inosca

Die kantonale Open Source Community

Adfinis

Christian Zosel

Software Engineer

Adfinis

Bern, Switzerland

christian.zosel@adfinis.com

https://linkedin.com/in/czosel


Adfinis

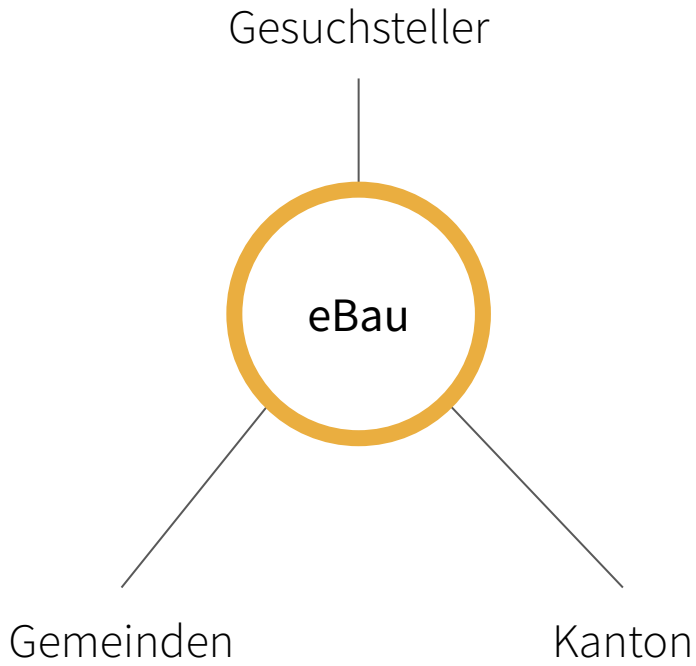
Vorstellung Adfinis

Wir sind ein Team von über **80 Open-Source-Experten**, die Sie bei Ihren technischen Herausforderungen unterstützen

Wir sind Experten in der Arbeit für stark regulierte Branchen und konzentrieren uns **auf Qualität** statt Quantität

Wir teilen unser Wissen und befähigen Ihr Team durch unser Inner Source Mindset

inosca Community



Inhalt

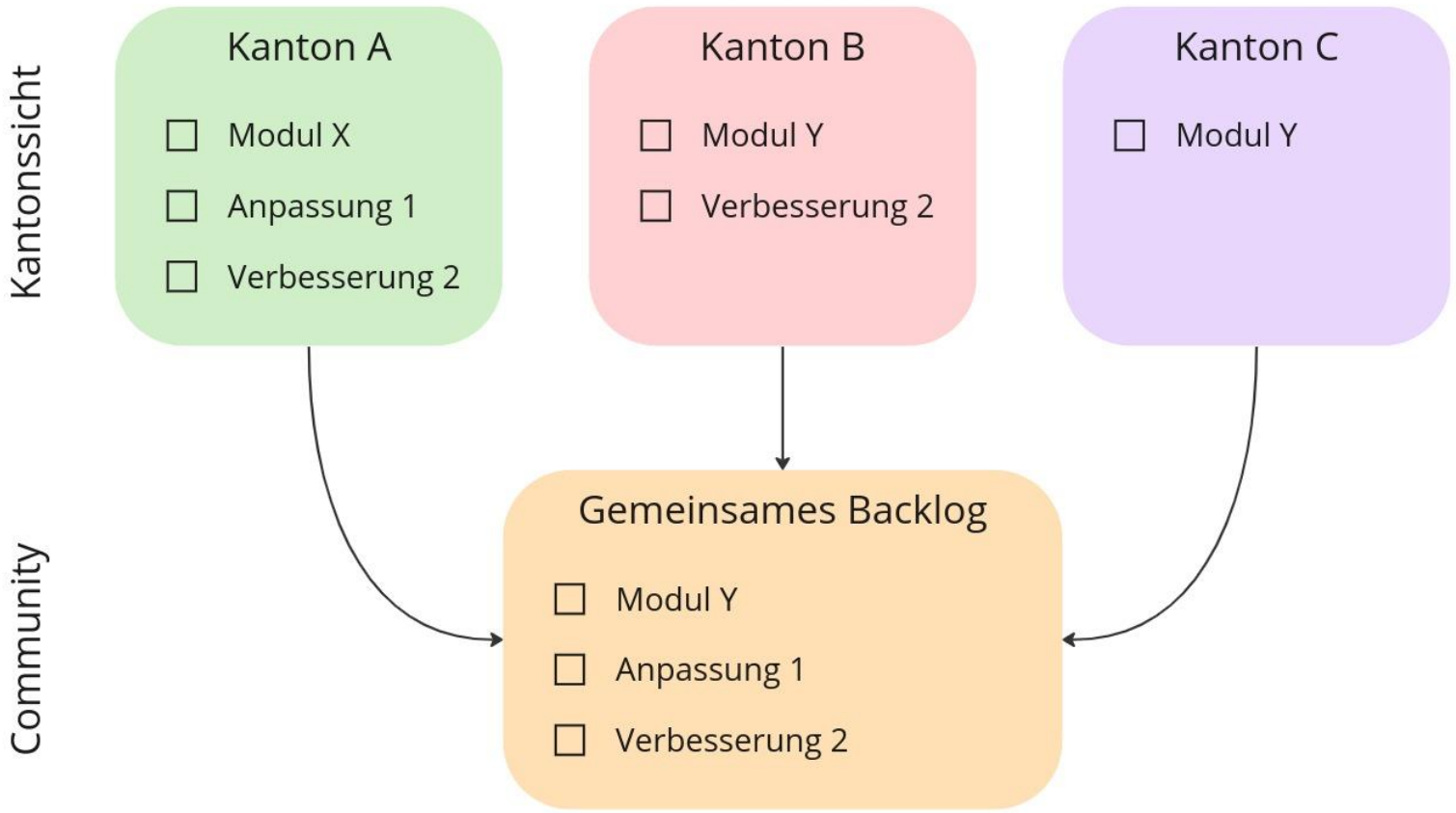
- Organisation
- Zusammenarbeit
- Architektur
- Betrieb
- Dienstleistersicht

Organisation

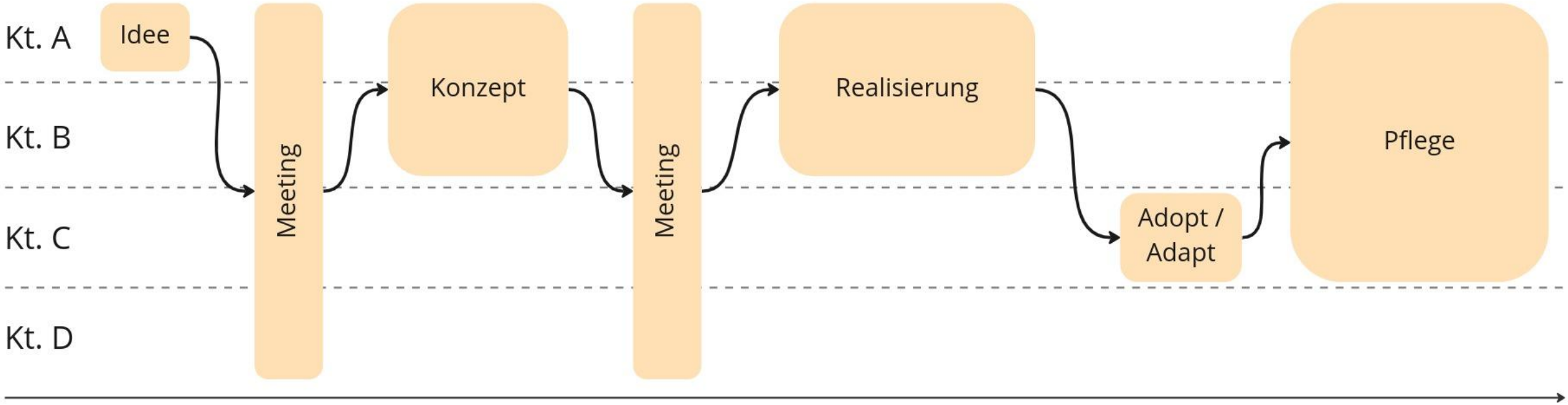
Organisation

Monatliche
Online-Meetings (2.5h)

- Genereller Austausch
- Gemeinsames Change-Management



Lifecycle eines Features



Spielregeln

- Entwicklungsergebnisse sind frei verfügbar
- Wer spezifiziert, finanziert
- Kostenschlüssel

inosca.ch
Bewilligungsprozesse
UR, SZ, BE, SO

Interkantonale Entwicklungsgemeinschaft für elektronische

inosca

Statement Stand 2.11.2022 / Version 5

Einführung

inosca.ch ist eine interkantonale Software-Entwicklungsgemeinschaft für elektronische Bewilligungsprozesse (in=innovativ, os=open source, ca = canton). Sie wurde im Jahre 2020 von den Kantonen Uri, Schwyz, Bern und Solothurn gegründet. Zuvor bestand eine mehrjährige Zusammenarbeit unter anderem Namen. Mit dem vorliegenden Statement regelt die Gemeinschaft die Mechanismen der Zusammenarbeit.

Strategie / Ziele

Aktuell werden unter dem Label von inosca verschiedene Software Varianten eingesetzt. Dieser Zustand ist vorübergehend. Langfristig wird für alle Projekte innerhalb der Gemeinschaft inosca eine einheitliche technische Basis angestrebt (Roadmap siehe Anhang). Dadurch können Synergien genutzt werden, die sich sowohl inhaltlich als auch finanziell positiv auf die Projekte auswirken. Ziel ist es einerseits, dass alle Mitgliederkantone von inosca einheitliche Technologien verwenden (z.B. Keycloak / CALUMA). Andererseits soll angestrebt werden, dass die Kantone künftig in möglichst vielen Bereichen (Formular, Workflow, Module, Layout usw.) auch die gleichen Applikationen und Prozesse verwenden und ein gemeinsames Changemanagement anstreben.

Dank dem Grundsatz, künftig Open Source Technologien einzusetzen, die für jeden Dienstleister frei zugänglich sind, besteht die Möglichkeit, innerhalb der Gemeinschaft inosca.ch mehrere Dienstleister zu berücksichtigen.

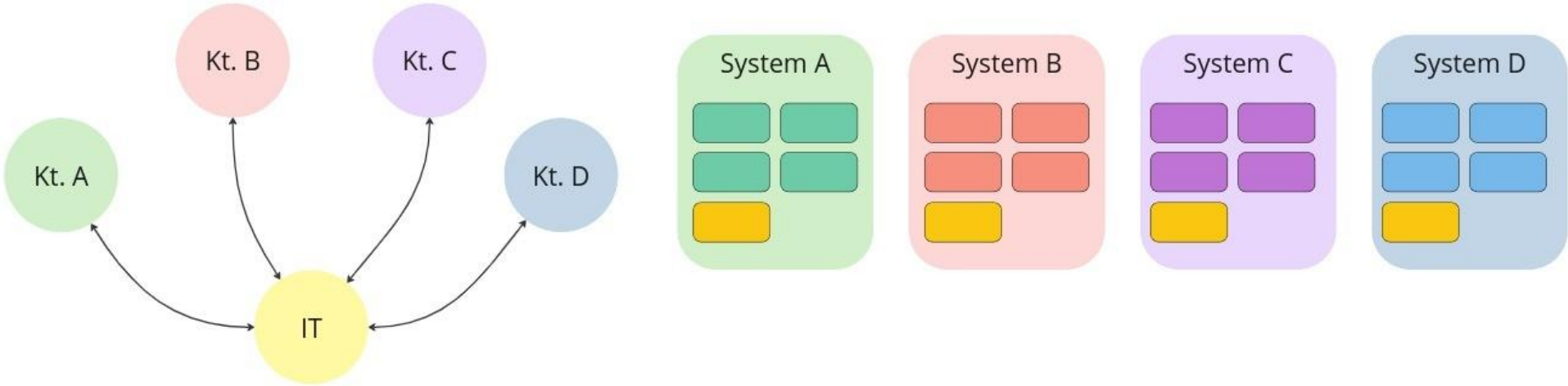
Die Teilnehmer an inosca wirken bei der Verbreitung von Informationen über die Gemeinschaft und deren Strategie proaktiv mit, indem sie diese interessierten Kantonen vorstellen. Sie engagieren sich, die Gemeinschaft inosca und deren Philosophie im eigenen Interesse und im Interesse aller beteiligten Kantone mit

Zusammenarbeit

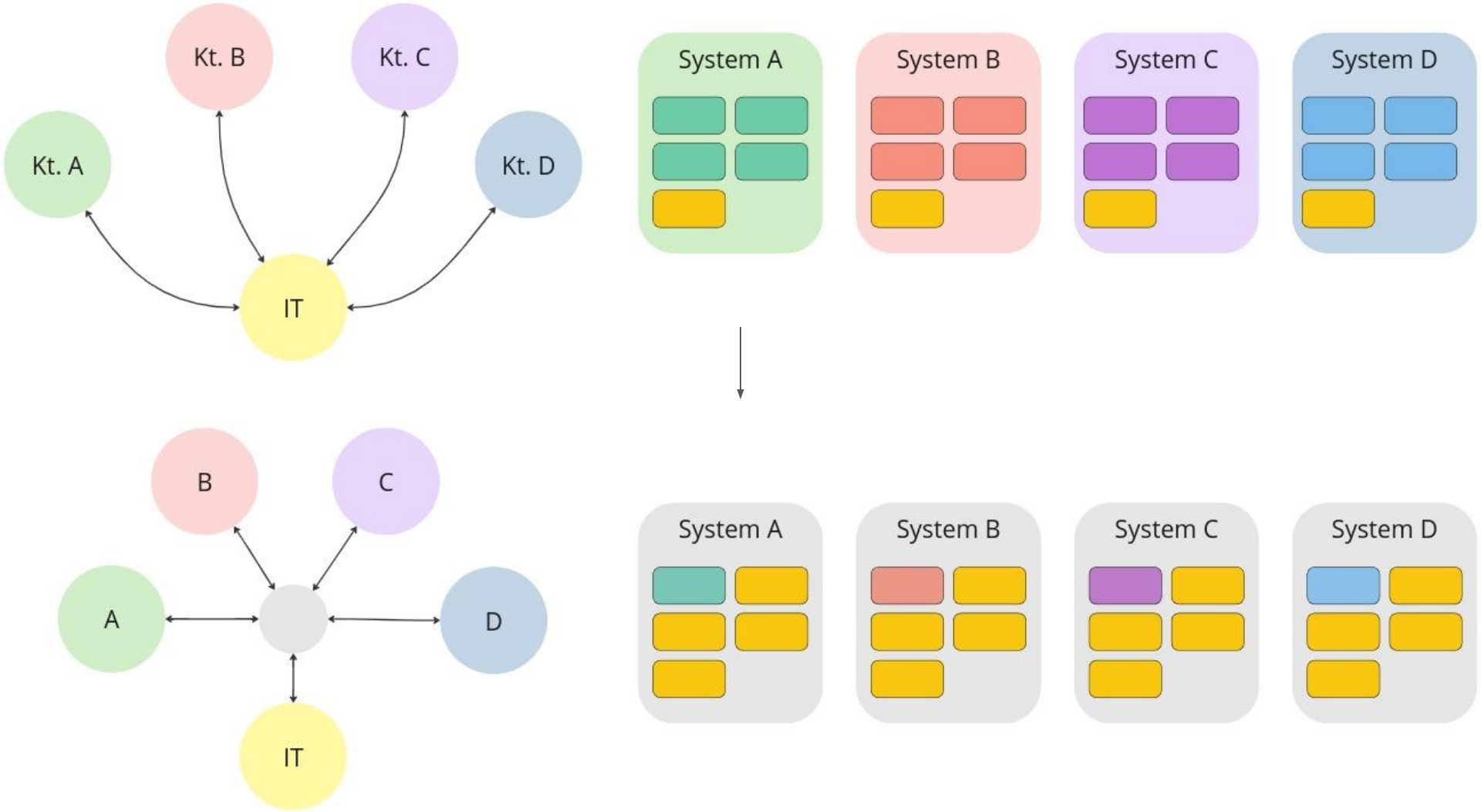
Gesetz von Conway

> Any organization that designs a system will produce a **design** whose structure is a **copy** of the **organization's communication structure**.
>
> – Melvin E. Conway

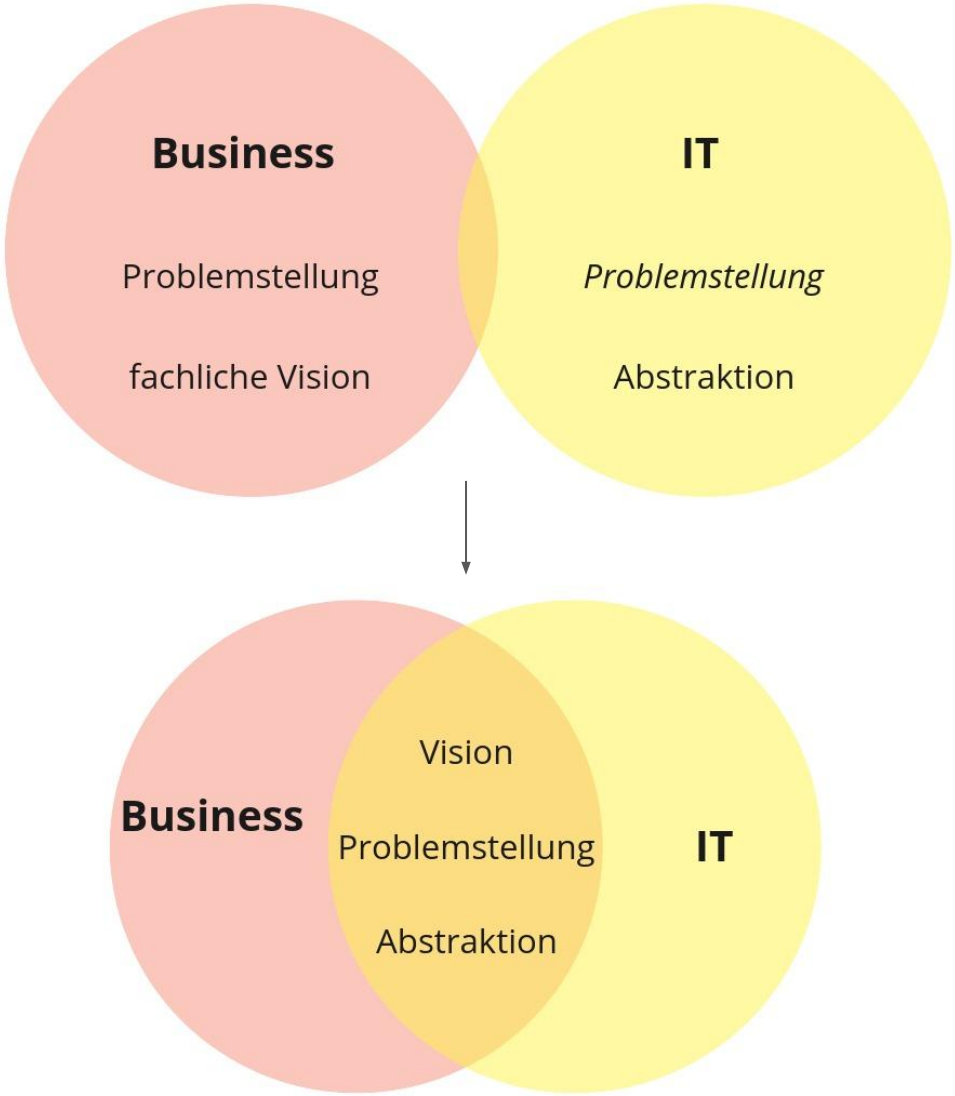
Gesetz von Conway



Gesetz von Conway



Zusammenarbeit Business und IT



- Technisches vs. fachliches Know-how: Horizonterweiterung auf beiden Seiten
- Gemeinsame Sprache finden
- Austausch auf Augenhöhe, Transparenz
- Vertrauen und gegenseitige Wertschätzung

Architektur

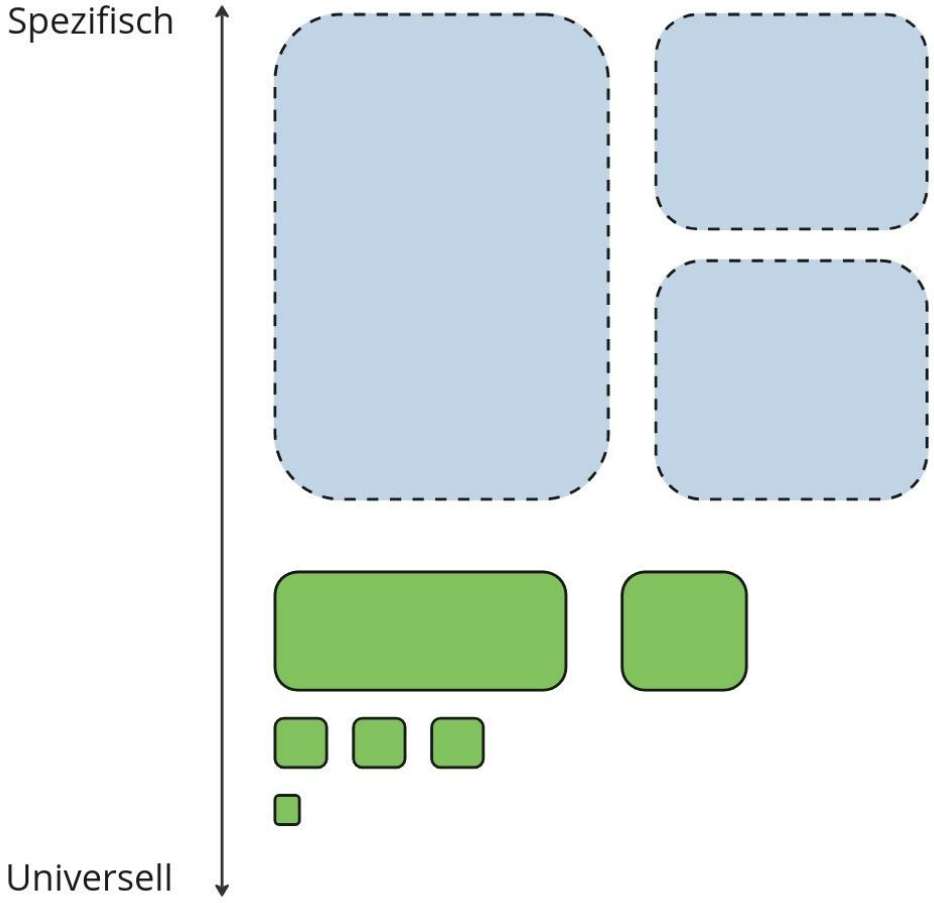
Open Source Architektur

Maintainer

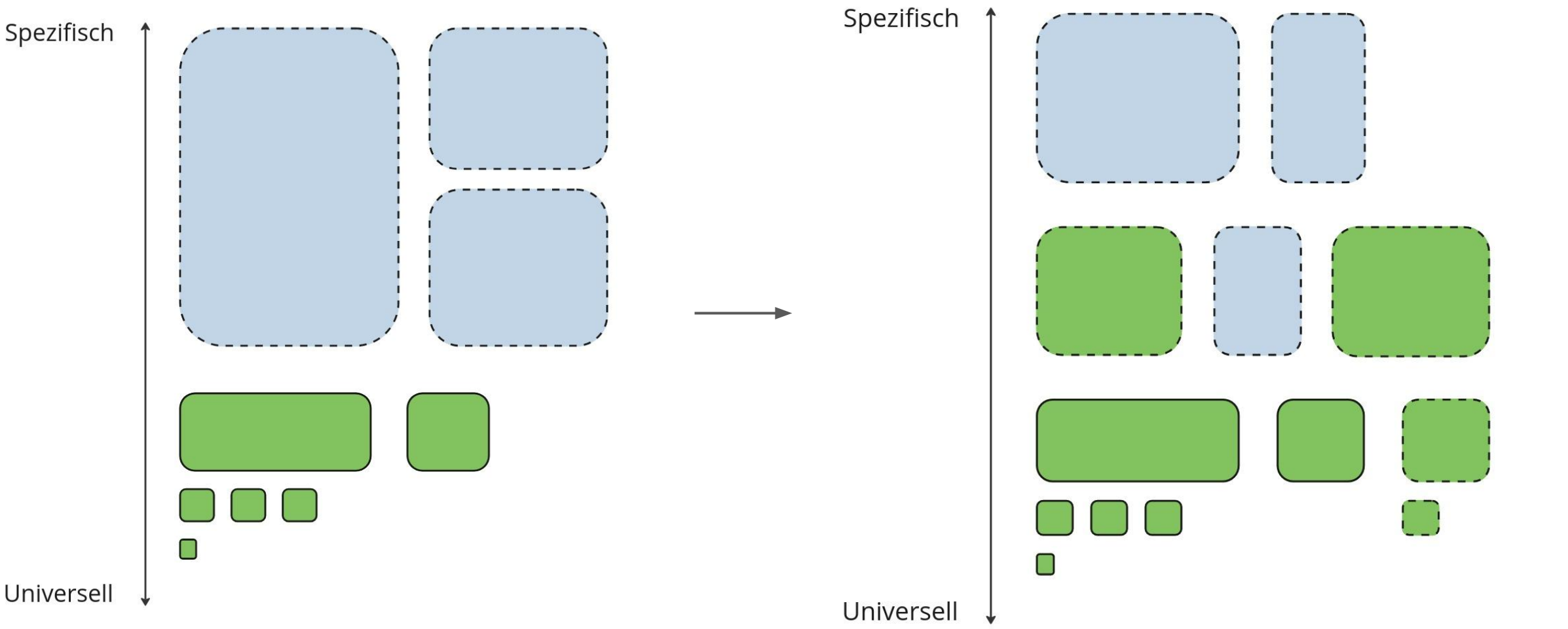
Consumer

Closed

Open



Open Source Architektur



inosca Technologiestack

- Programmiersprachen
- Frameworks
- Storage
- Weiteres

High Level Open Source Komponenten

caluma

- Formulare
- Workflows

Alexandria

- Dokumentenmanagement

Document Merge Service

- Vorlagen-Generierung

Emeis

- Benutzer- und Berechtigungsverwaltung

Gemeinsame Basis, individuelle Ausgestaltung

Kantonsspezifische Konfigurationen

inosca

Standardisierte Fachapplikation

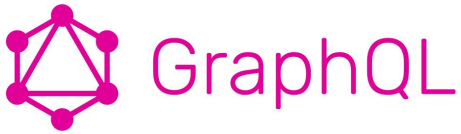
caluma

Generisches Framework

Offene Standards

- Authentifizierung

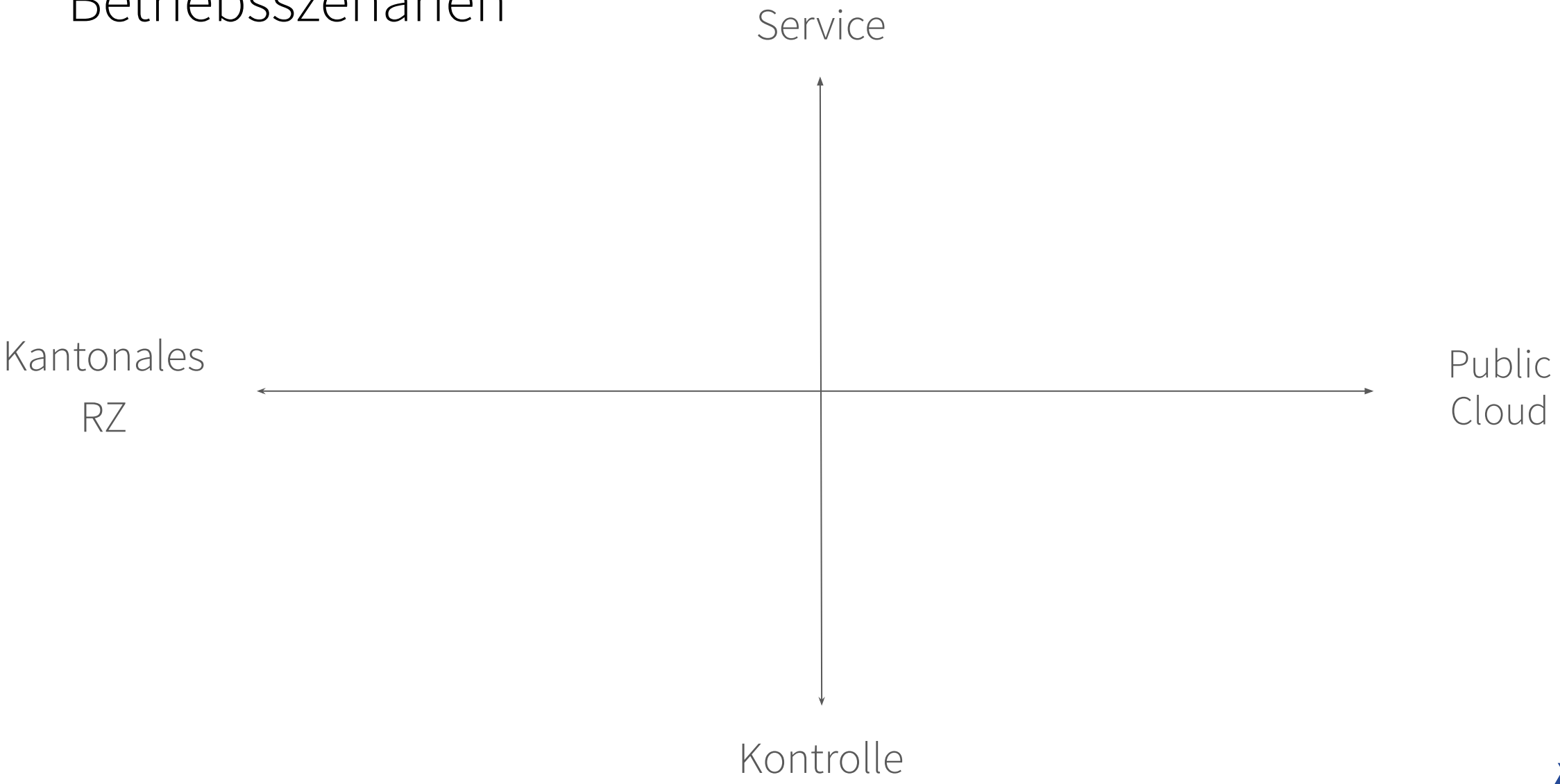
- Kommunikation Frontend - Backend



- Schnittstellen
  (Gemeindesoftware, GWR)

Betrieb

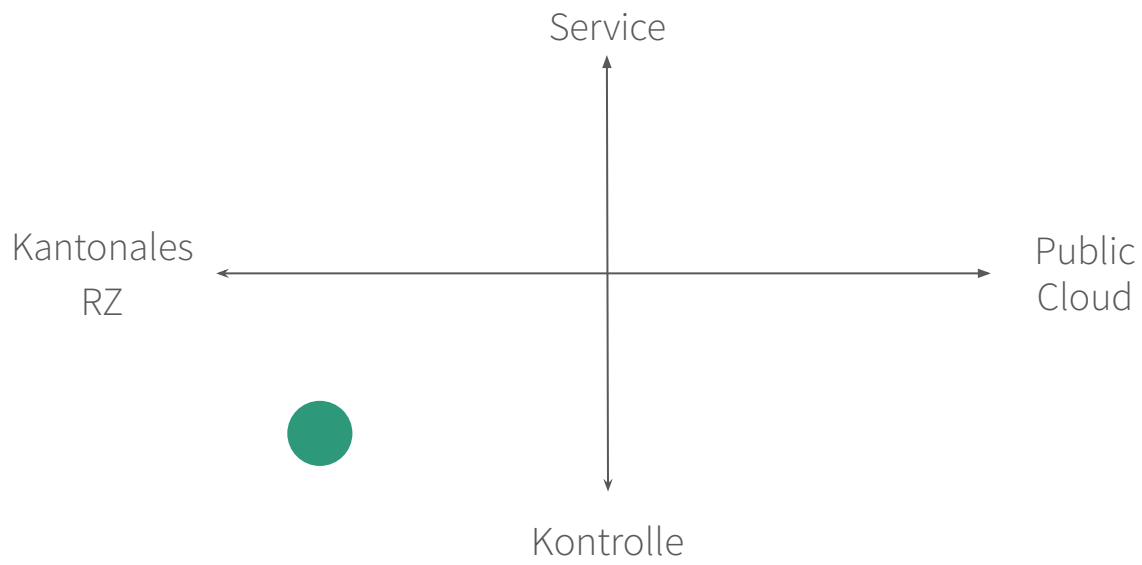
Betriebsszenarien



Szenario 1: Betrieb durch Kanton, volle Kontrolle

Kantonale Informatik

- Hardware, Netzwerk
- Monitoring + Backup
- OS Updates
- Einspielen neuer Releases



Dienstleister

- Ausliefern neuer Releases

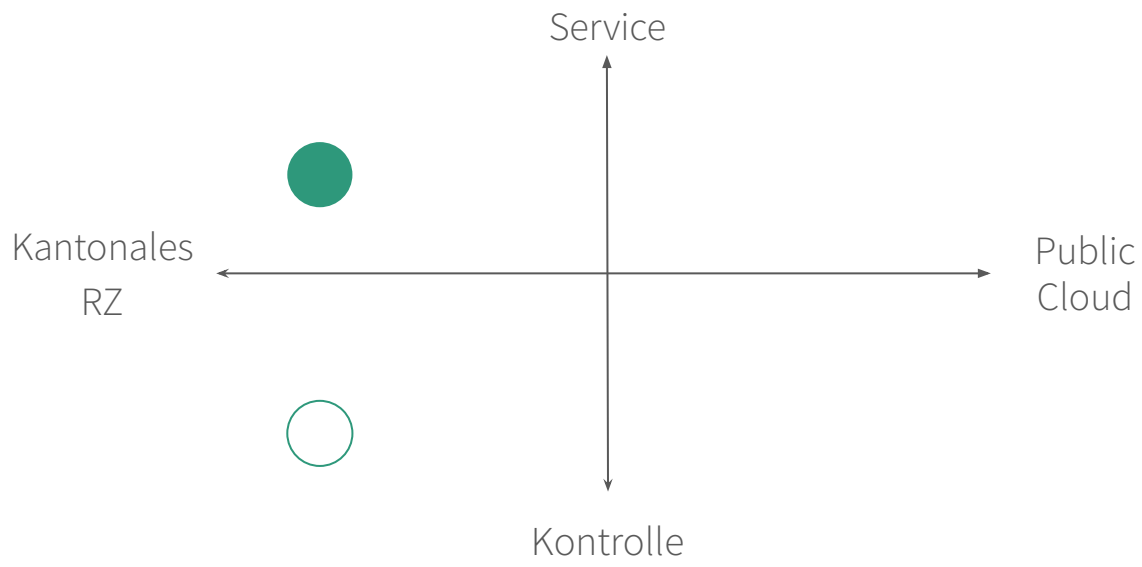
Szenario 2: Betrieb durch Kanton, Pflege durch Dienstleister

Kantonale Informatik

- Hardware, Netzwerk

Dienstleister

- Releases
- OS Updates
- Monitoring + Backup



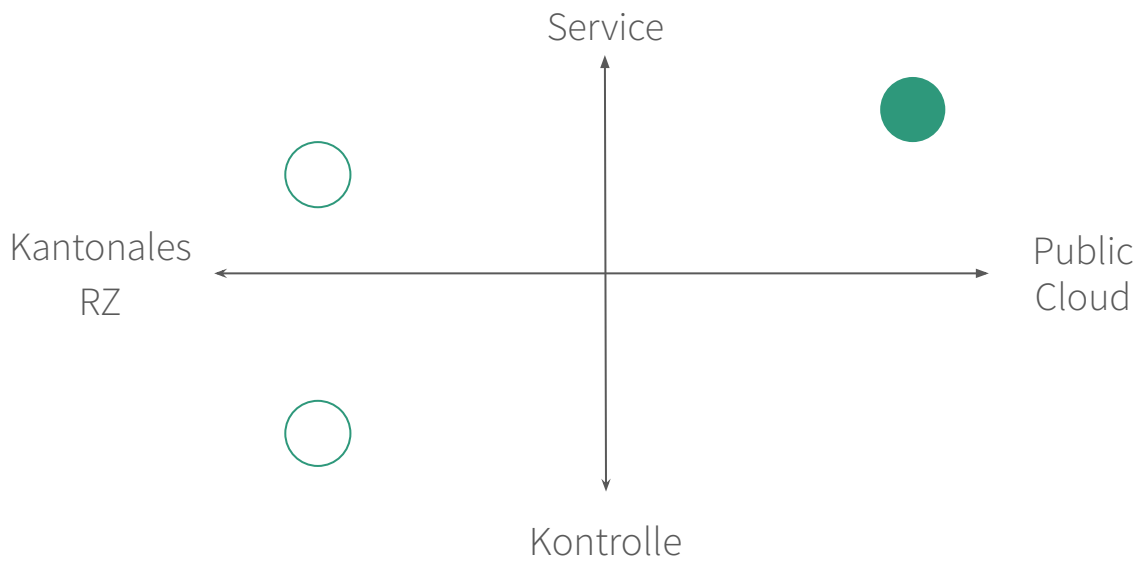
Szenario 3: Betrieb in Cloud, Pflege durch Dienstleister

Cloud Provider

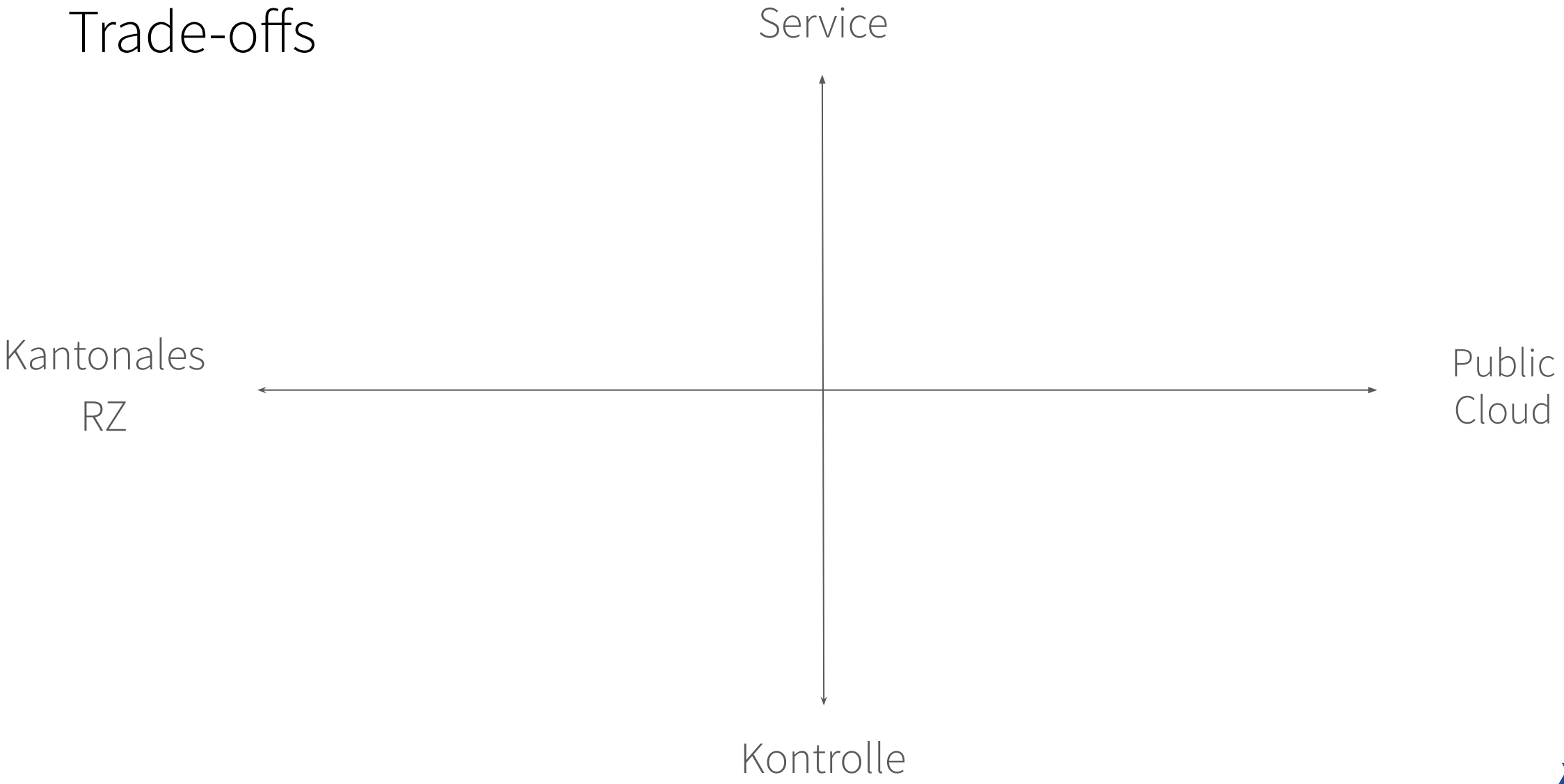
- Hardware, Netzwerk
- Basisdienste (Datenbank, Storage, Kubernetes Cluster)

Dienstleister

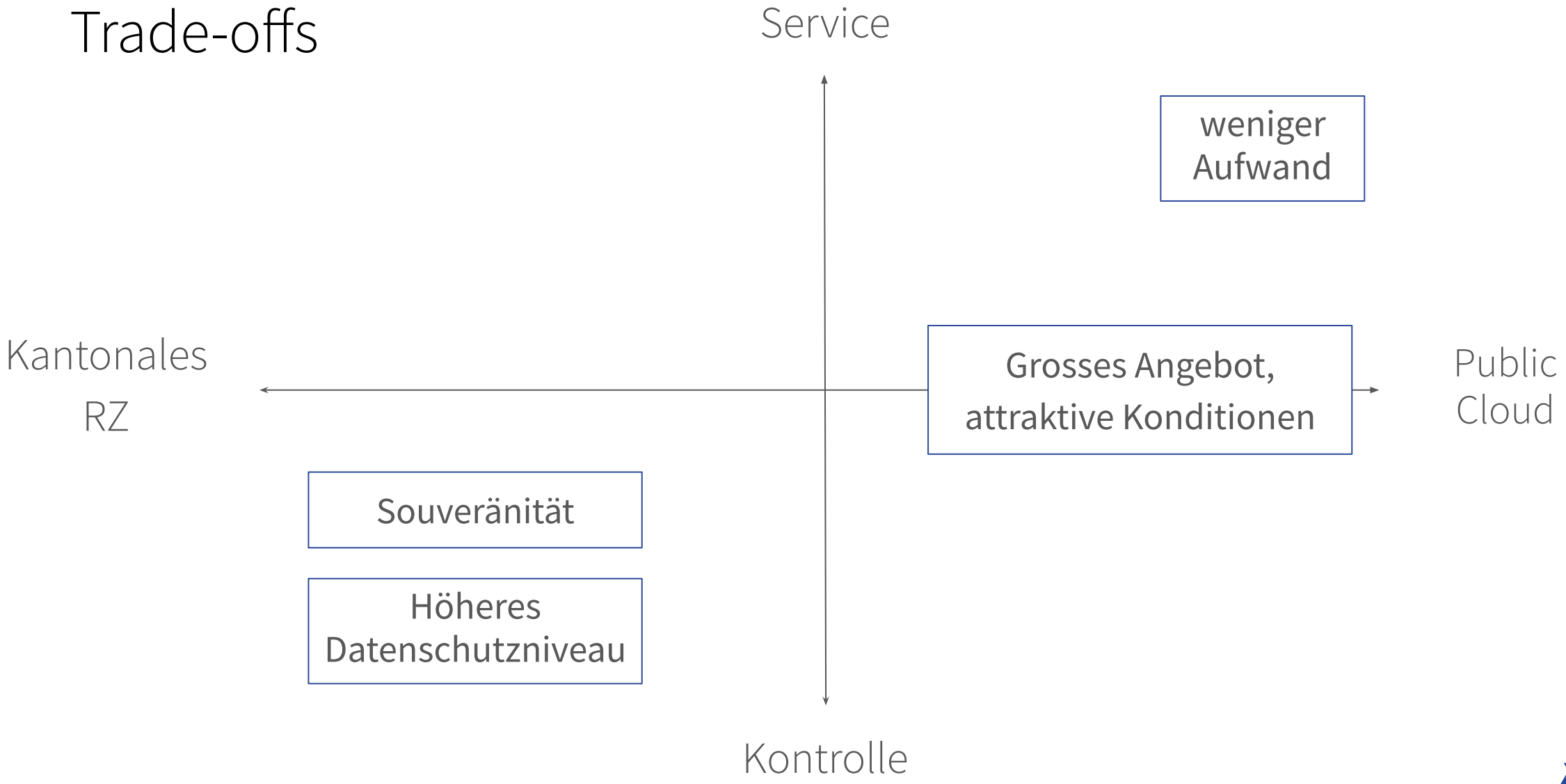
- Releases
- Monitoring + Backup
- OS Updates (falls nötig)



Betriebsszenarien: Trade-offs



Betriebsszenarien: Trade-offs



Dienstleistersicht

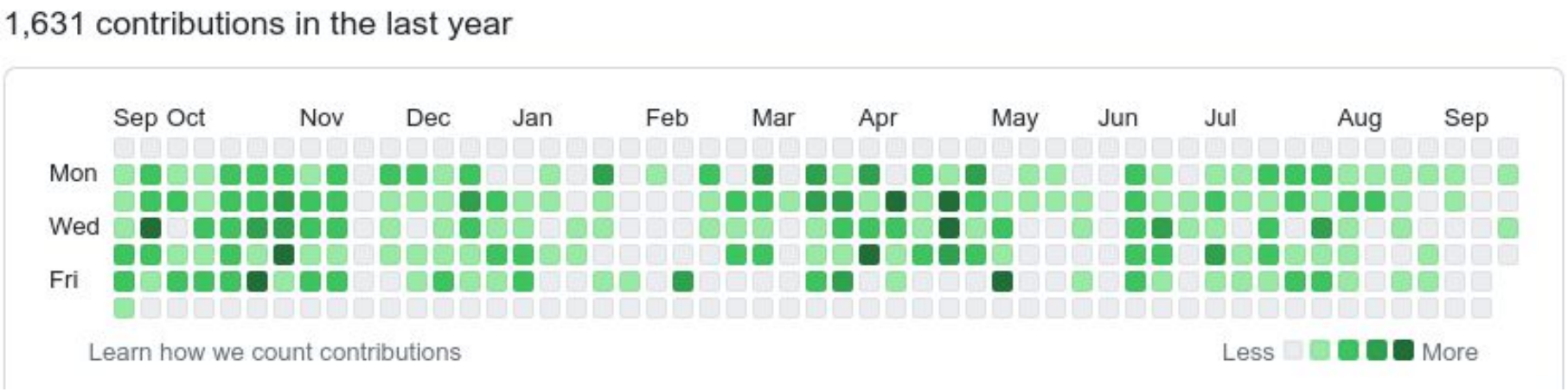
Geschäftsmodell

- **Integration**, Support und Wartung, Hosting
- Kaum Eigeninvestition in Produktentwicklung
- IP gehört dem Kunden → Nutzungsrechte dank Open Source Lizenz
- Kompetitiver Vorteil durch Know-How und Kundenbeziehung (statt Lock-In)

Quelle: Getty Images/iStockphoto

Vorteile aus Dienstleistersicht

- Effizienzsteigerung, Synergien zwischen Projektteams
- Produktifizierung
- Code-Qualität, Architektur
- Arbeitgeberattraktivität
  - Portfolio
  - Portabilität von Fähigkeiten
  - Nachhaltigkeit, «Zurückgeben»



Fazit

Open Source Mindset

- **Flexible** Organisation und Betrieb
- **Gemeinschaftliche** Zusammenarbeit auf Augenhöhe
- **Interoperable** Architektur
- **Offenheit** für organisationsübergreifende Zusammenarbeit
- **Transparente** Kommunikation

- Beste Ergebnisse, wenn «Open Source Mindset» von allen Beteiligten gelebt wird!

Vorteile von Open Source

Aus **Kantonssicht**:

- Kostenersparnis
- Fachliches Know-How, Erfahrungen
- Reduzierter Vendor Lock-In
- Transparenz und Nachhaltigkeit

Aus **Dienstleistersicht**:

- Projektübergreifende Synergieeffekte
- Effizienz
- Höhere Qualität
- Arbeitgeberattraktivität

Stay in Touch

inosca

inosca.ch

Adfinis

adfinis.com

Anhang

Open Source

inosca / ebau

Type / to search

Code | Issues 1 | Pull requests 1 | Actions | Projects | Wiki | Security 31 | Insights | Settings

ebau

Public

Edit Pins | Unwatch 2 | Fork 4 | Starred 9

main | 2 branches | 0 tags | Go to file | Add file | Code

README.md

Electronic building permit application for Swiss cantons.

Table of Contents

About

Electronic building permit application for swiss cantons.

inosca.ch/

switzerland | egovernment | building-permits

Readme

EUPL-1.2 license

Activity

9 stars

2 watching

4 forks

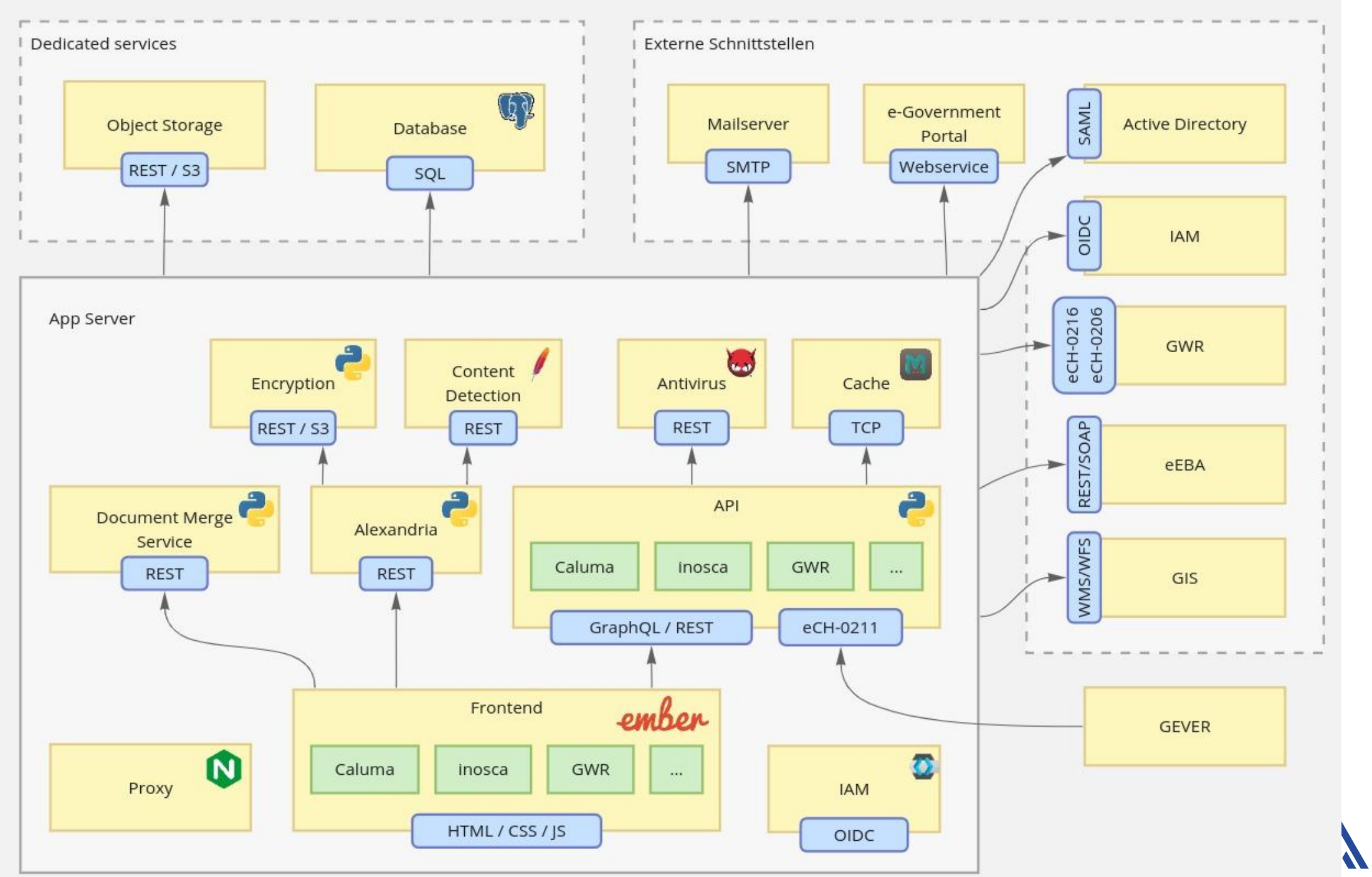
Report repository

Contributors 15

EUPL-1.2 Lizenz

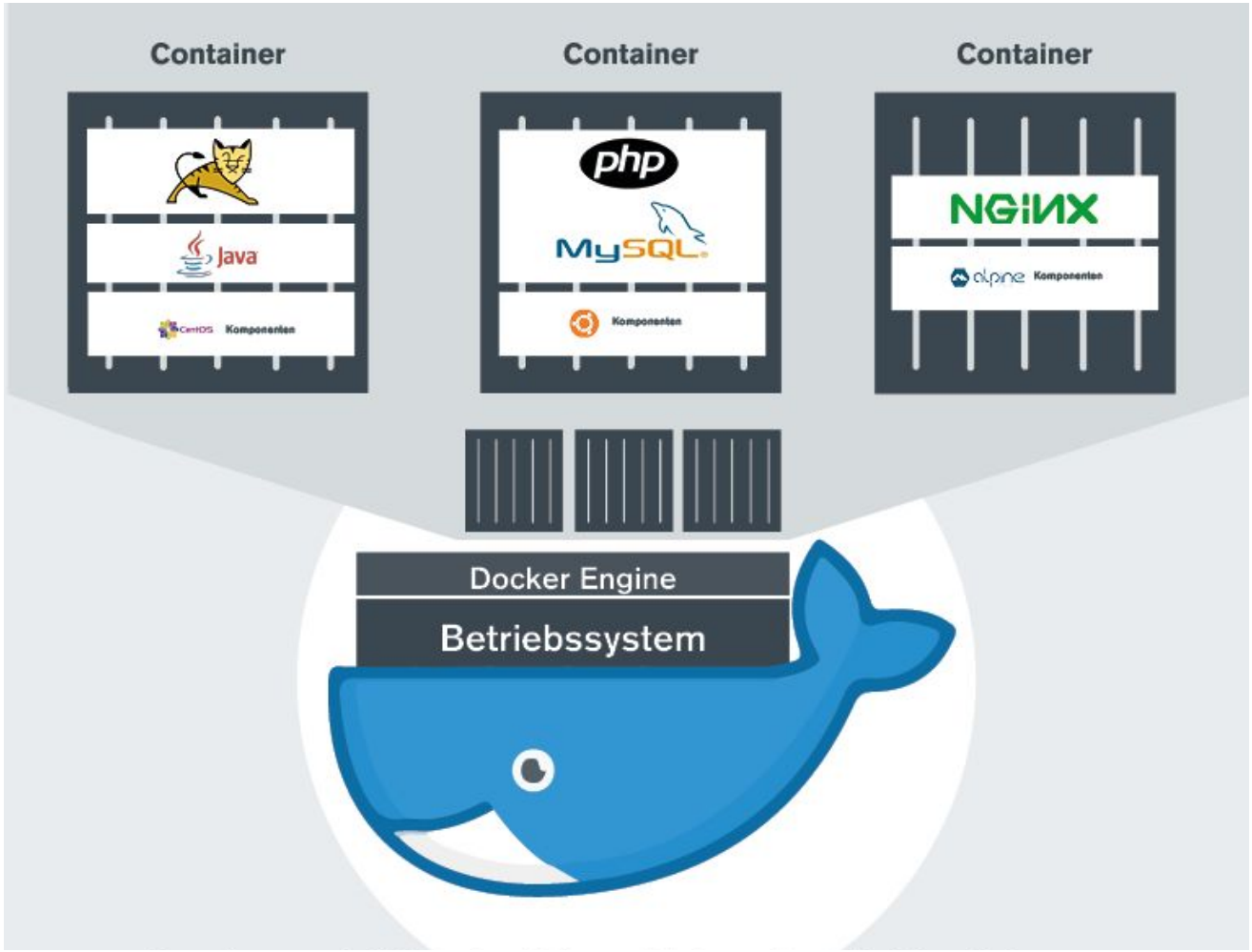
- Copyleft-Lizenz
- Übersetzt in 23 Amtssprachen der EU
- Siehe auch Leitfaden Lizenzwahl, Kanton Bern

Architektur


Dedicated services
Object Storage
REST / S3
Database
SQL
Externe Schnittstellen
Mailserver
SMTP
e-Government Portal
Webservice
SAML
Active Directory
OIDC
IAM
eCH-0216
eCH-0206
GWR
REST/SOAP
eEBA
WMS/WFS
GIS
App Server
Encryption
REST / S3
Content Detection
REST
Antivirus
REST
Cache
TCP
Document Merge Service
REST
Alexandria
REST
API
Caluma
inosca
GWR
...
GraphQL / REST
eCH-0211
Frontend
ember
Caluma
inosca
GWR
...
HTML / CSS / JS
Proxy
IAM
OIDC
GEVER

Container

Container



- Effizient: Leichtgewichtige Form der Virtualisierung
- Portabel: Enthalten sämtliche Abhängigkeiten
- Sicher: Laufen isoliert voneinander

Quelle: claranet.de

Container Orchestrierung

- Ausfallsicherheit
  - Bei Ausfall einer Festplatte, eines Servers, eines Rechenzentrums
  - Unterbruchsfreie Deployments
- Automatisches Skalieren
- Automatisierung (CI/CD)

Einfach

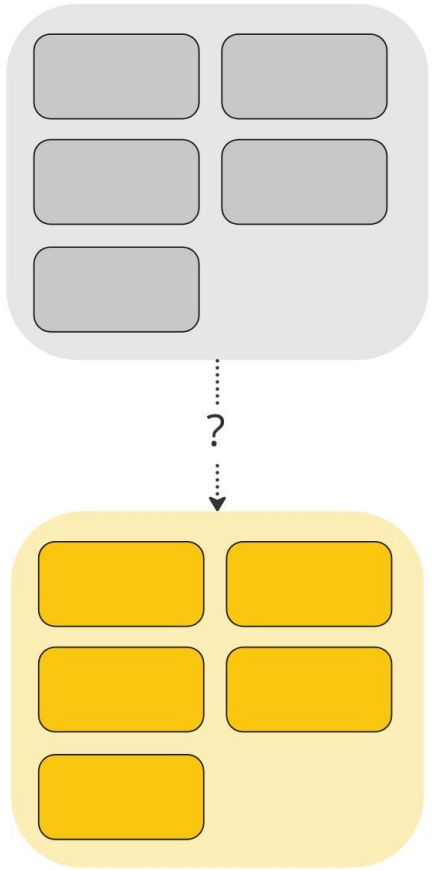
Vollumfänglich

Fallbeispiel: Modernisierung von eBau

Fallbeispiel: Modernisierung von eBau

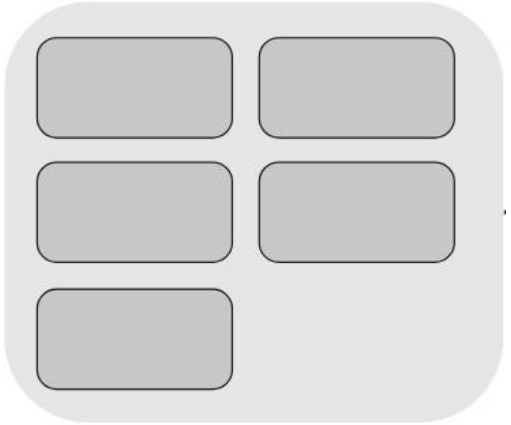
Unübersichtliche **Ausgangslage**

- Grosse Community
- Heterogene Lösungen
- Geringer Anteil gemeinsam genutzter Komponenten
- Keine einheitliche Vision



Big Bang / Rewrite

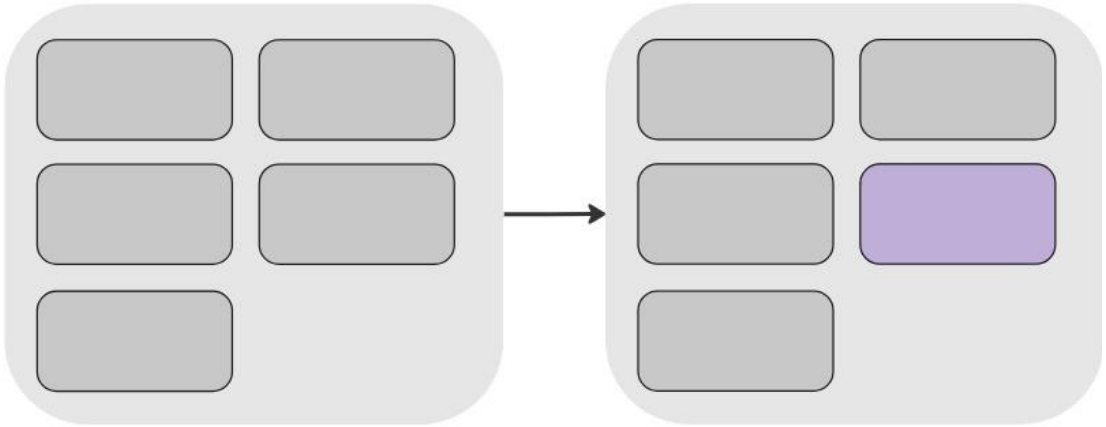
Alt



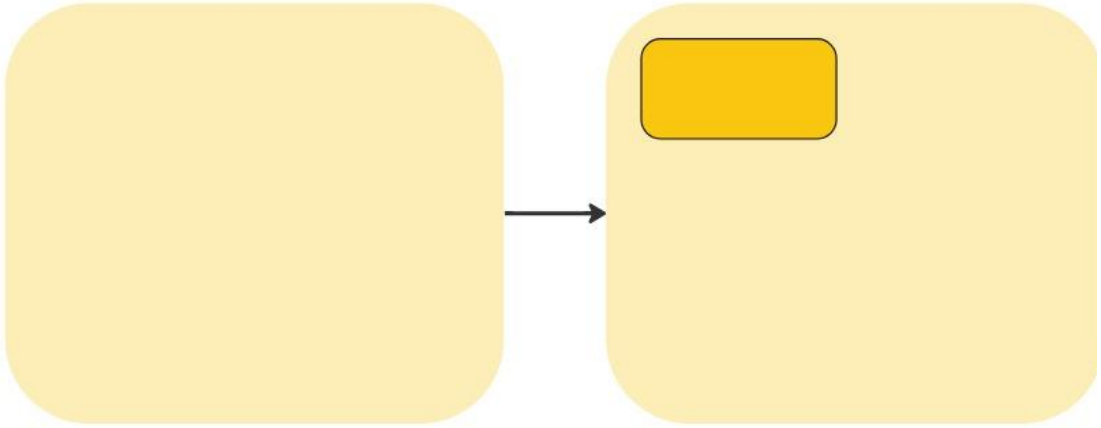
Neu

Big Bang / Rewrite

Alt

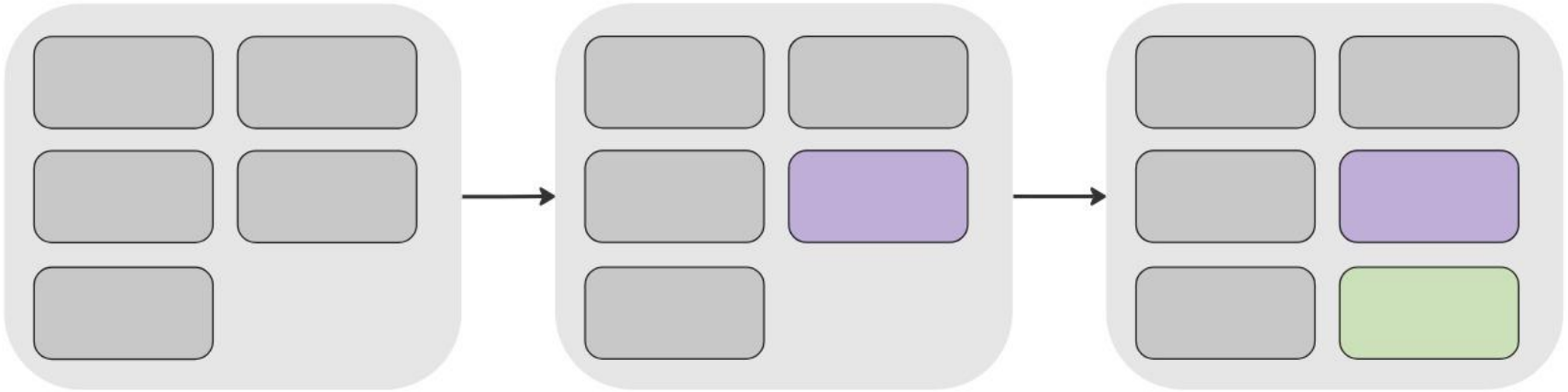


Neu

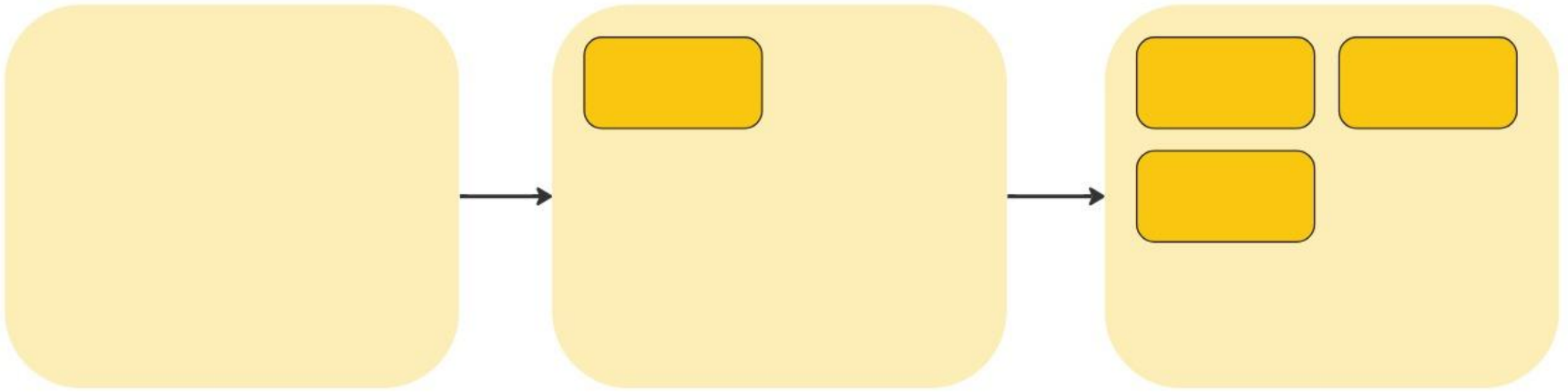


Big Bang / Rewrite

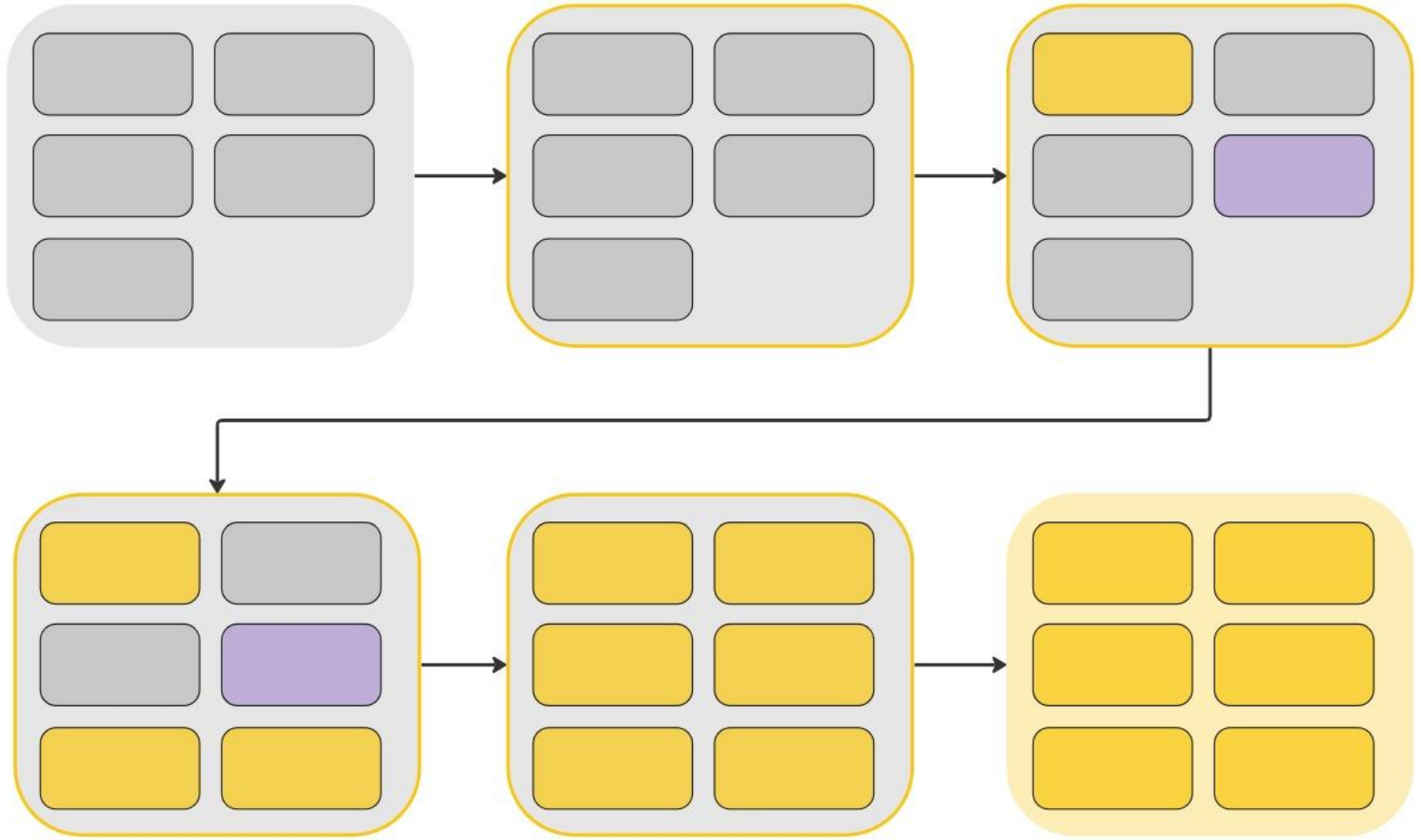
Alt



Neu



Inkrementell



Erkenntnisse nach der Modernisierung

Start simple

- Grundlagen legen (Architektur, PoC, Vision, …)
- Lösungsraum erkunden

Release early

- Feedback einholen
- Community aufbauen
- Lösung muss bzw. sollte noch nicht «fertig» sein