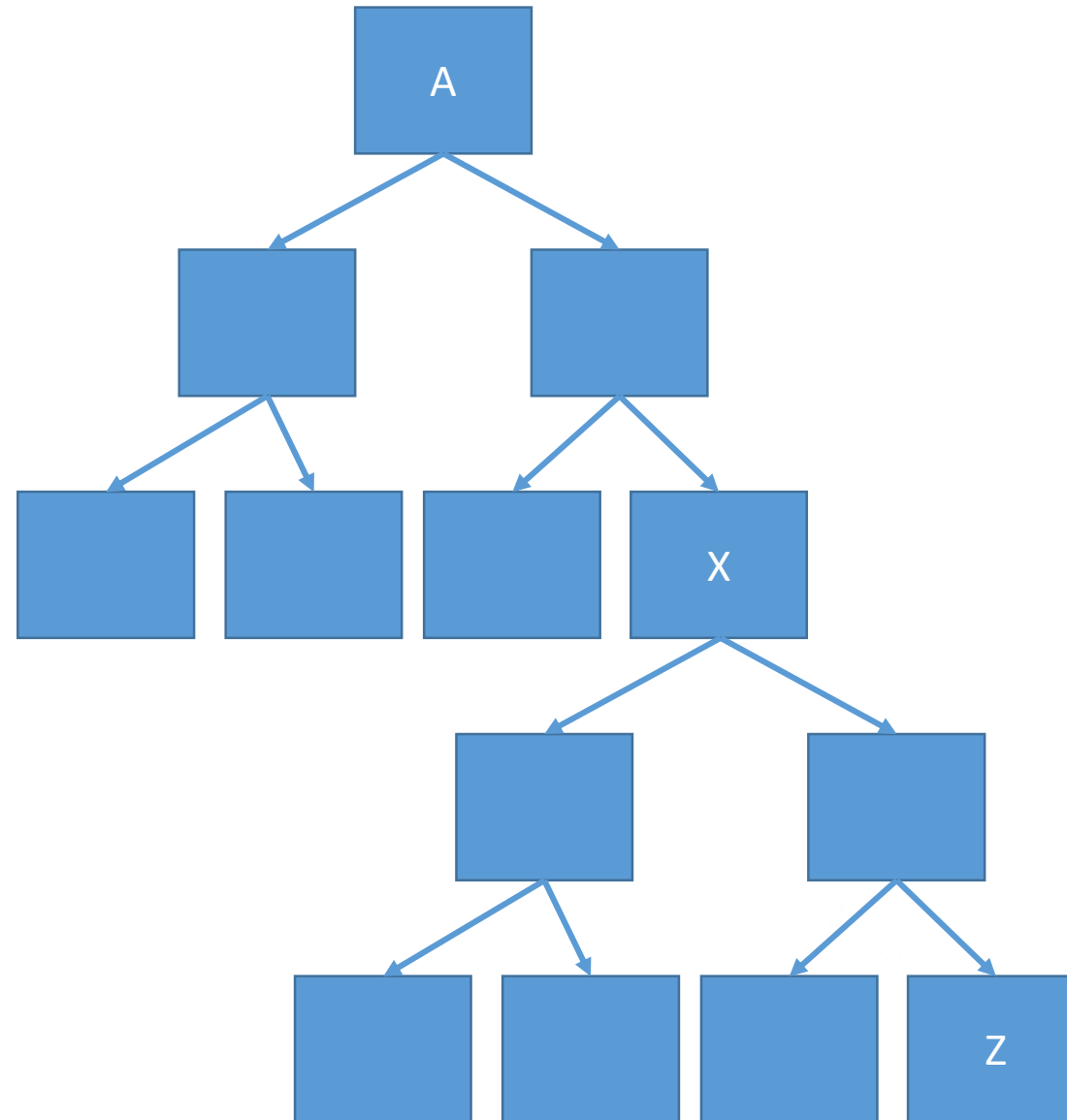


# Abhängigkeiten

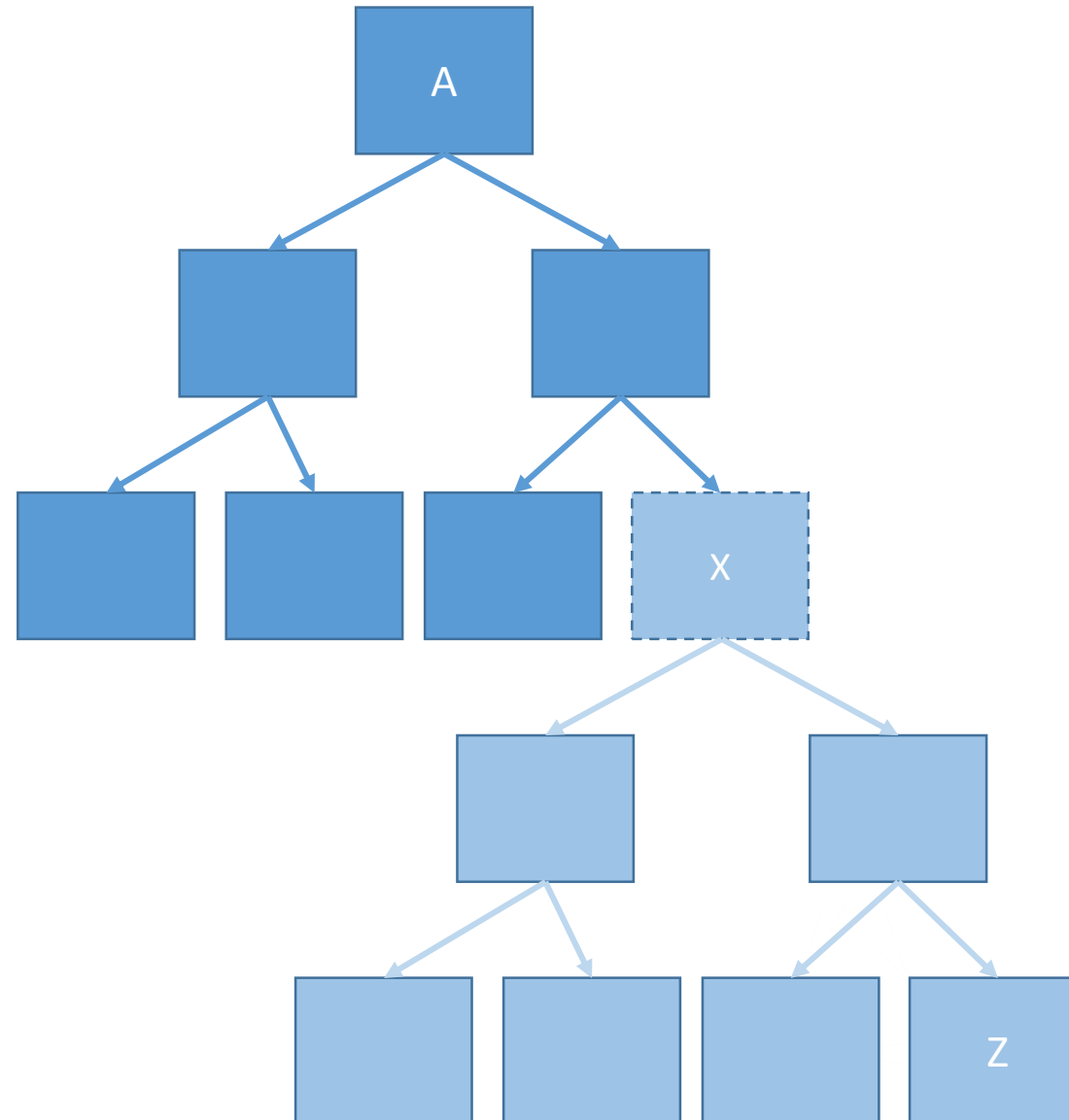


Kumulierte Komponentenabhängigkeit =  $7^2 = 49$   
(cumulated component dependency)

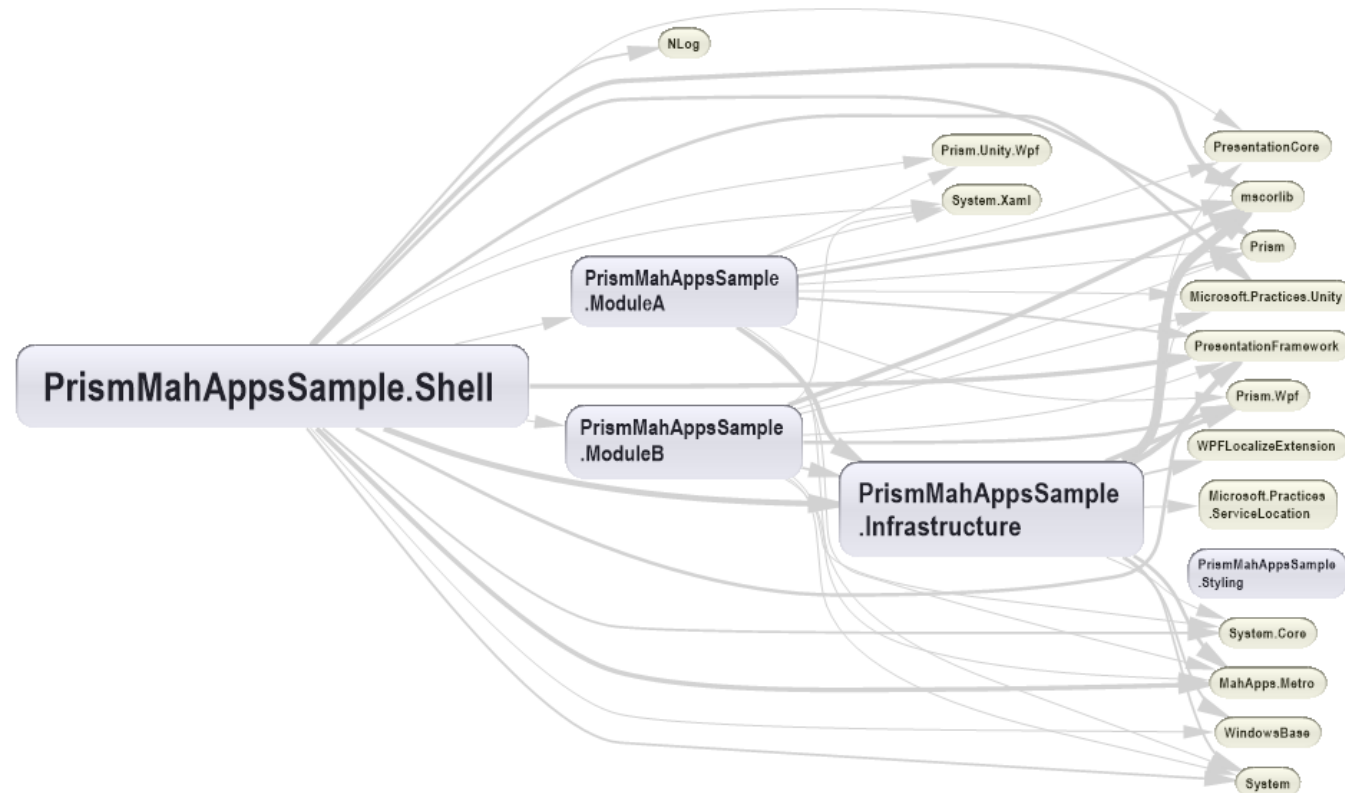
# Abhängigkeiten



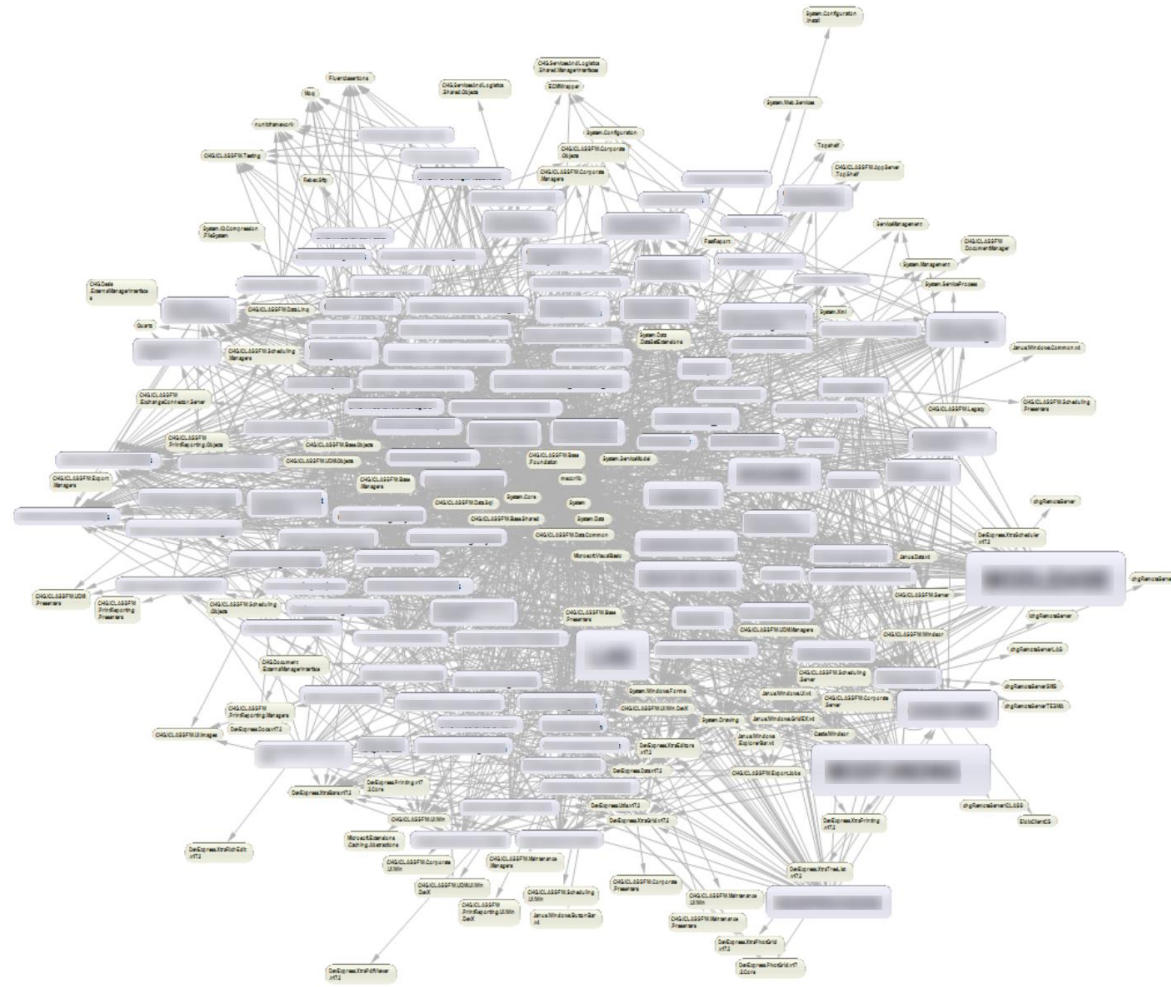
# Abhängigkeiten



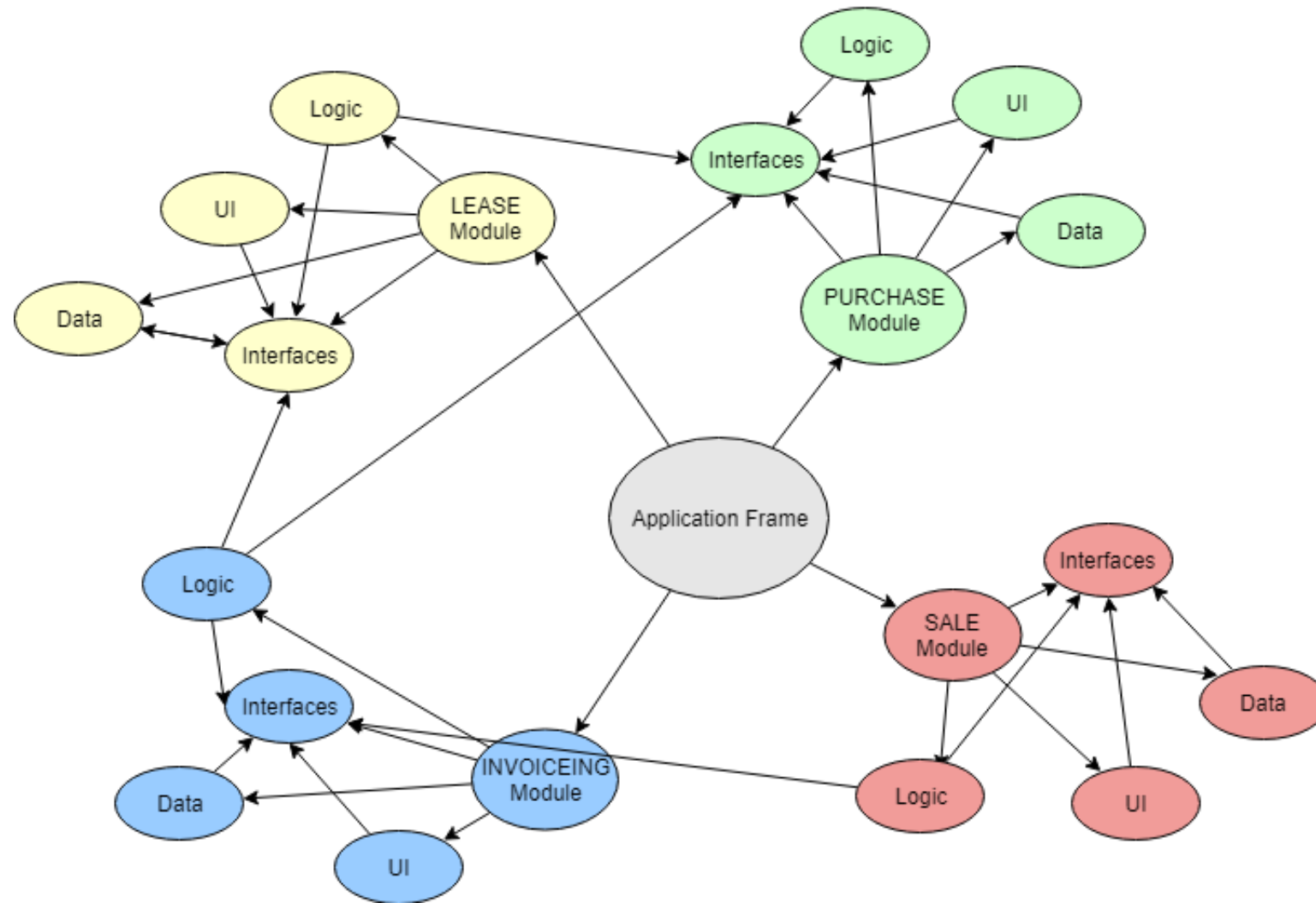
# Abhängigkeitsgraph



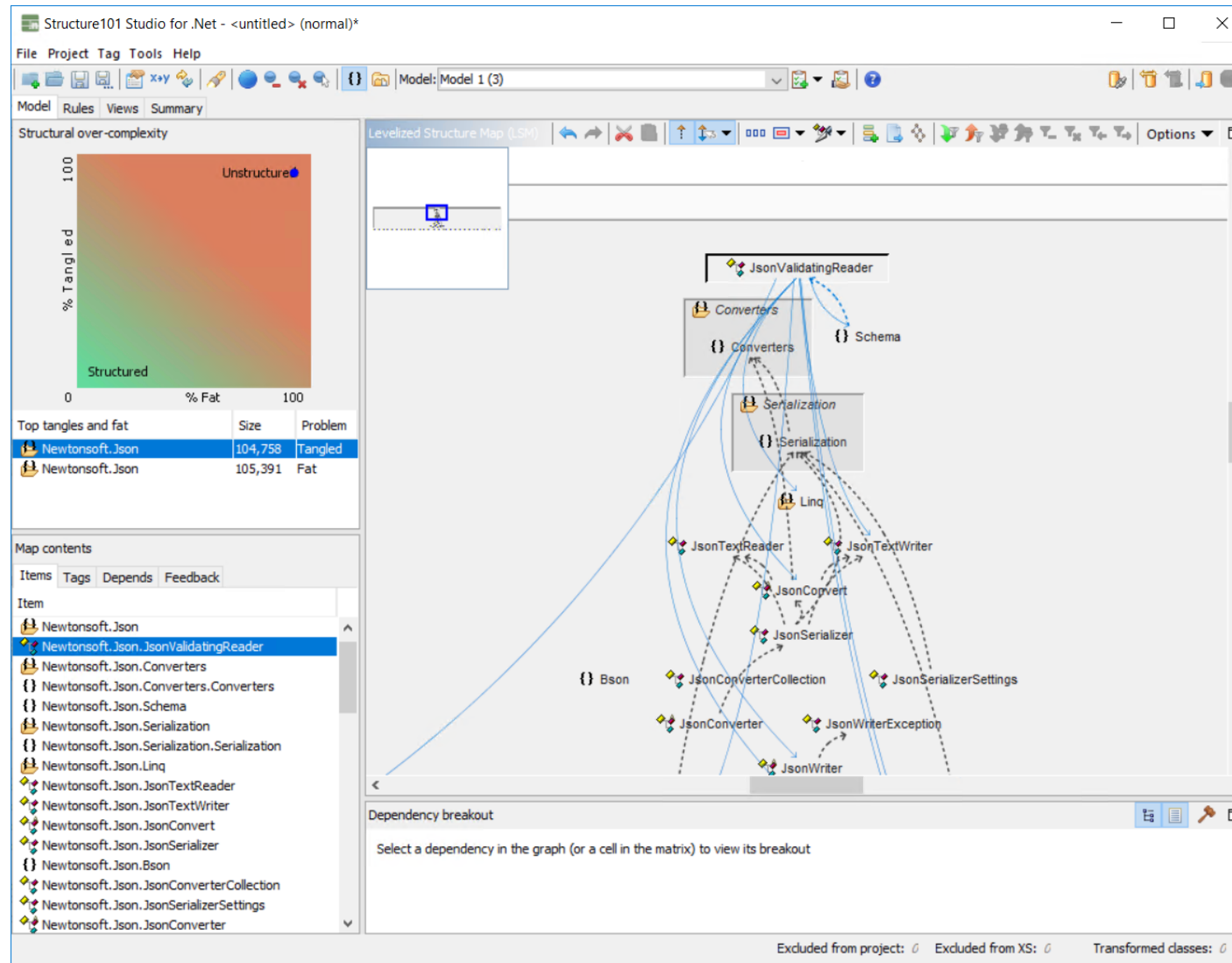
# Abhängigkeitsgraph



# Abhängigkeitsgraph



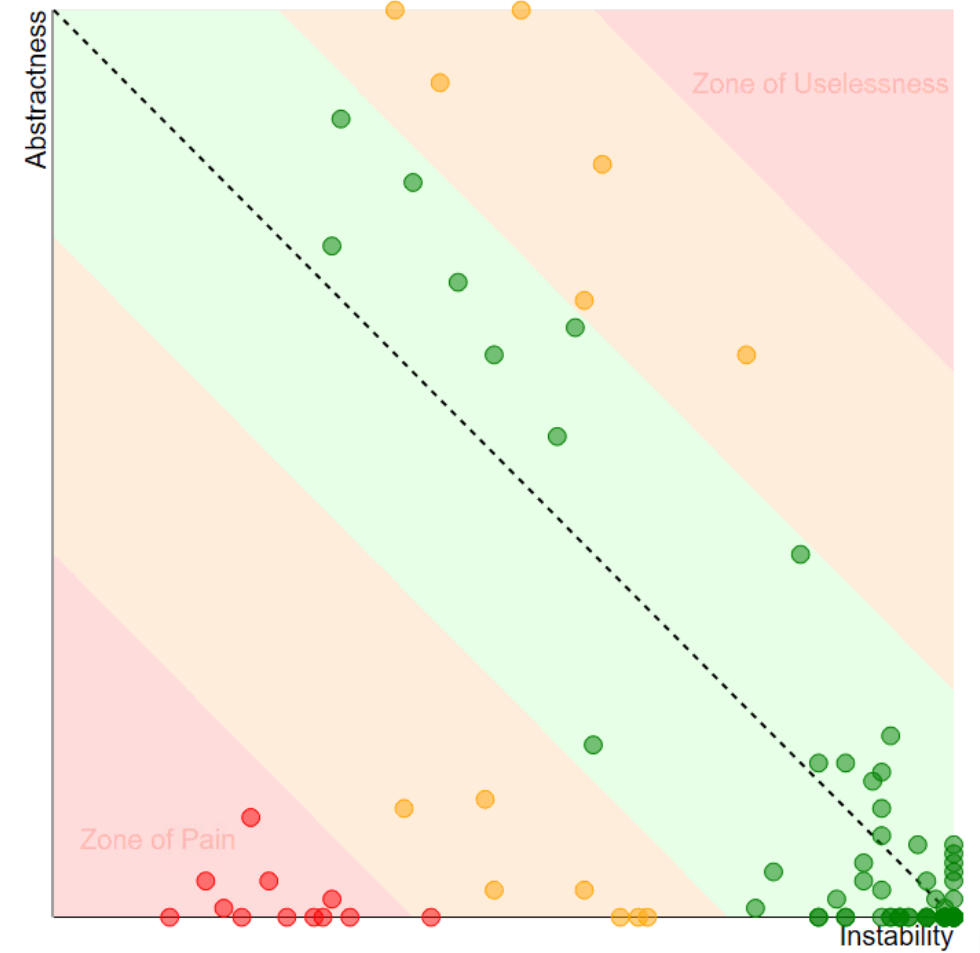
# Structure101 Studio



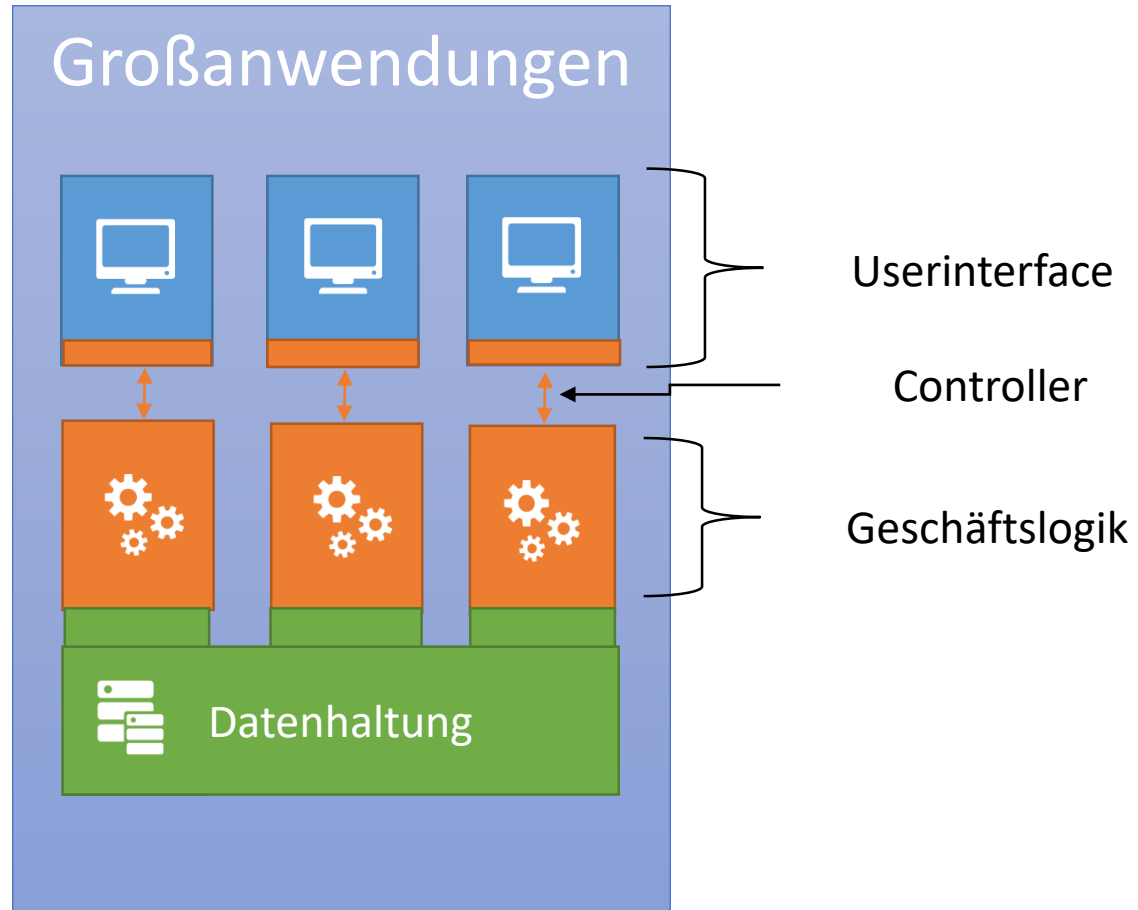


# Projektstruktur

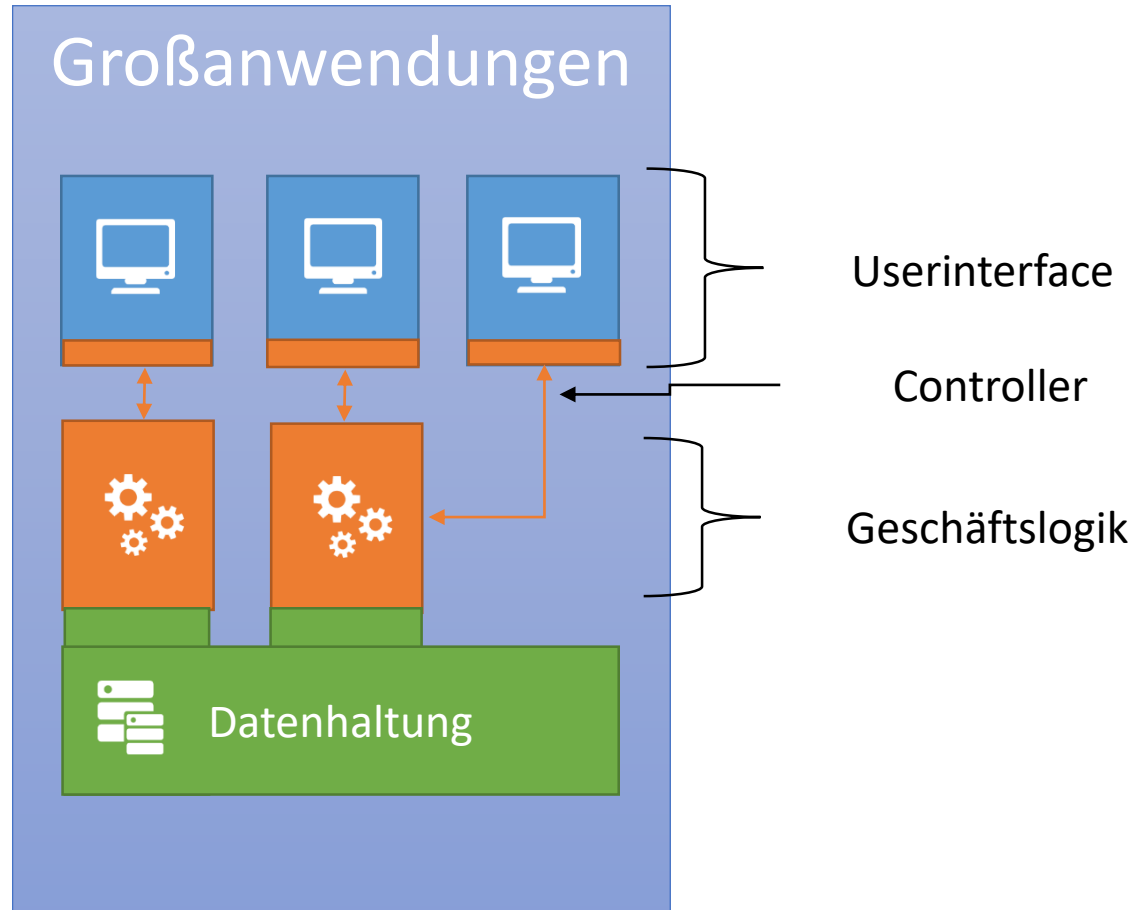
- Erste Hinweise auf die Schichten(-trennung) der Architektur.
- Verhältnis zwischen Abstraktion und Instabilität von Typen.
  - Typen die viel genutzt werden sollten über eine Abstraktion verfügen.
- Wie viele 3rd Party Bibliotheken werden genutzt und wie aktuell sind sie?



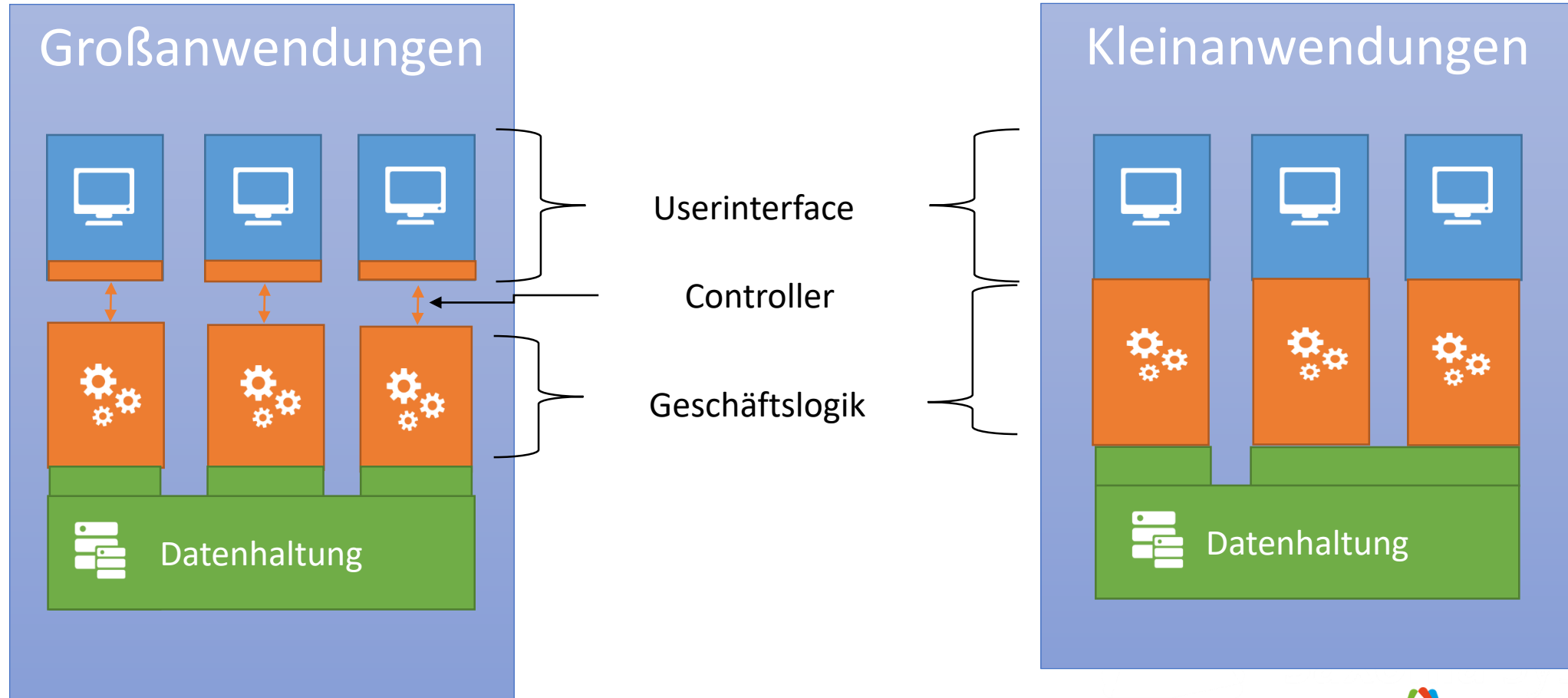
# Schichtentrennung



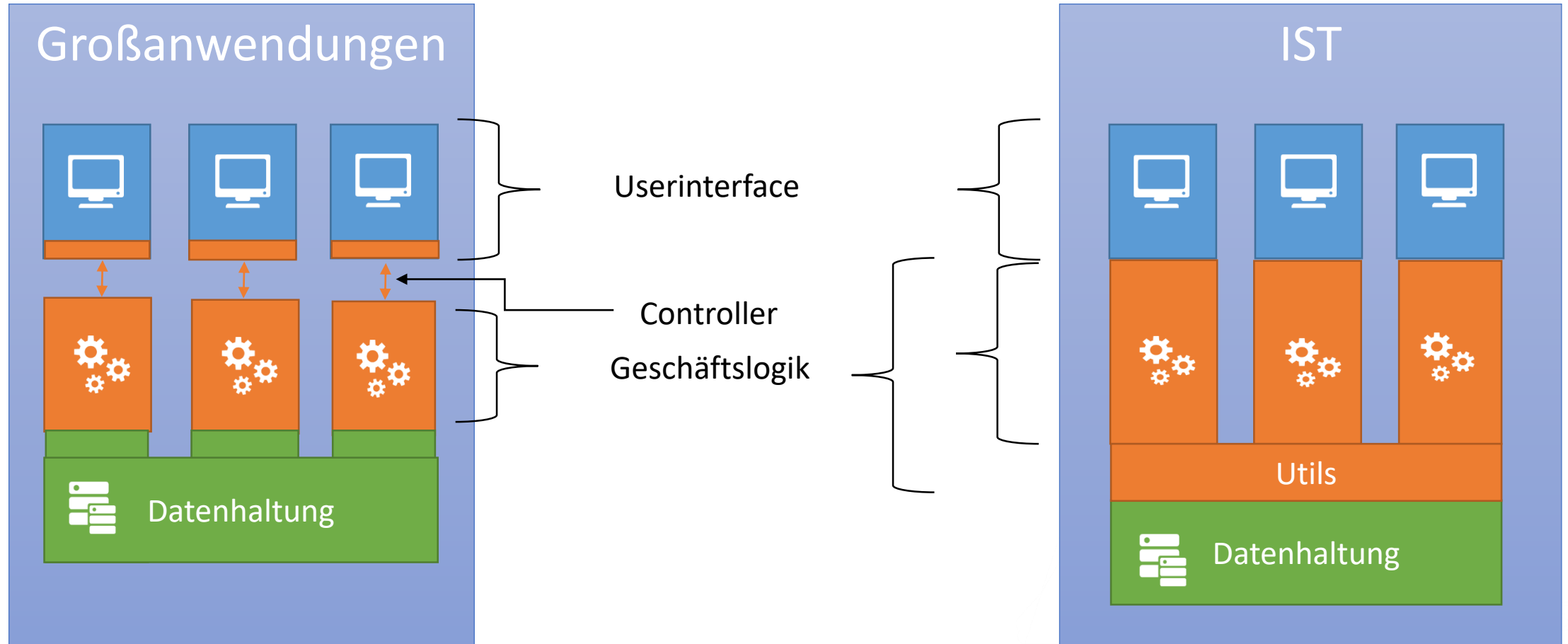
# Schichtentrennung



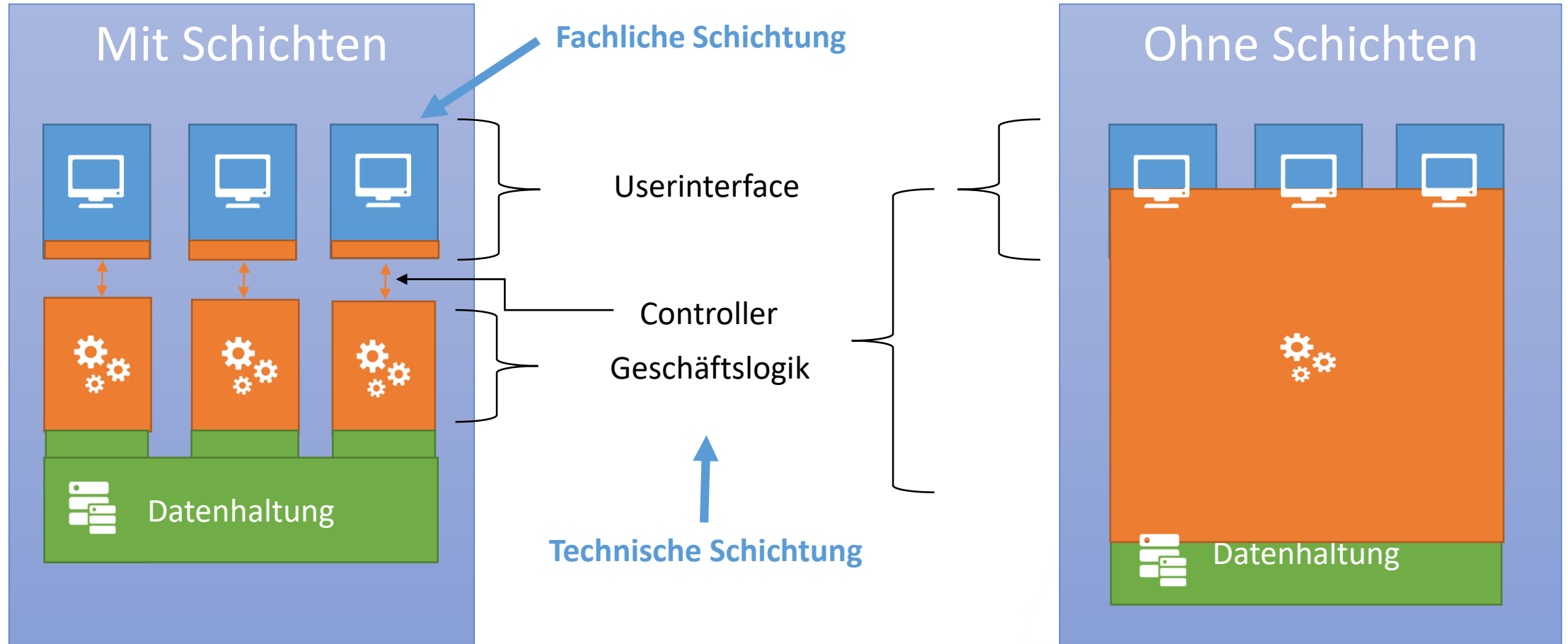
# Schichtentrennung



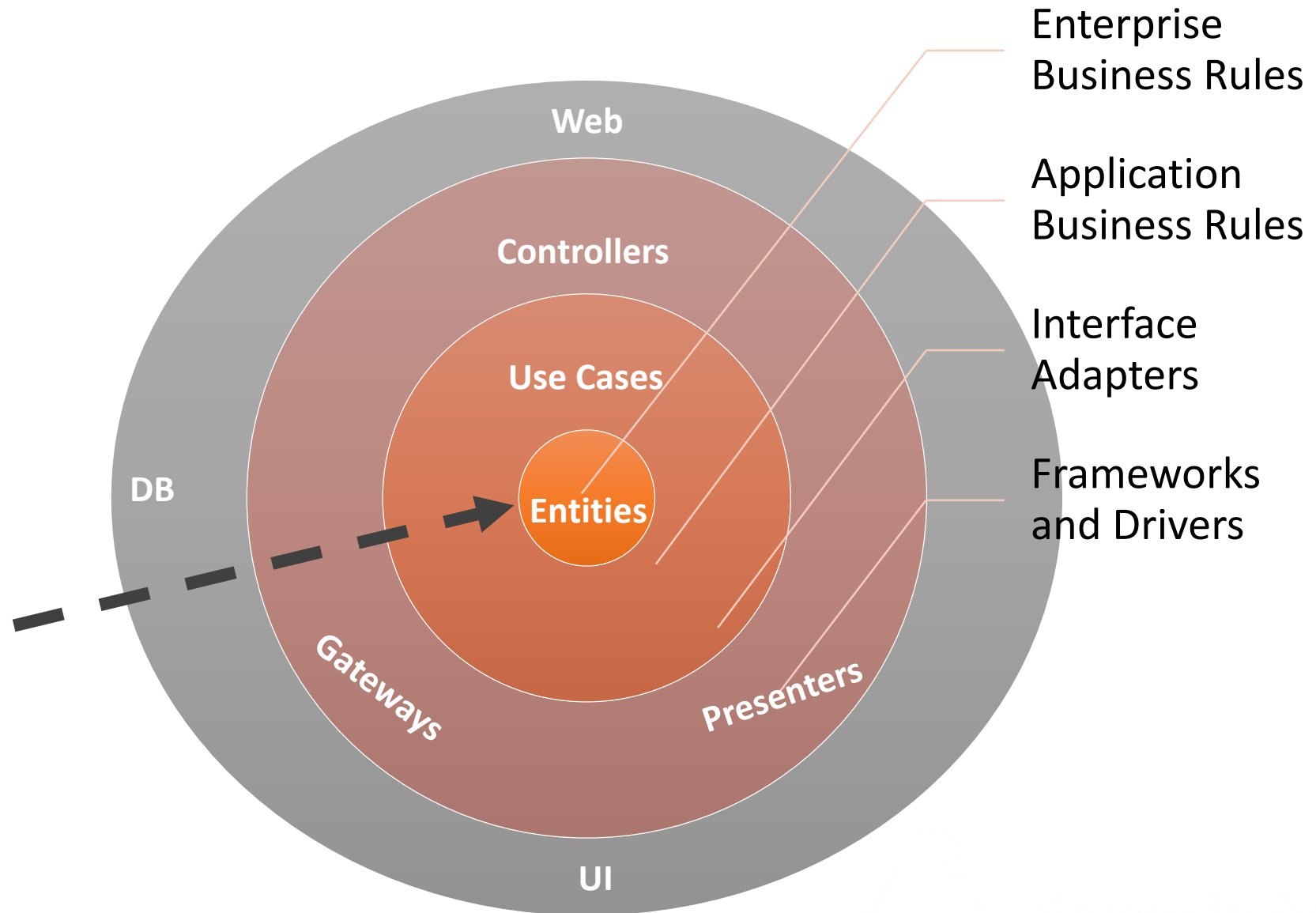
# Schichtentrennung



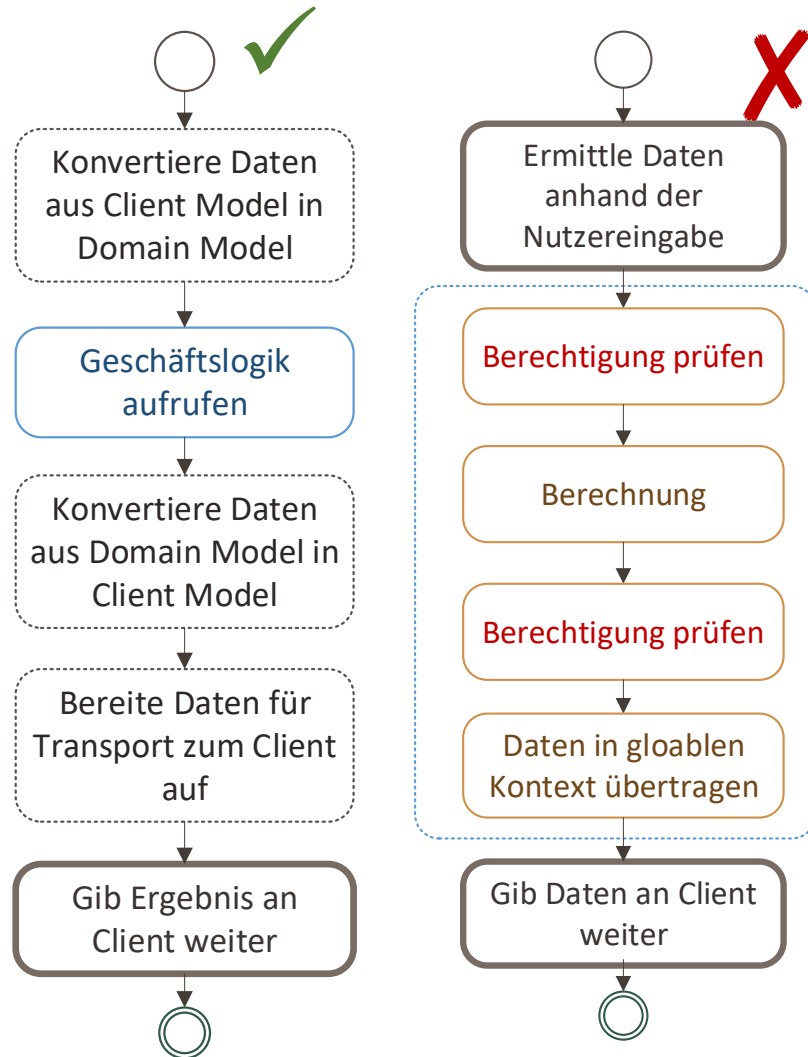
# Schichtentrennung



# Architektur



# Ablaufsteuerung und Geschäftsregeln






















- Ist der Ablauf von Aktionen klar nachvollziehbar?
- Gibt es eine strikte Schichtentrennung?
- Gibt es eine deutliche Trennung zwischen Controller und Geschäftslogik und daher eine sichtbare Vermittlungsschicht?
- Finden sich an vielen Stellen im Code duplizierte Helfermethoden?
- Wie groß ist der „Utils“ Bereich im Verhältnis zum Rest des Systems?
- Gibt es eine Trennung zwischen Logik und Daten?
- Sind Datenobjekte mit fremder Frameworklogik verschmutzt?





# Generelle Codequalität

	552 Source Files	# Source Files
	16 716 Lines	# Lines of Comments
	2 112 Methods	# Public Methods
	3 164 Methods	# Methods
	0.44 %	% Methods with more than 100 LOC
	688 Types	# Types
	445 Types	# Public Types
	546 Types	# Classes
	4 Types	# Abstract Classes
	22 Types	# Interfaces
	16 Assemblies	# Assemblies
	1.11 %	% class with more than 500 LOC
	14	Methods with more than 100 LOC
	5	Methods with more than 250 LOC
	6	Classes with more than 500 LOC
	4	Classes with more than 1000 LOC
	10	Potentially dead Types
	154	Potentially dead Methods
	2	Potentially dead Fields

- Gibt es wenige große oder viele kleine Klassen?
- Konkrete vs. Abstrakte Datentypen?
- Interfaces vs. Abstrakte Klassen
- Vererbung vs. Komposition?
- Werden Entwurfsmuster sinnvoll eingesetzt?
- Anteil der Kommentare am Gesamtvolumen?
- Anteil toten Codes am Gesamtvolumen?

# Generelle Codequalität

$$\text{C.R.A.P.}(m) = \text{CC}(m)^2 * (1 - \text{Coverage}(m)/100)^3 + \text{CC}(m)$$

Method's CC	% of coverage required to be below CRAPpy threshold
0 – 5	0%
10	42%
15	57%
20	71%
25	80%
30	100%
31+	No amount of testing will keep methods this complex out of CRAP territory.

- Coding Guidelines?
- Lesbarkeit des Codes?
- Anzahl Code Clones?
- Höhe der Testabdeckung?
- System vs. Unit-Tests?
- C.R.A.P. Factor?



# Wiederkehrende Muster

```
Publications =  
repository.GetPublications().Select(x => new  
PublicationViewModel(x)).ToList();
```

!=

```
Publications =  
repository.GetPublications().Select(x => new  
PublicationViewModel(x));
```

- Treten Anti-Pattern systematisch auf?
- Wurden Frameworks oder Vorgehen evtl. falsch verstanden?
- Wurden Vorgehen aus anderen Programmiersprachen verwendet, die in der vorliegenden aber schädlich sind?
- Lassen sich im Code Wissenslücken des Teams ablesen?

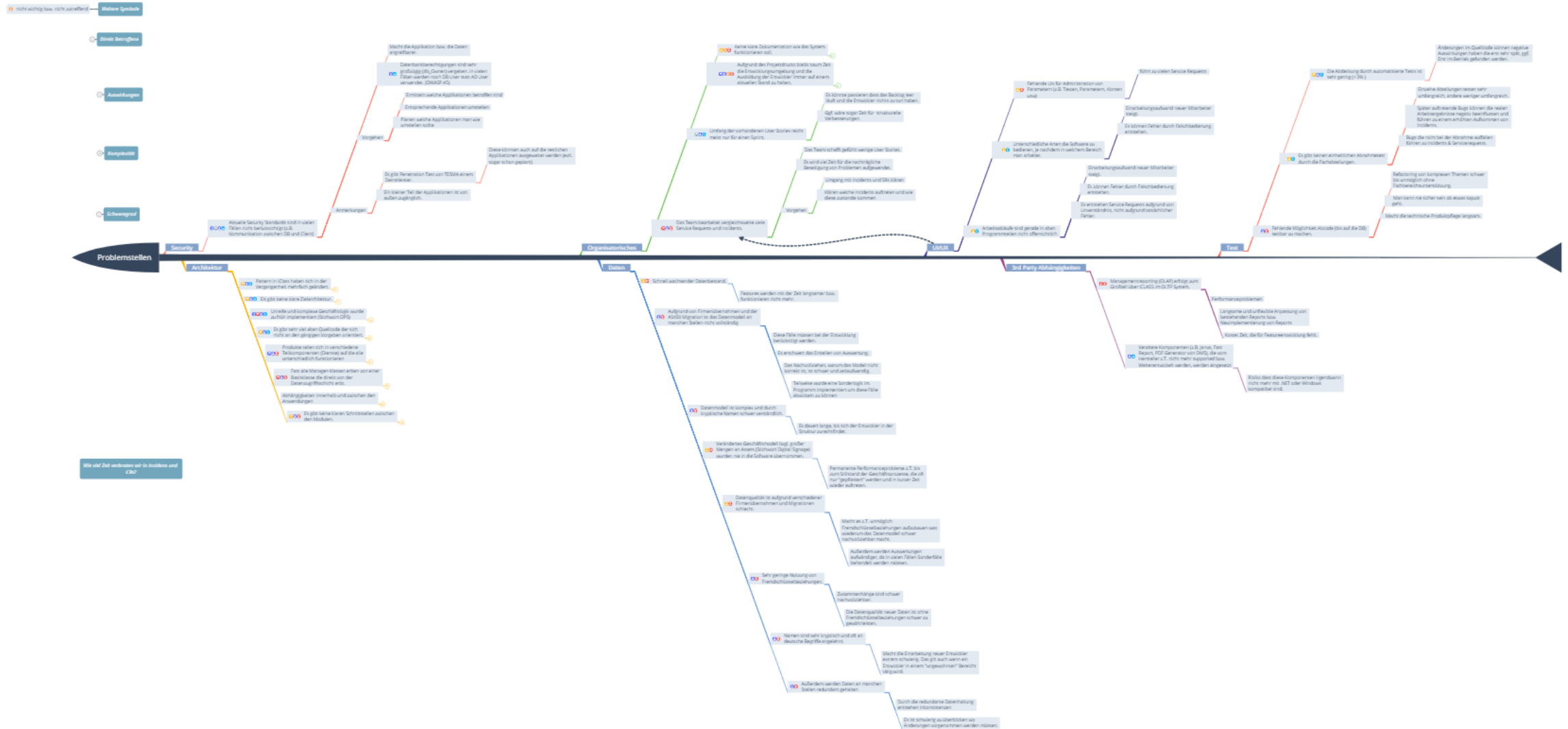


## Weitere Punkte

- User Interface
  - Erwartungskonformität?
  - Welche Frameworks werden eingesetzt?
  - Werden sie „richtig“ eingesetzt?
- Persistenz
  - Finden sich Stored Procedures?
  - Wurden Normalformen und Schlüsselbeziehungen eingehalten?
  - Gibt es Indizes in den Tabellen?
- Sicherheit
  - Ist Code Injection möglich?
  - Wird HTTPS verwendet?
  - Werden Passwörter und Konfigurationen verschlüsselt?
- Ausnahme und Fehlerbehandlung
- Testautomatisierung



# Issue Map

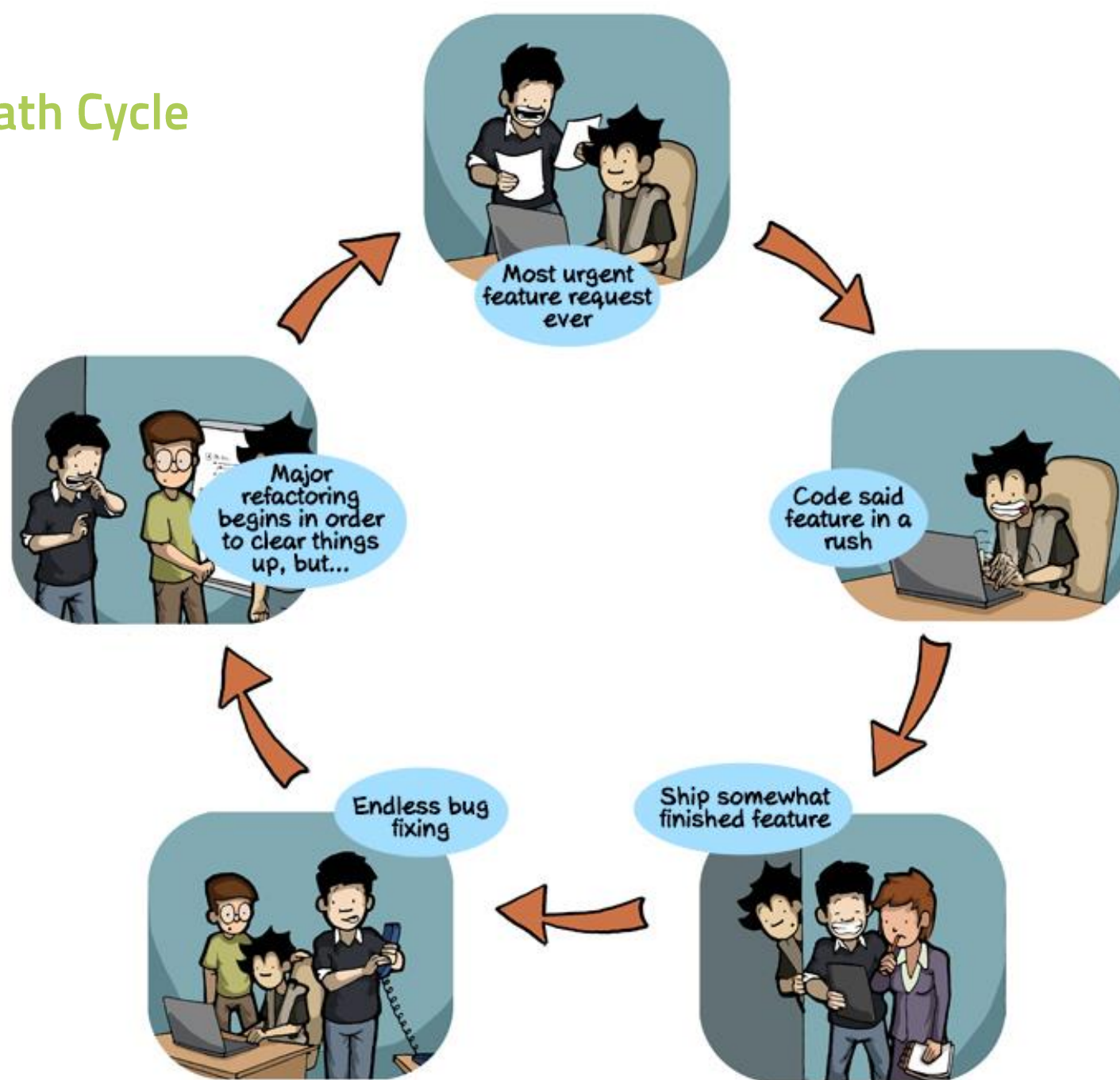


# DER ENTWICKLUNGSPROZESS



**Saxonia Systems**  
So geht Software.

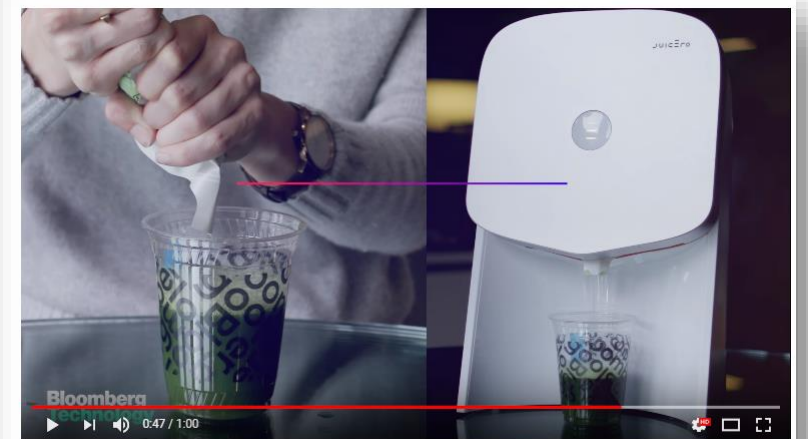
# Software Death Cycle



## Developers gone wild



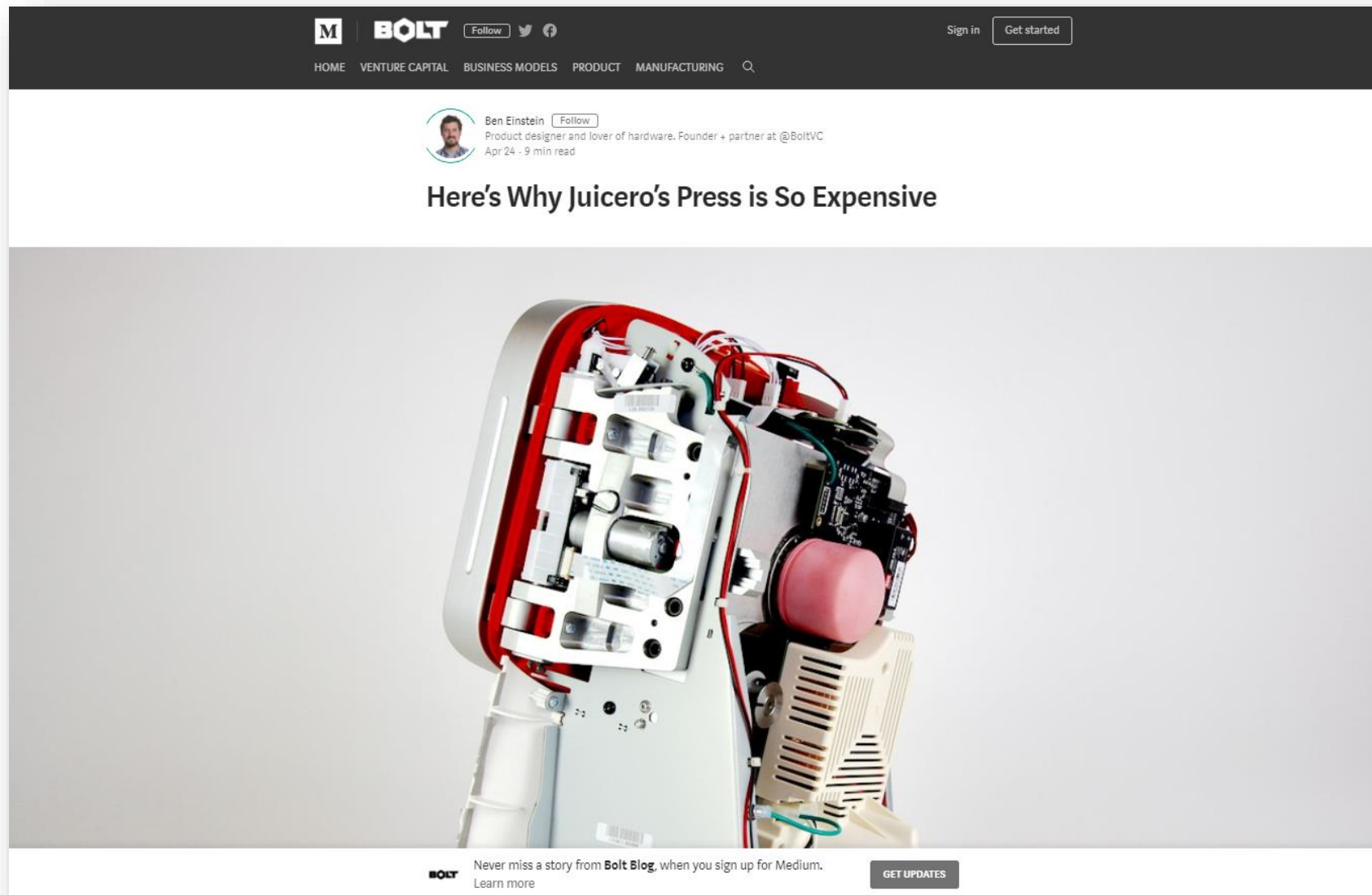
Do You Need a \$400 Juicer?



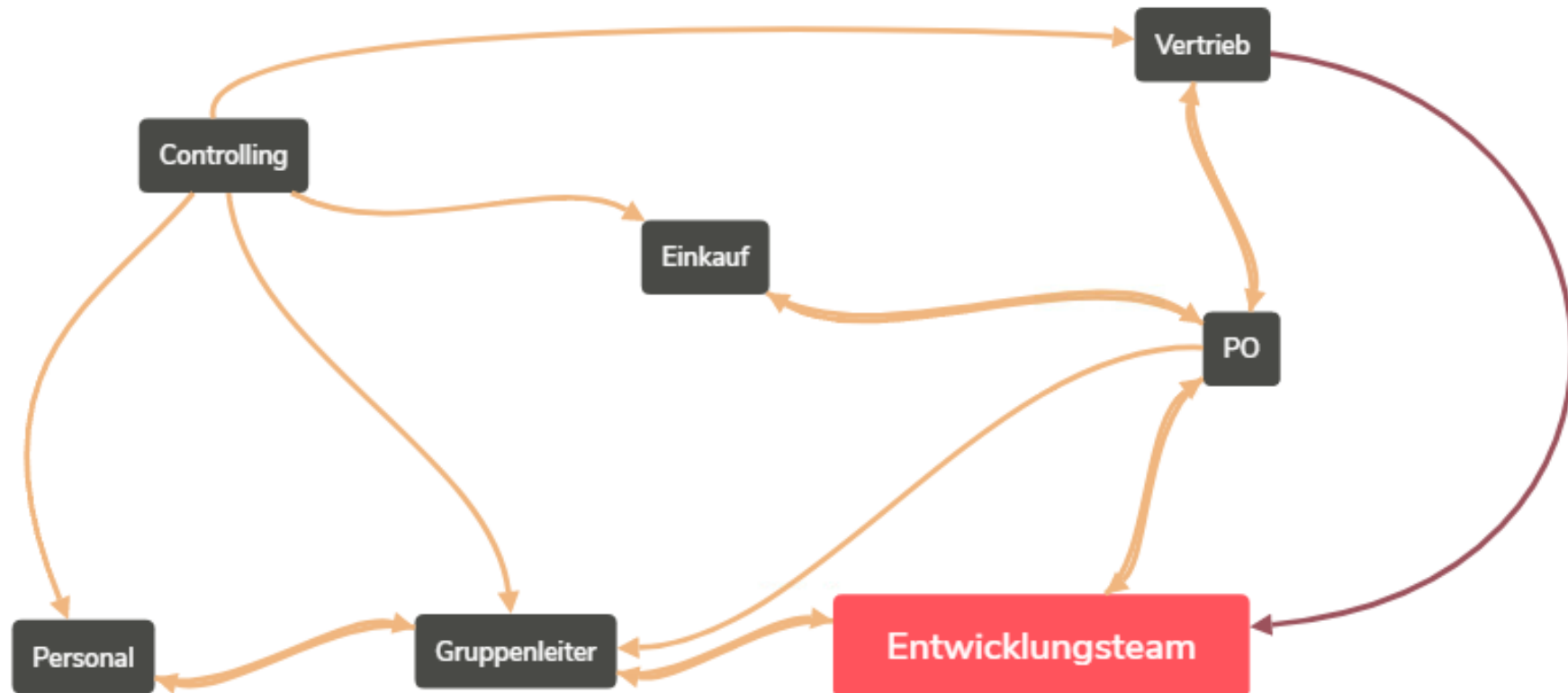
Do You Need a \$400 Juicer?



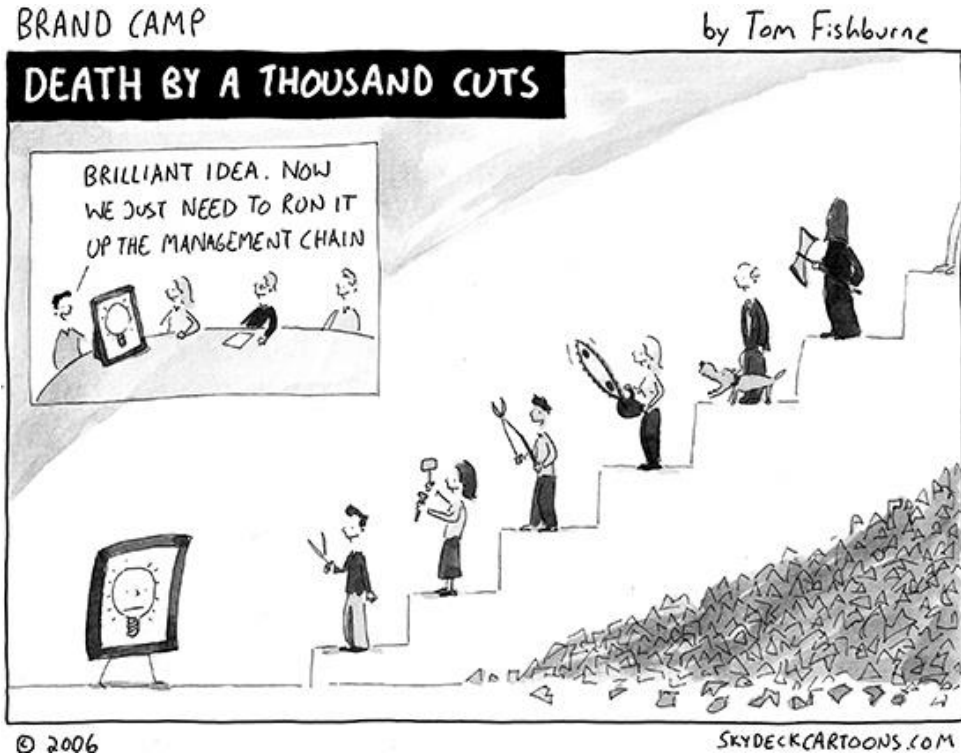
# Engineers gone wild



# Stakeholder Map



# Wichtige Punkte



- Wann kann mit der Umsetzung eines Features begonnen werden?
  - Gibt es einen Anforderungsprozess?
  - Definition of Ready?
- Wann ist ein Feature „fertig“?
  - Definition of done?
  - Gibt es einen Testprozess?
- Sind Rollen und Verantwortlichkeiten geklärt?
  - Global Code Ownership vs. Inselwissen
  - Architekt vs. Lead Developer vs. Autonomes Team
- Gibt es definierte Prozesse für Test und Deployment?
- Wie viele Entscheidungsträger gibt es?

# HINWEISE



**Saxonia Systems**  
So geht Software.

## Hinweise

Erst das **Ziel**, dann die **Umsetzung**.



**Saxonia Systems**  
So geht Software.

## Hinweise

Lieber **Freund** als **Feind**.



**Saxonia Systems**  
So geht Software.

## Hinweise

Die richtigen **Leute** zur richtigen **Zeit**.



**Saxonia Systems**  
So geht Software.

**Weniger** ist nicht immer **mehr**.





**Metriken** sind nur **Indikatoren**.



Lieber sinnvolle **Muster** als **Aktualität**.



# Der Sprecher



## Hendrik Lösch

Senior Consultant & Coach

[Hendrik.Loesch@saxsys.de](mailto:Hendrik.Loesch@saxsys.de)

[@HerrLoesch](#)

[Hendrik-Loesch.de](http://Hendrik-Loesch.de)



LinkedIn Learning



### ReSharper lernen

Effektiver programmieren mit der Erweiterung zu Visual Studio

[Hendrik Lösch](#)



### WPF-Anwendungen mit MVVM und Prism

Modulare Architekturen verstehen und umsetzen

[Hendrik Lösch](#)



### Windows 8 Store Apps mit MVVM und Prism

XAML-Entwurfsmuster, Bootstrapping, Navigation, Messaging

[Hendrik Lösch](#)



### LINQ Grundkurs

Wichtige Spracheigenschaften von C#, Joins, Variablen und andere Operationen, erweiterte Techniken

[Hendrik Lösch](#)



### Grundlagen der Programmierung: Test Driven Development

Business-Applikationen testgetrieben entwickeln

[Hendrik Lösch](#)



### Inversion of Control und Dependency Injection - Grundlagen

Prinzipien der modernen Software-Architektur ...

[Hendrik Lösch](#)



### C#: Test Driven Development

Grundlagen, Frameworks, best Practices

[Hendrik Lösch](#)



### Grundlagen der Programmierung: Codemetriken

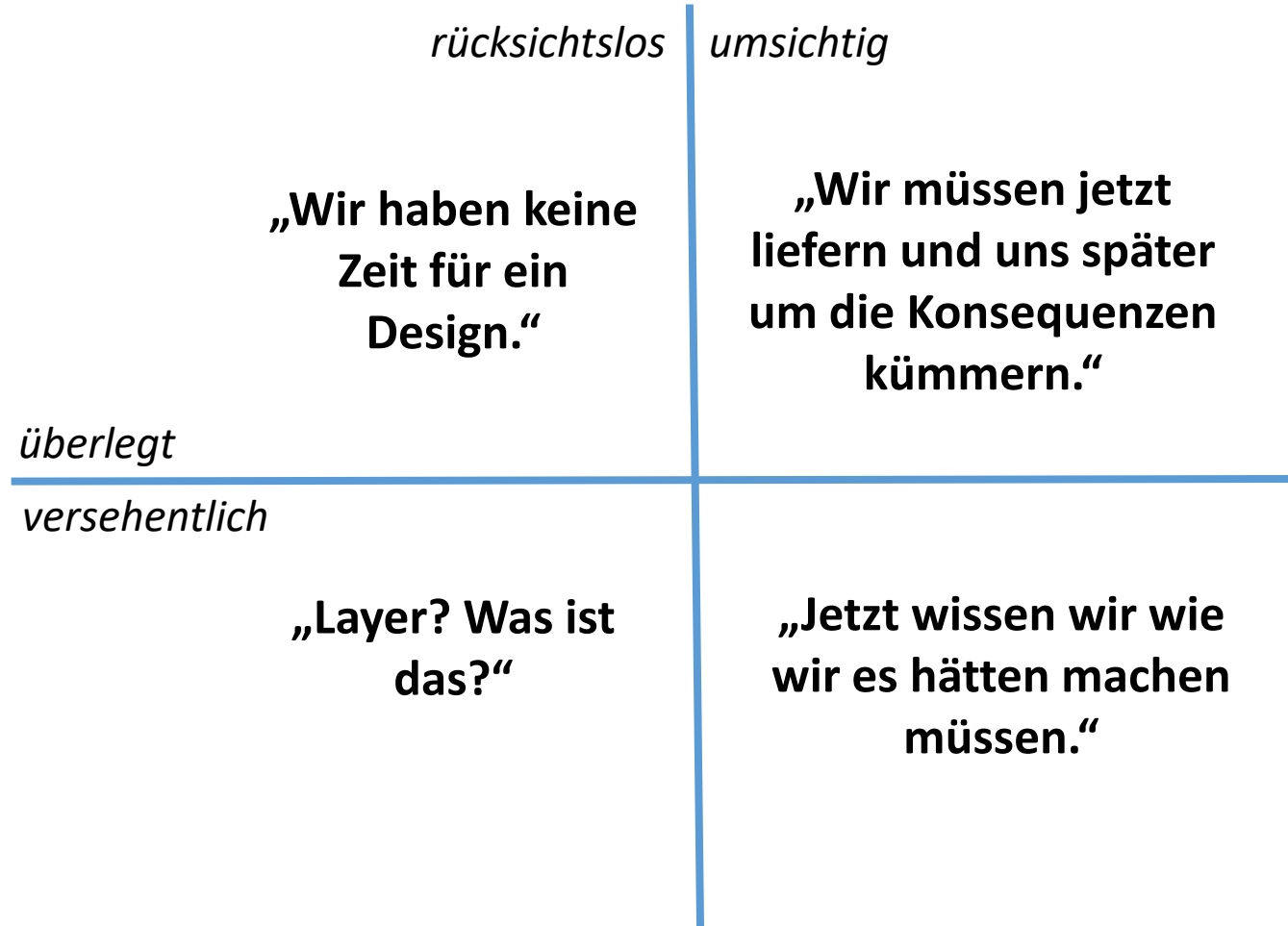
Softwarequalität einschätzen, sicherstellen und ...

[Hendrik Lösch](#)



**Saxonia Systems**  
So geht Software.

# Technische Schuld



Ward Cunningham



**Saxonia Systems**  
So geht Software.

# Horseshoe Process Model

