31.8.–3.9.2015
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Chicken Little, Cold Turkey etc.

Welche Migrationsstrategie ist die richtige?

Stephan Kaps

Bundesversicherungsamt

Migrationsstrategien

Stephan Kaps (info@kitenco.de)

@kitenco1

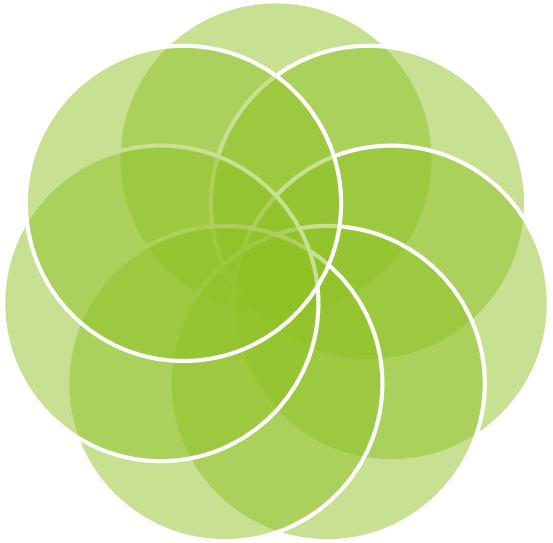
Chicken Little, Cold Turkey .. oder was?

Welche Migrationsstrategie ist die Richtige?

Stephan Kaps

Techn.
Projektleiter

Leidenschaft
sind neue
Technologien
und Methoden



Software-
Architekt

ISTQB, ISAQB,
IREB und ITIL
zertifiziert

JEE – SOA -
Host

Speaker &
Autor

Java seit 2002

Agenda

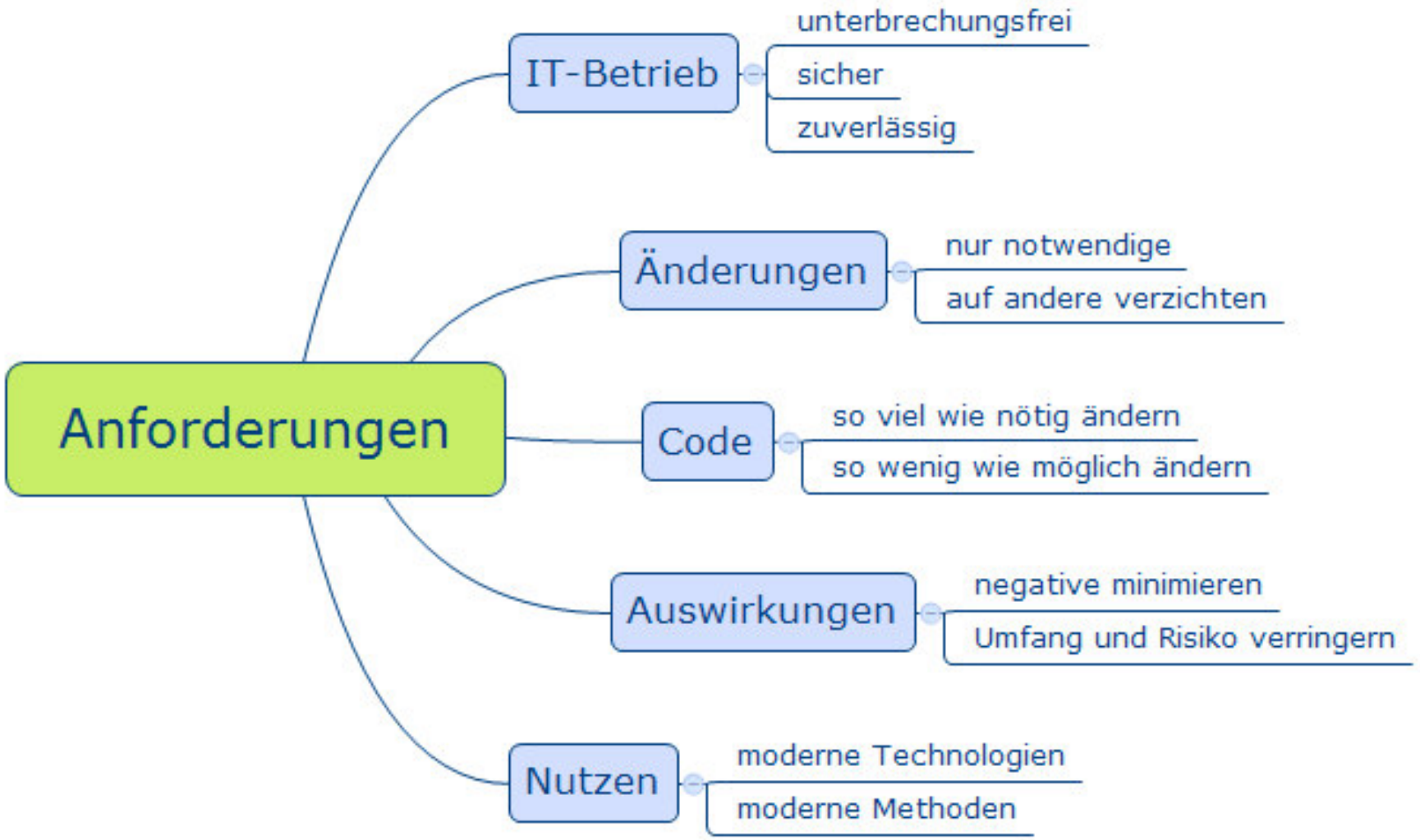
- Einleitung
- Migrationsstrategien
  - Big Bang Approach (Cold Turkey)
  - Chicken Little
  - Database First
  - Database Last
  - Composite Database Approach
  - Butterfly Methodology
- Weitere Ansätze
- Zusammenfassung

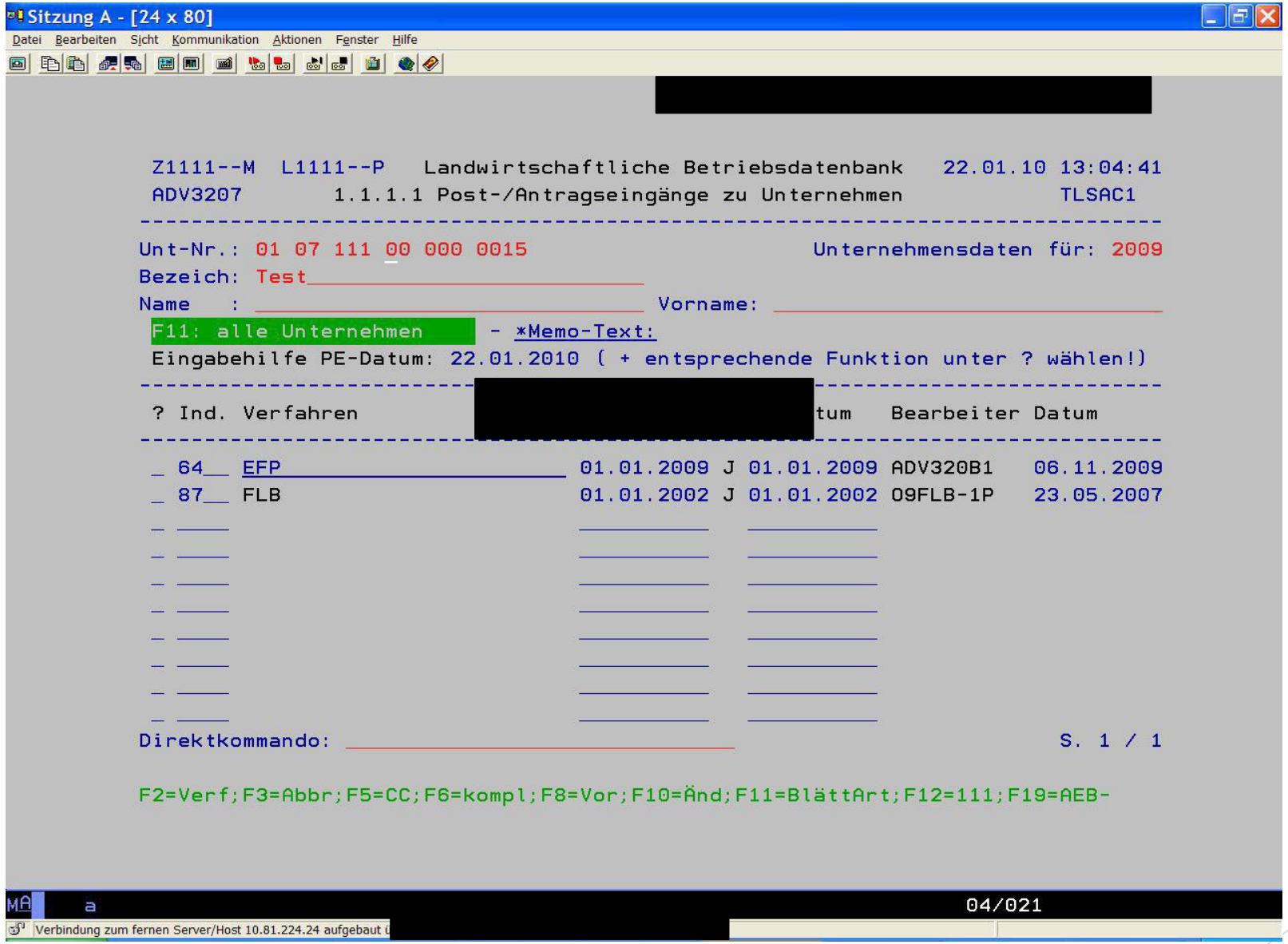
Einleitung

- Unter Migration versteht man
  - Jegliche Art von Umstellung
- Migrationen sind
  - rein technische Transformationen
  - mit einer klaren Anforderungsdefinition
- Zählen zum Software-Reengineering
- Ziel ist
  - Fachliche Funktionalität beizubehalten

Weitere Ziele

- Durchgängige Systemintegration
- Realisierung flexibler und transparenter Arbeitsabläufe
- Einsparung von Personal und Zeit
- Arbeitserleichterungen
- Produktivitätserhöhungen

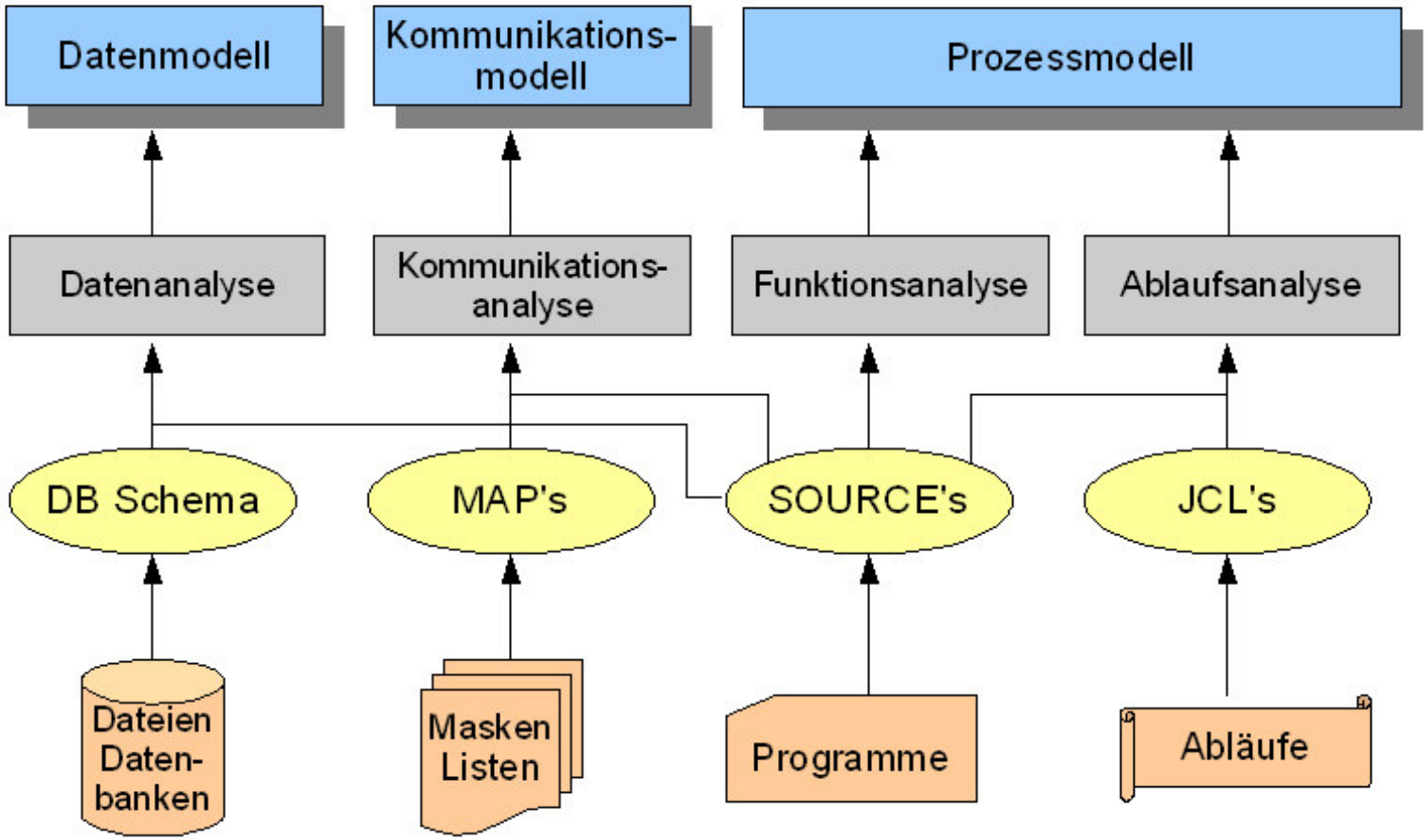

Anforderungen
IT-Betrieb
unterbrechungsfrei
sicher
zuverlässig
Änderungen
nur notwendige
auf andere verzichten
Code
so viel wie nötig ändern
so wenig wie möglich ändern
Auswirkungen
negative minimieren
Umfang und Risiko verringern
Nutzen
moderne Technologien
moderne Methoden


Sitzung A - [24 x 80]
Datei Bearbeiten Sicht Kommunikation Aktionen Fenster Hilfe
Z1111--M L1111--P Landwirtschaftliche Betriebsdatenbank 22.01.10 13:04:41
ADV3207 1.1.1.1 Post-/Antragseingänge zu Unternehmen TLSAC1
Unt-Nr.: 01 07 111 00 000 0015 Unternehmensdaten für: 2009
Bezeich: Test
Name : Vorname:
F11: alle Unternehmen - *Memo-Text:
Eingabehilfe PE-Datum: 22.01.2010 (+ entsprechende Funktion unter ? wählen!)
? Ind. Verfahren tum Bearbeiter Datum
_ 64 EFP 01.01.2009 J 01.01.2009 ADV320B1 06.11.2009
_ 87 FLB 01.01.2002 J 01.01.2002 09FLB-1P 23.05.2007
Direktkommando:
S. 1 / 1
F2=Verf;F3=Abbr;F5=CC;F6=kompl;F8=Vor;F10=Änd;F11=BlättArt;F12=111;F19=AEB-
MA a
04/021
Verbindung zum fernen Server/Host 10.81.224.24 aufgebaut

Altsystem

- Historisch gewachsen
- Keine ausreichende Dokumentation
- Veraltete Betriebs- und Entwicklungsumgebungen
- Zahlreiche Schnittstellen
- Hohe Komplexität
- Mangel an Expertenwissen

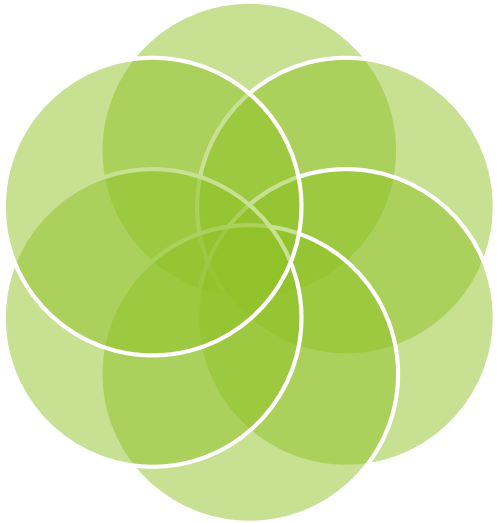
Nachdokumentation



Wird herausgezögert, weil

Zunächst kein Mehrwert durch die Umstellung entsteht

Meist große Datenmengen existieren



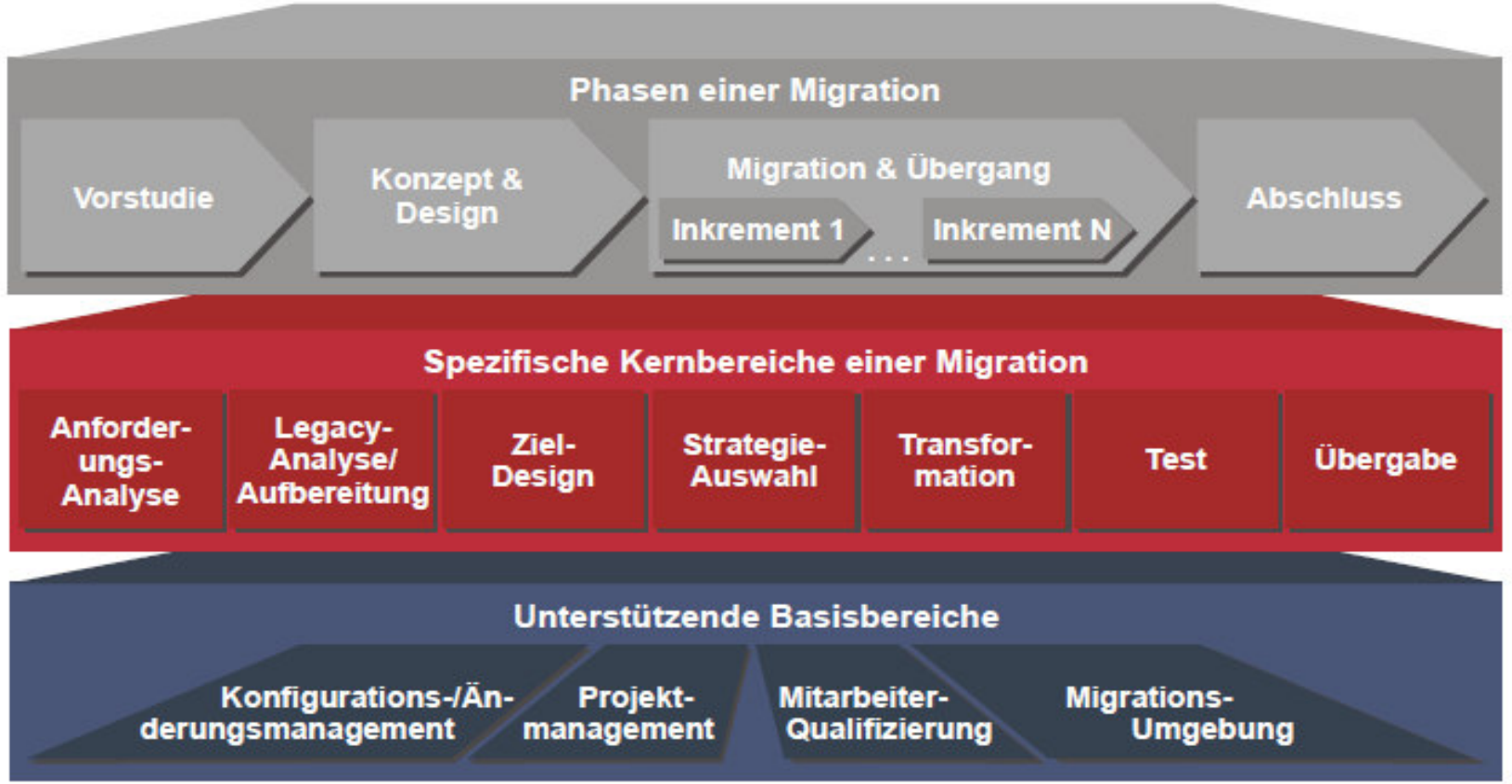
Umstellungskosten entstehen

An Neuentwicklung meist kein Weg vorbei führt

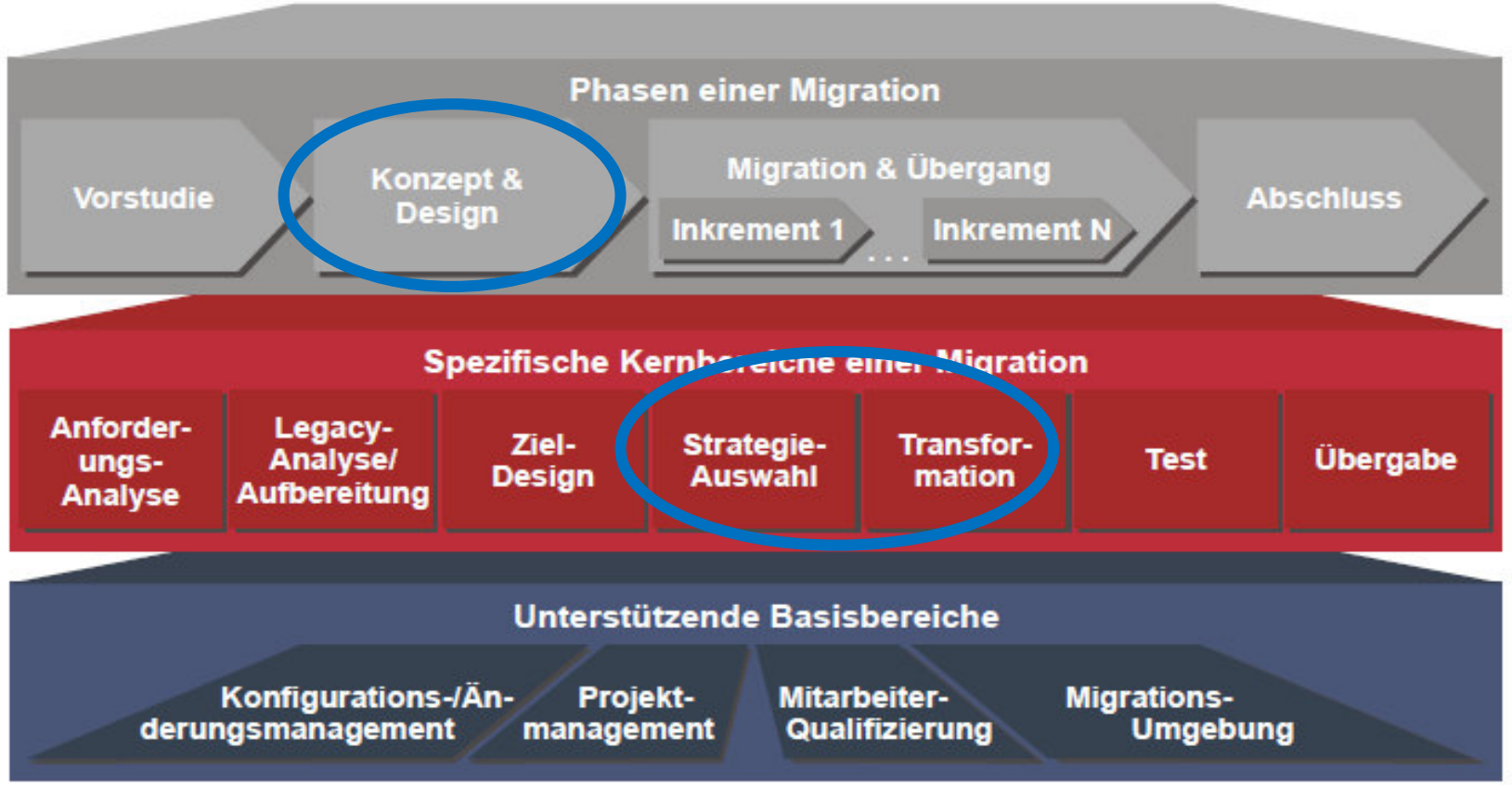
Erhöhte Ausfallrisiken entstehen

Gewachsener Umfang an Funktionen existiert

Prozess


Phasen einer Migration
Vorstudie
Konzept & Design
Migration & Übergang
Inkrement 1
...
Inkrement N
Abschluss
Spezifische Kernbereiche einer Migration
Anforder-ungs-Analyse
Legacy-Analyse/ Aufbereitung
Ziel-Design
Strategie-Auswahl
Transfor-mation
Test
Übergabe
Unterstützende Basisbereiche
Konfigurations-/Än-derungsmanagement
Projekt-management
Mitarbeiter-Qualifizierung
Migrations-Umgebung

Prozess

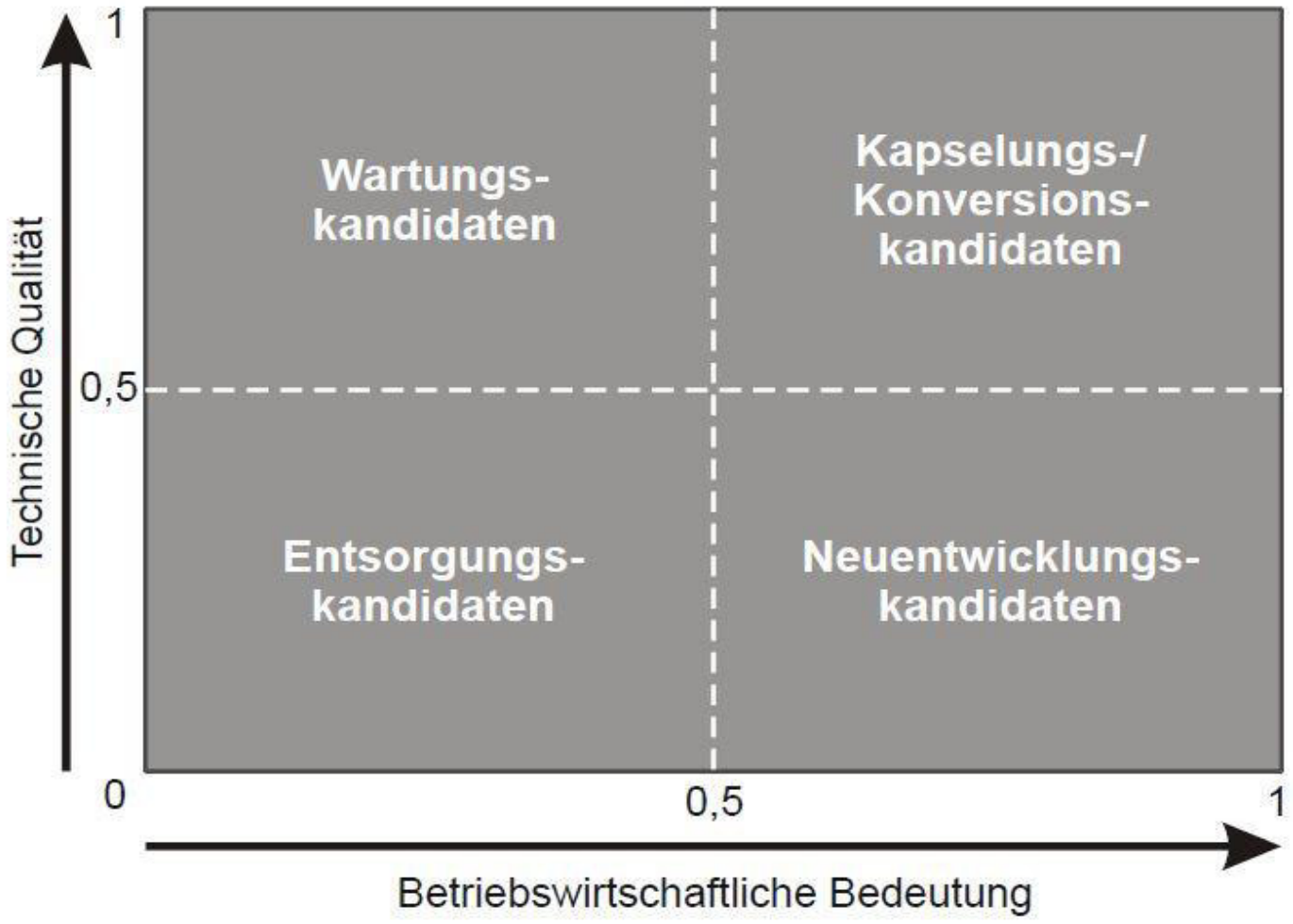

Phasen einer Migration
Vorstudie
Konzept & Design
Migration & Übergang
Inkrement 1
Inkrement N
Abschluss
Spezifische Kernbereiche einer Migration
Anforder-ungs-Analyse
Legacy-Analyse/ Aufbereitung
Ziel-Design
Strategie-Auswahl
Transfor-mation
Test
Übergabe
Unterstützende Basisbereiche
Konfigurations-/Än-derungsmanagement
Projekt-management
Mitarbeiter-Qualifizierung
Migrations-Umgebung

Transformationsstrategien

- Konversion
- Kapselung
- Neuentwicklung

Kann für jedes Modul individuell bestimmt werden

Portfolio Analyse



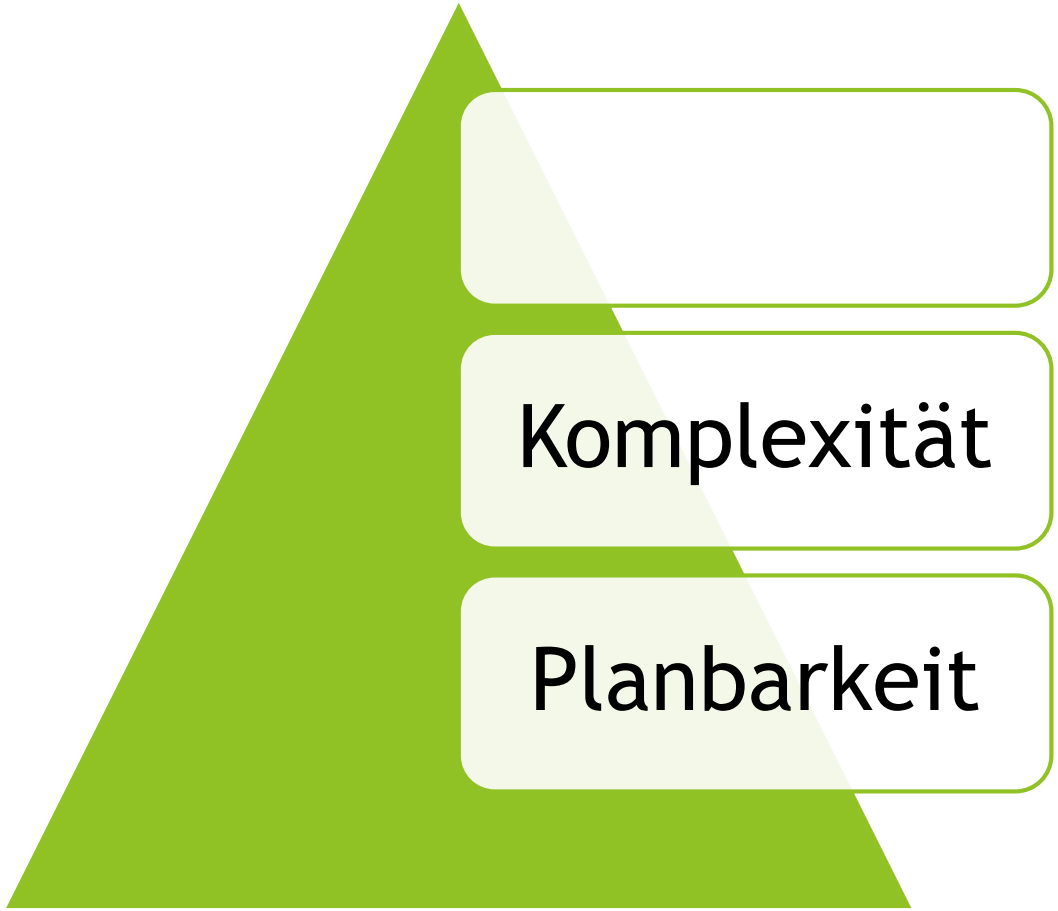
Übergabestrategie

- Cut-over
- Als Gesamtheit zu einem Termin oder schrittweise
- Wird bestimmt durch die Umstellungsstrategie

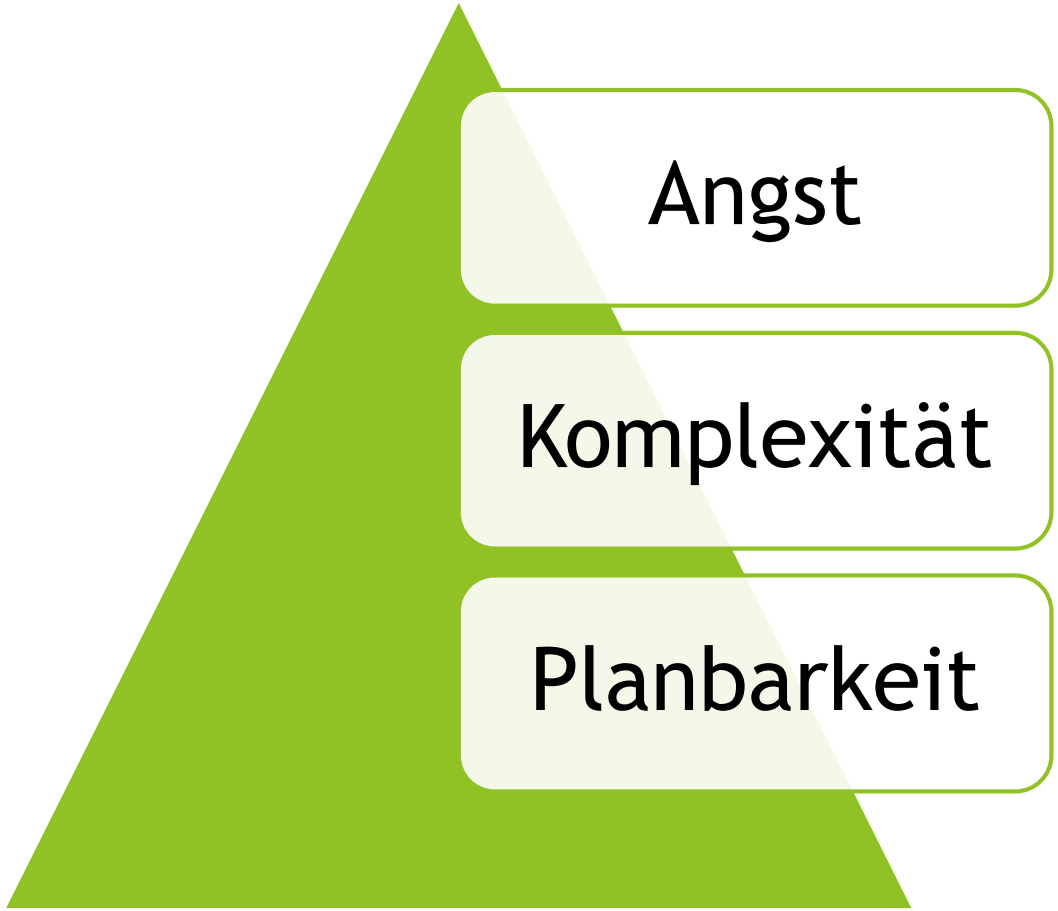
Big Bang Approach

Probleme und Risiken

Probleme und Risiken



Probleme und Risiken



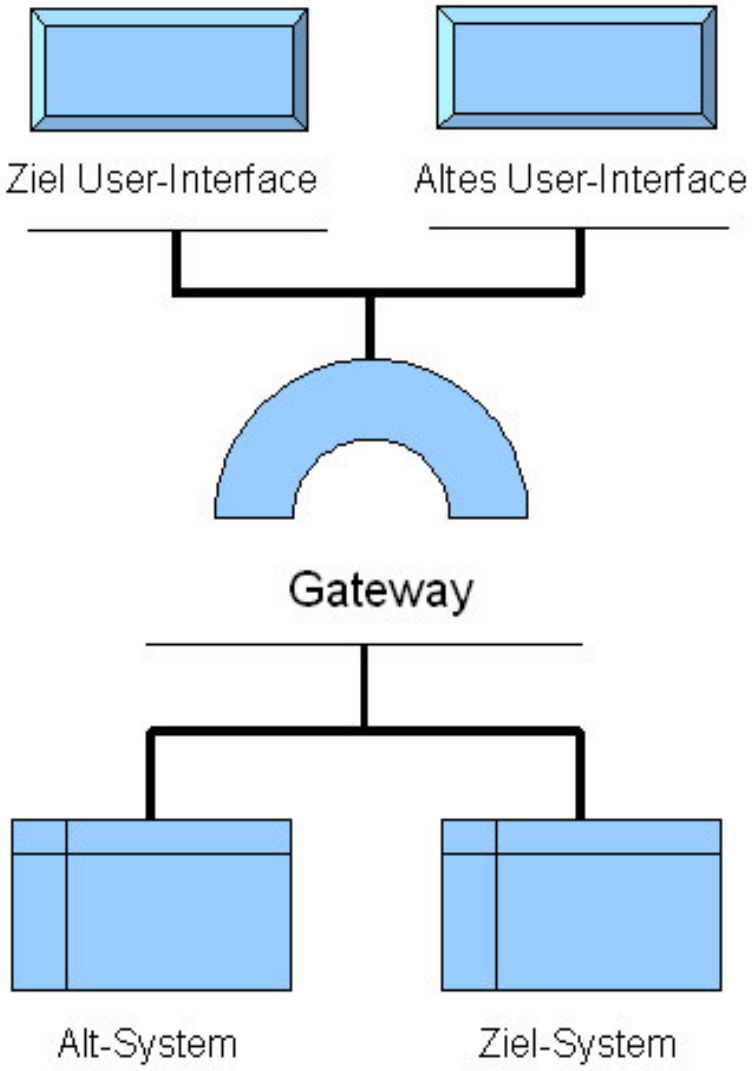
Vor- und Nachteile

- PRO
  - Geringere Entwicklungskosten
  - Geschwindigkeit des Migrationsprozesses

- CONTRA
  - Flash cut ist ohne eine Downtime von Datenbank oder Applikation nicht möglich

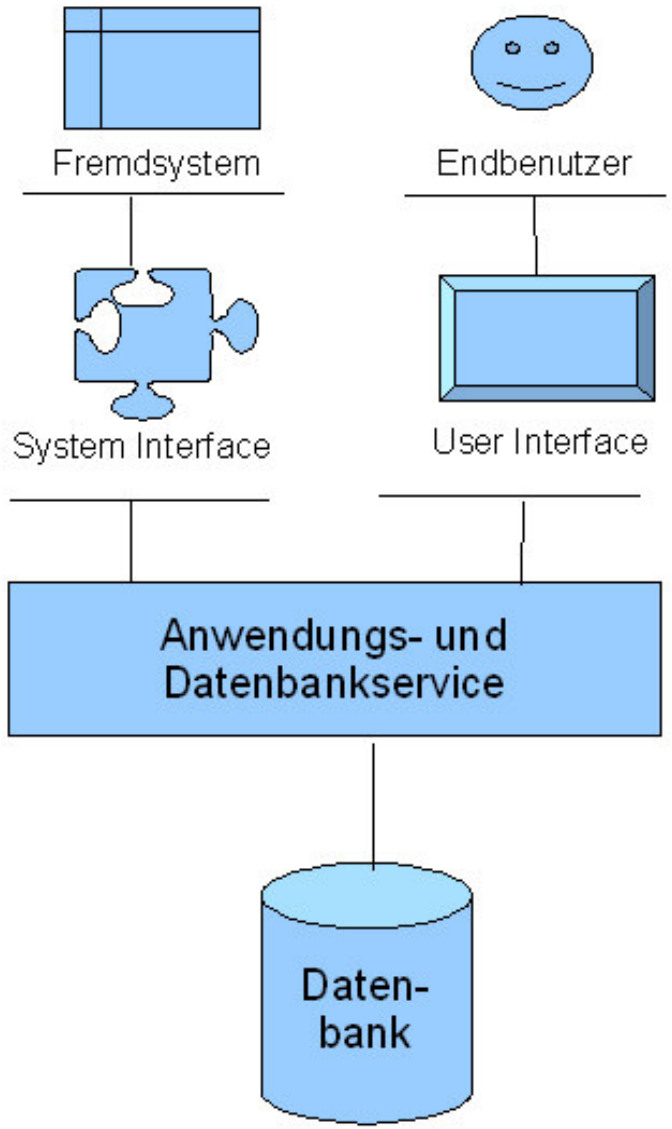
Chicken Little Strategy

Gateways


Ziel User-Interface
Altes User-Interface
Gateway
Alt-System
Ziel-System

Wann geeignet?

- Unstrukturiert
  - alles in einem Modul
  - Hoher Sanierungsaufwand
- Semistrukturiert
  - Z.B. Anwendungs- und Datenschicht eng miteinander verbunden
- Gut strukturiert
  - Präsentation, Datenzugriff und Logik voneinander getrennt



Die 11 Schritte

1. Inkrementelle Analyse des Altsystems

Anforderungen des Alt- und Ziel-Systems müssen erstellt werden, eventuell muss das Altsystem mit Reverse-Engineering Techniken analysiert werden.

Die 11 Schritte

2. Inkrementelle Strukturierung des Altsystems

Definierte Schnittstellen zwischen Modulen und Datenbank-Services müssen gegebenenfalls erstellt werden.

Das kann aufwändig bis unmöglich sein!

Die 11 Schritte

3. Inkrementelles Design der Ziel-Interfaces

An dieser Stelle werden hauptsächlich die User-Interfaces spezifiziert und es wird entschieden, ob ein Gateway erstellt wird.

Die 11 Schritte

4. Inkrementelles Design der Ziel-Anwendungen

Funktionalität und Prozessflow der Alt-Anwendungen werden übernommen oder neu entwickelt

Die 11 Schritte

5. Inkrementelles Design der Ziel-Datenbank

Eventuell müssen Tabellen in relationalen Schemas anders designed werden.

Die 11 Schritte

6. Inkrementelles Installieren der Ziel-Umgebung

Anforderungen werden identifiziert, installiert und getestet.

Die 11 Schritte

7. Inkrementelles Erstellen und Installieren der benötigten Gateways

Anhand der Anforderungen (Funktionen, Architektur und nicht-funktionalen Anforderungen) ist ein Gateway zu entwickeln oder zu kaufen, um es daraufhin zu installieren. **Das kann sehr komplex sein!**

Die 11 Schritte

8. Inkrementelles Migrieren der Alt-Datenbank

Installation des Datenbankmanagementsystems (DBMS) und Implementation des Schemas aus Schritt 5

Die 11 Schritte

9. Inkrementelles Migrieren der Alt-Anwendungen

Auswahl und Migration eines oder mehrerer Module

Die 11 Schritte

10. Inkrementelles Migrieren der Alt-Interfaces

Auswahl und Migration eines oder mehrerer Interfaces

Die 11 Schritte

11. Inkrementelles Umschalten auf die Zielumgebung

Modul für Modul, Oberfläche für Oberfläche kann umgeschaltet werden

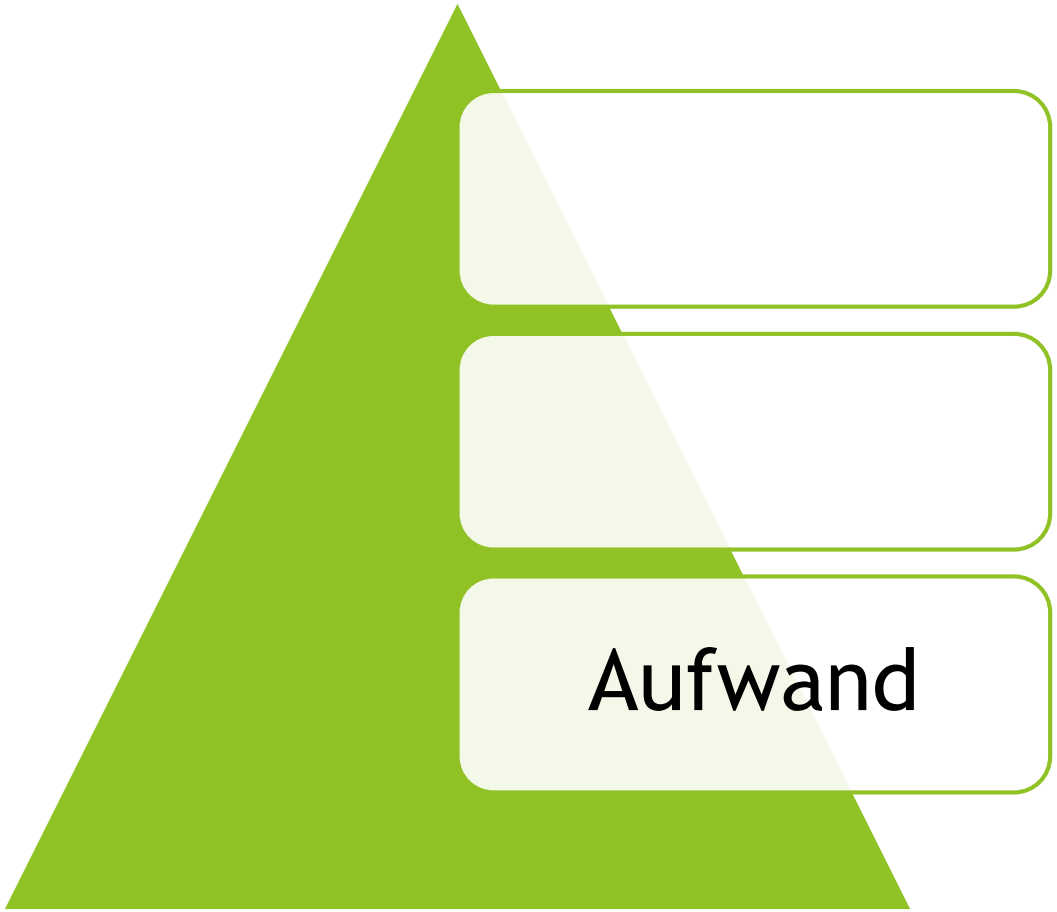
Die 11 Schritte

1. Inkrementelle Analyse des Altsystems
2. Inkrementelle Strukturierung des Altsystems
3. Inkrementelles Design der Ziel-Interfaces
4. Inkrementelles Design der Ziel-Anwendungen
5. Inkrementelles Design der Ziel-Datenbank
6. Inkrementelles Installieren der Ziel-Umgebung
7. Inkrementelles Erstellen und Installieren der Gateways
8. Inkrementelles Migrieren der Alt-Datenbank
9. Inkrementelles Migrieren der Alt-Anwendungen
10. Inkrementelles Migrieren der Alt-Interfaces
11. Inkrementelles Umschalten auf die Zielumgebung

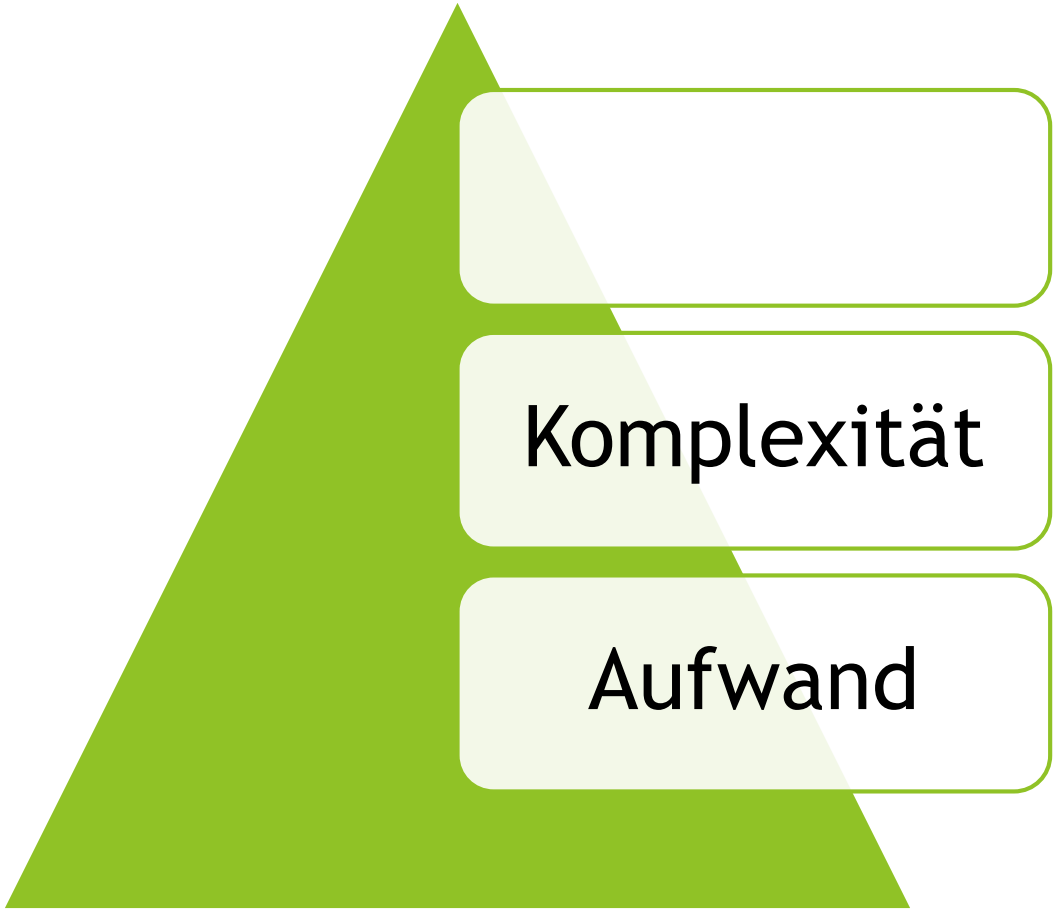
Zusammenfassung

Chicken Little bedeutet im Endergebnis dennoch eine komplette Neuentwicklung des Systems, allerdings mit geringeren Risiken als bei dem Big-Bang-Ansatz

Probleme



Probleme

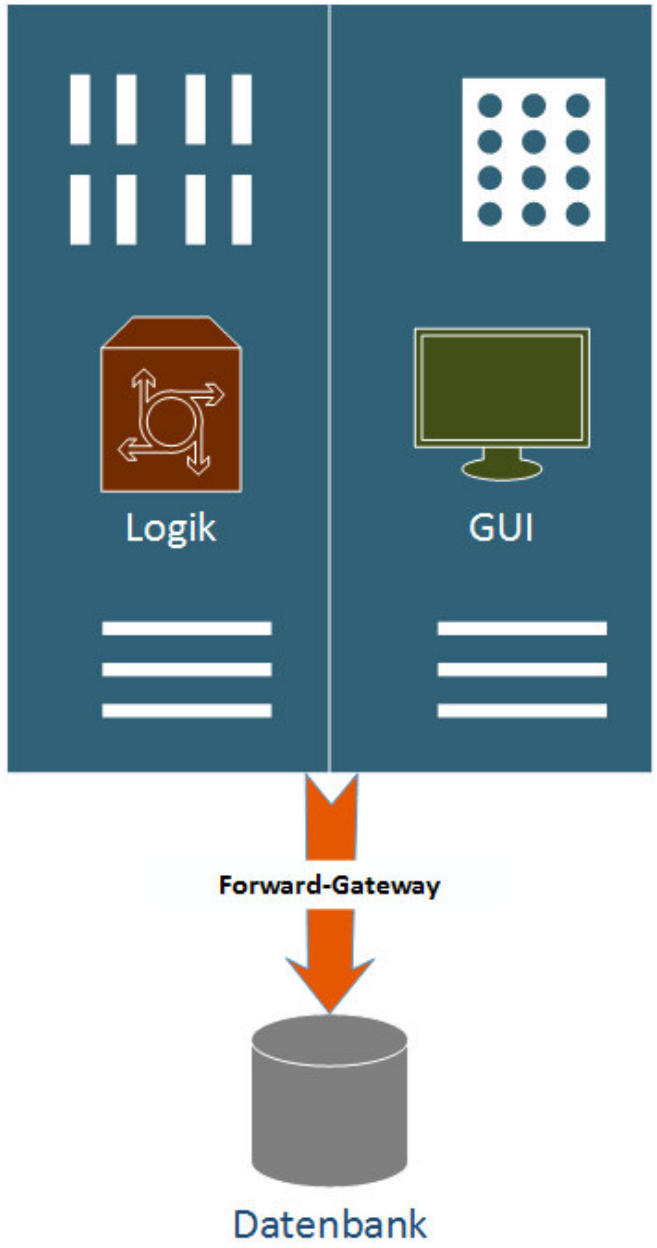


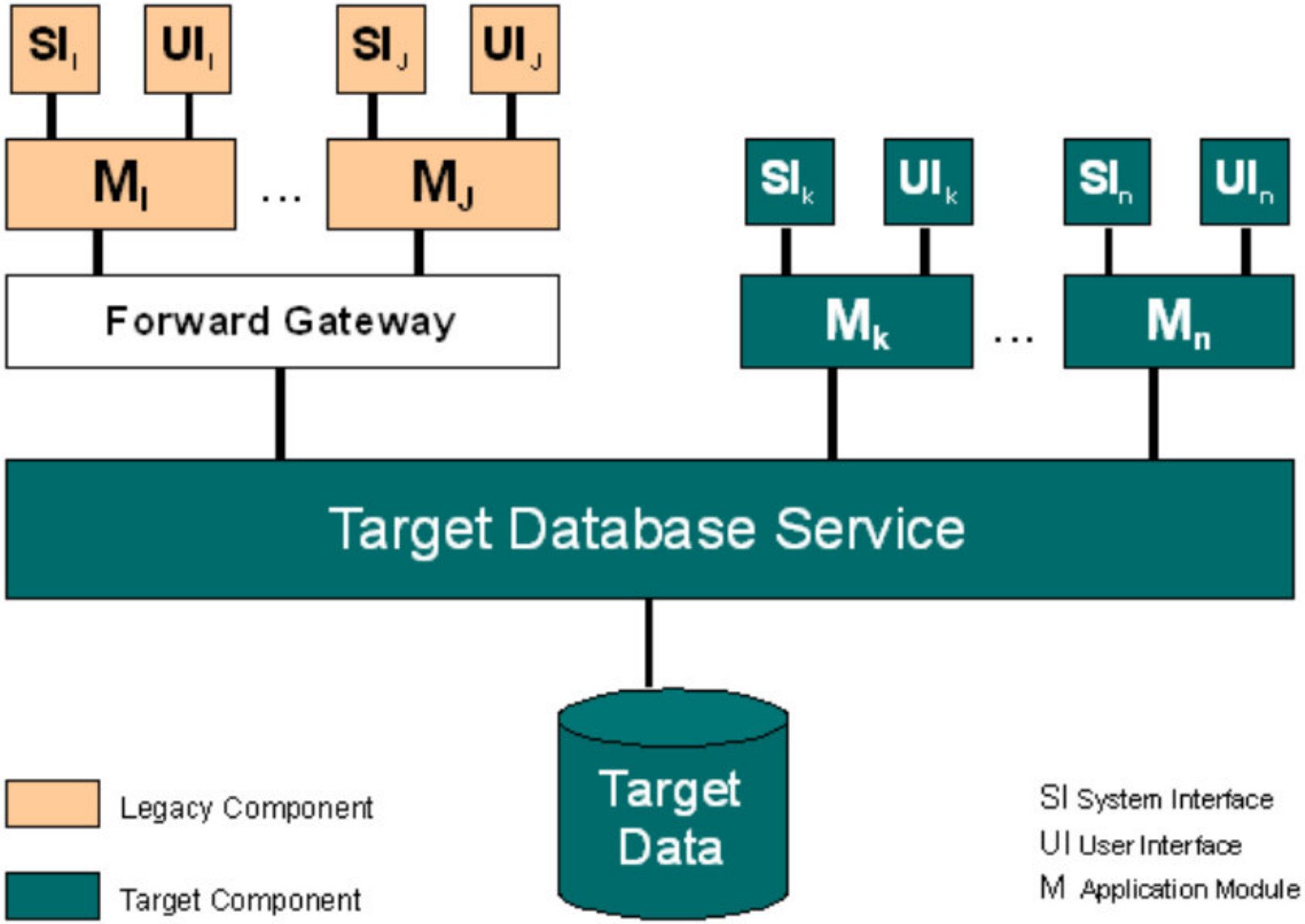
Probleme

Zusammenfassung

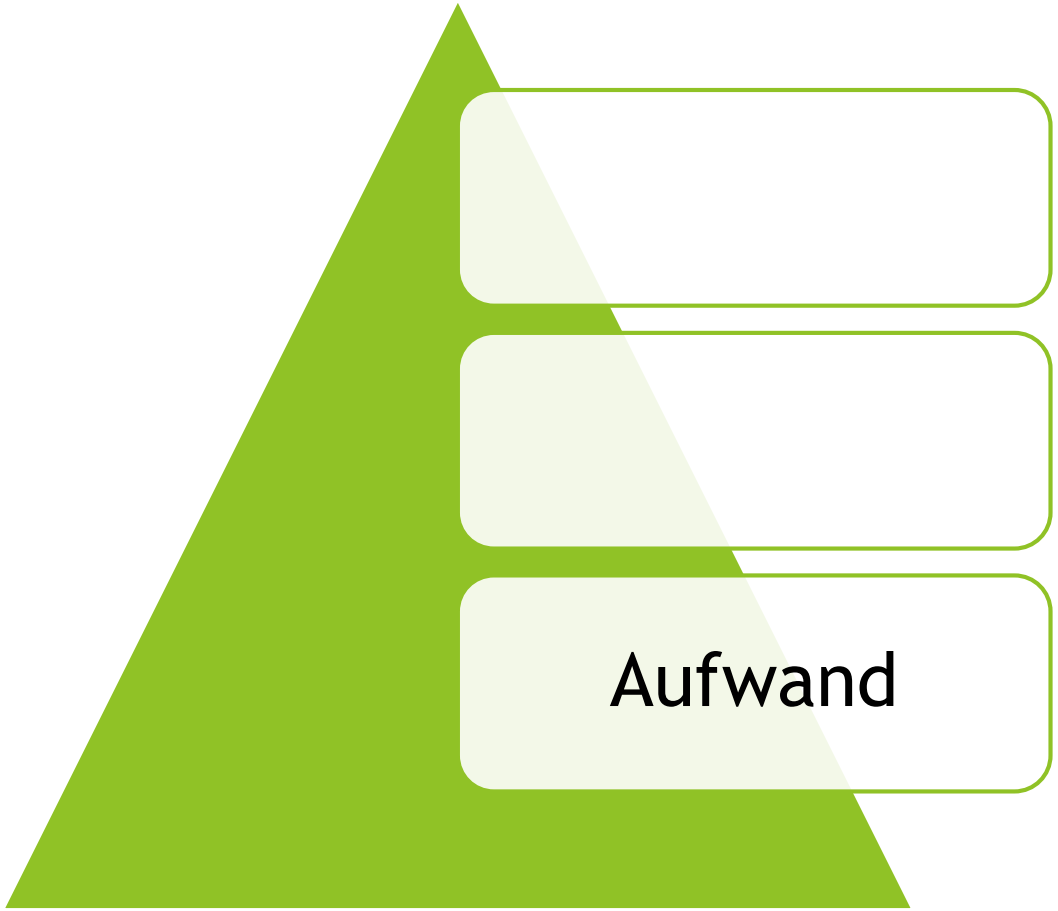
	Cold Turkey	Chicken Little
Risiko	Groß	Kontrollierbar
Fehleranfälligkeit	Das ganze Projekt schlägt fehl	Nur einzelne Schritte schlagen fehl
Ergebnisse	Sofort nach Fertigstellung sichtbar, evtl. aber nicht von langer Dauer	Nutzen steigt stetig mit der Entwicklung des neuen Systems
Ausblick	Nicht vorhersagbar bis zum Endergebnis	Konservativ optimistisch

Database First Approach

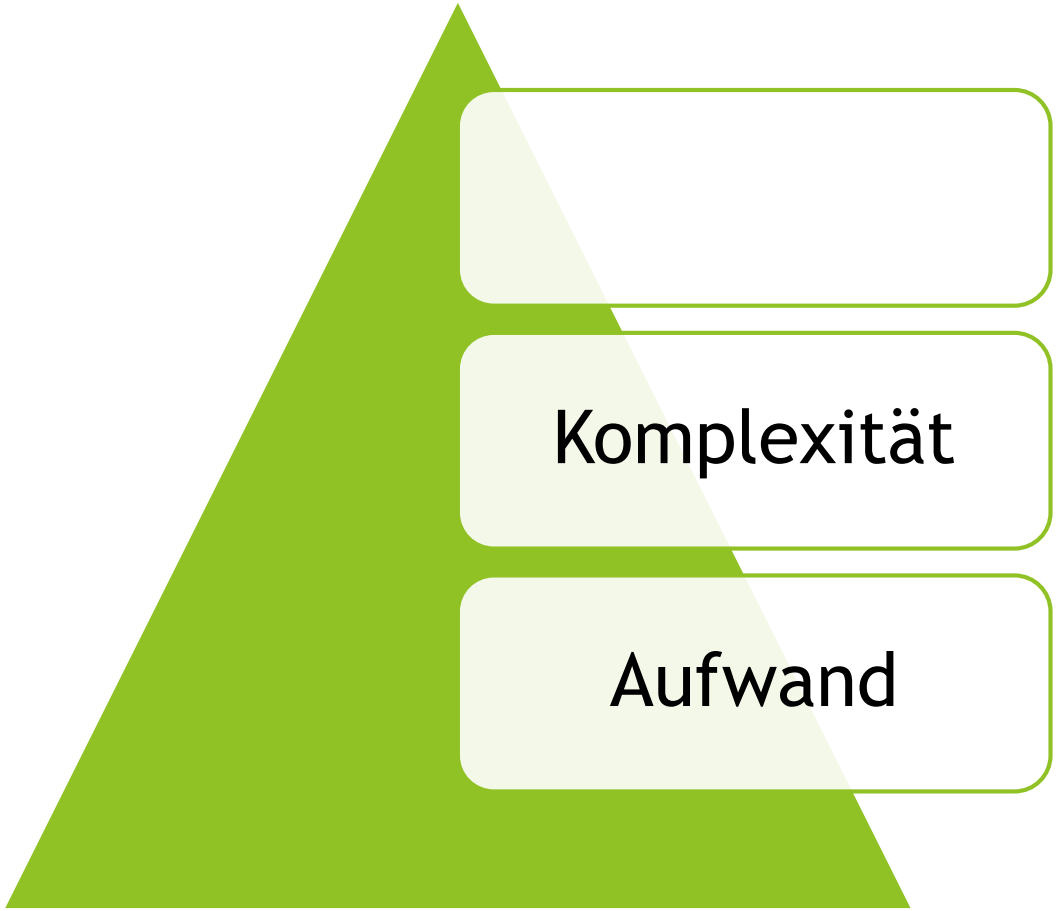

Logik
GUI
Forward-Gateway
Datenbank


SI_I
UI_I
SI_J
UI_J
M_I
...
M_J
SI_k
UI_k
SI_n
UI_n
Forward Gateway
M_k
...
M_n
Target Database Service
Target Data
Legacy Component
Target Component
SI System Interface
UI User Interface
M Application Module

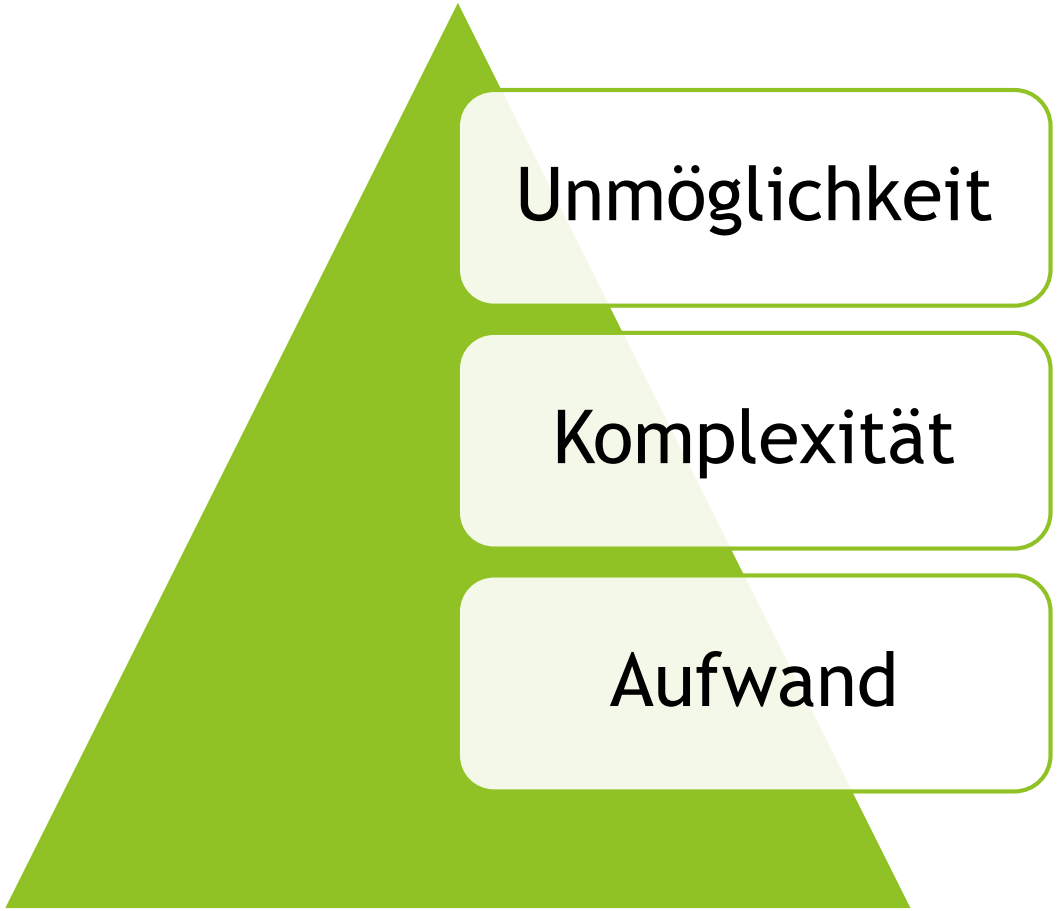
Probleme



Probleme

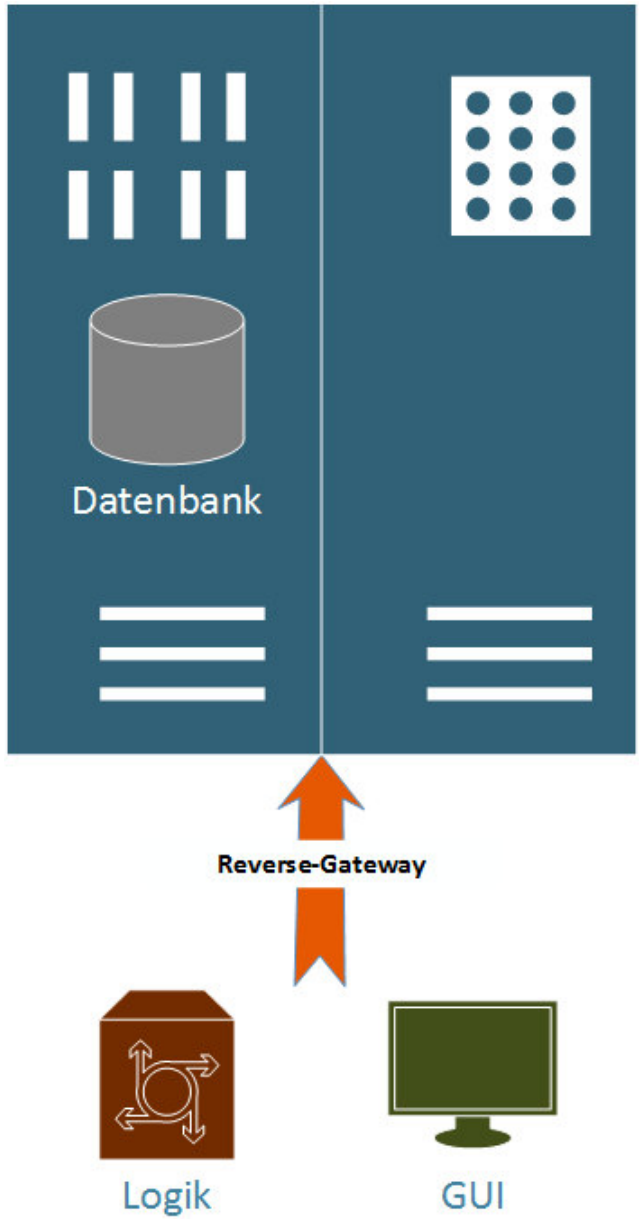


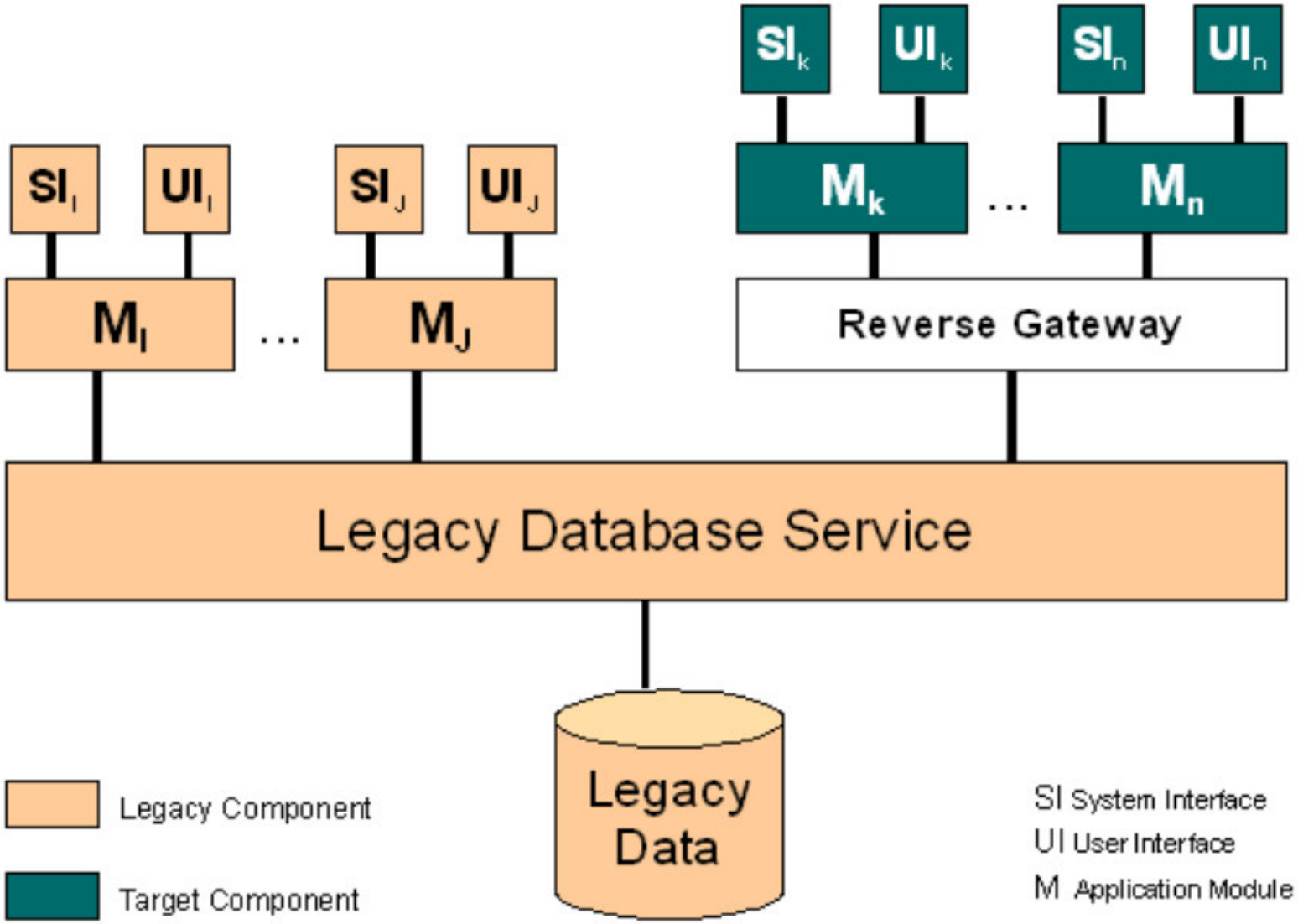
Probleme



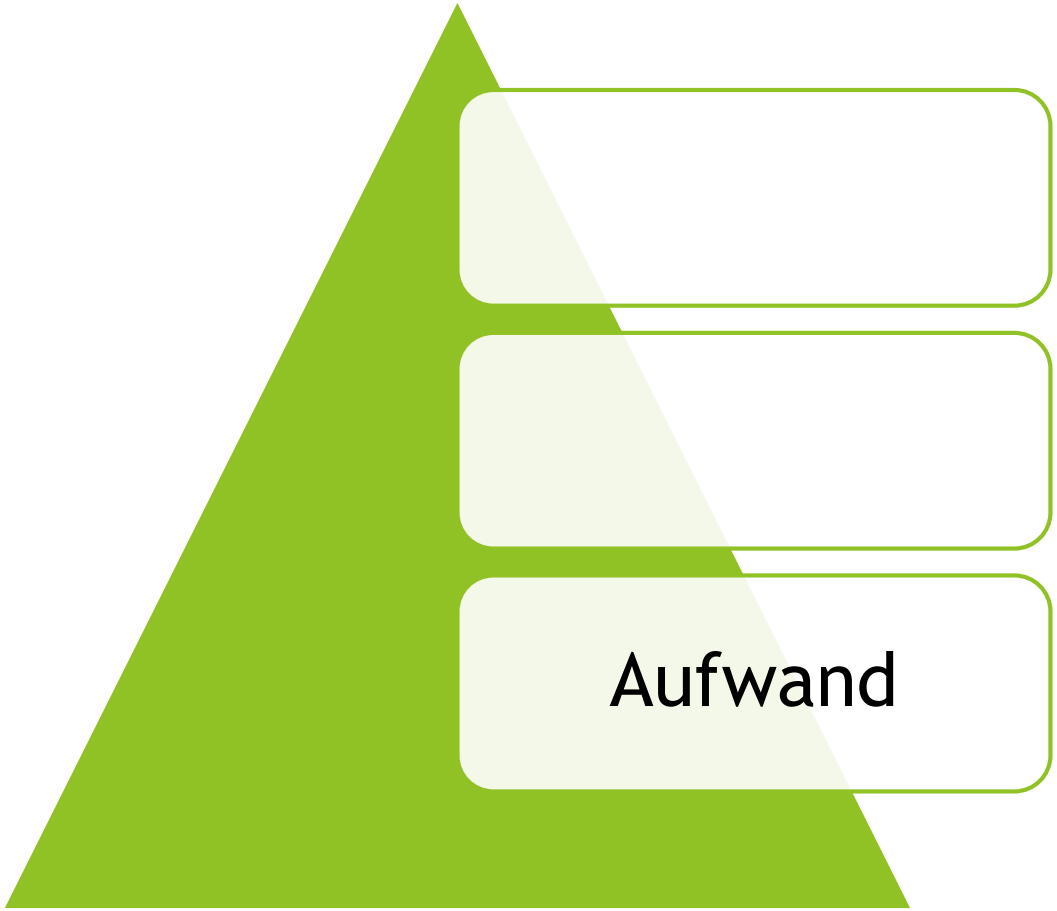
Datenmigration?

Database Last Approach

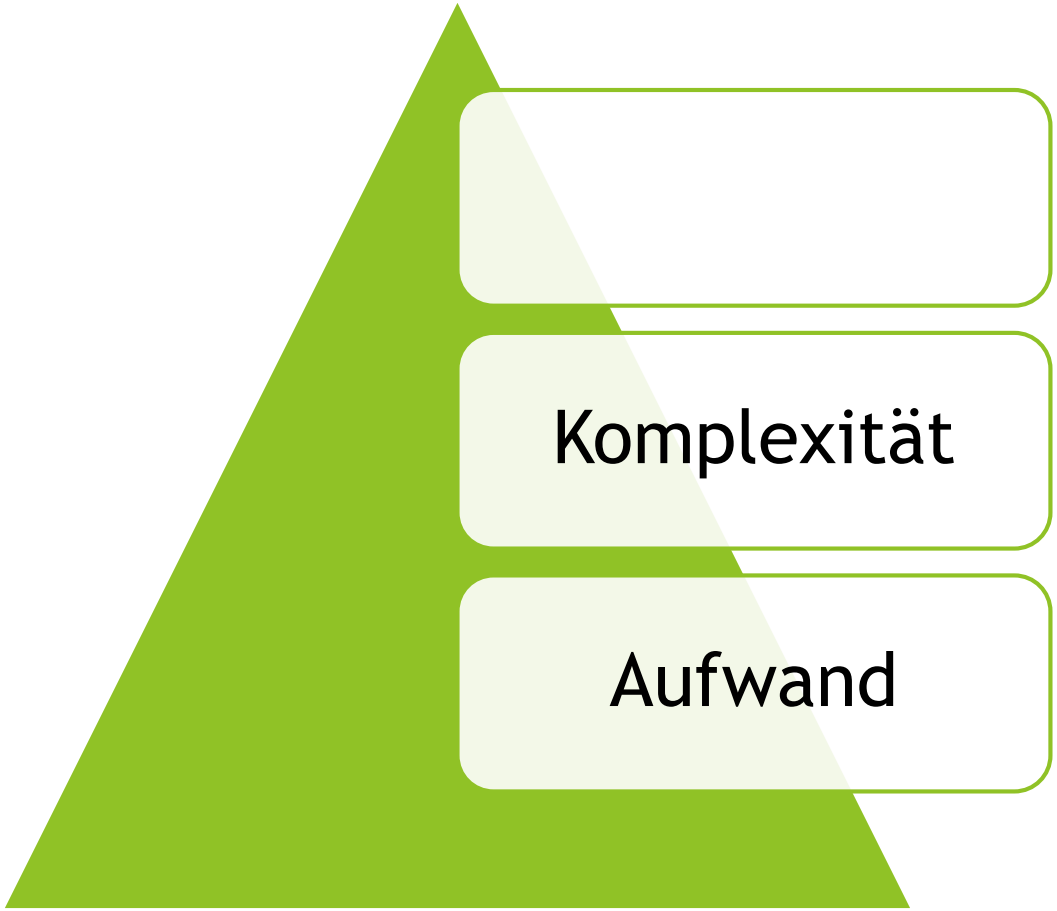

Datenbank
Reverse-Gateway
Logik
GUI


SI_k
UI_k
SI_n
UI_n
SI_I
UI_I
SI_J
UI_J
M_k
...
M_n
M_I
...
M_J
Reverse Gateway
Legacy Database Service
Legacy Data
Legacy Component
Target Component
SI System Interface
UI User Interface
M Application Module

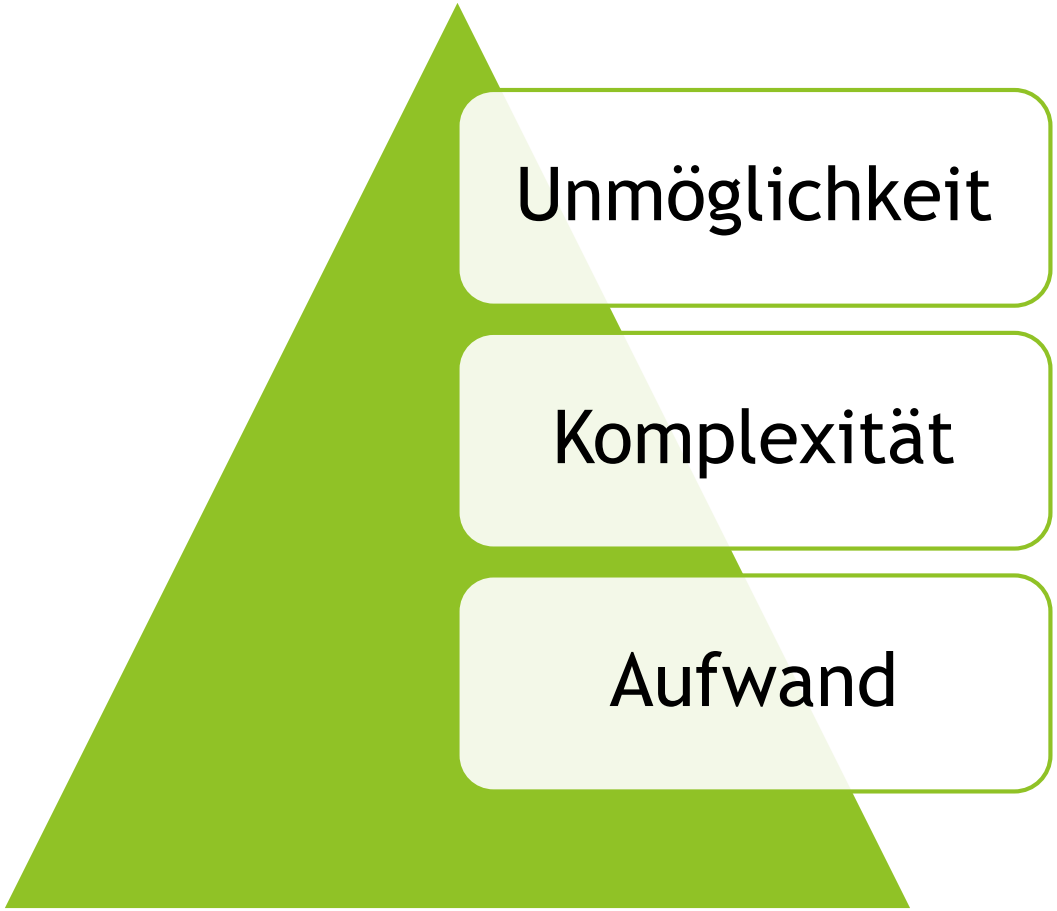
Probleme



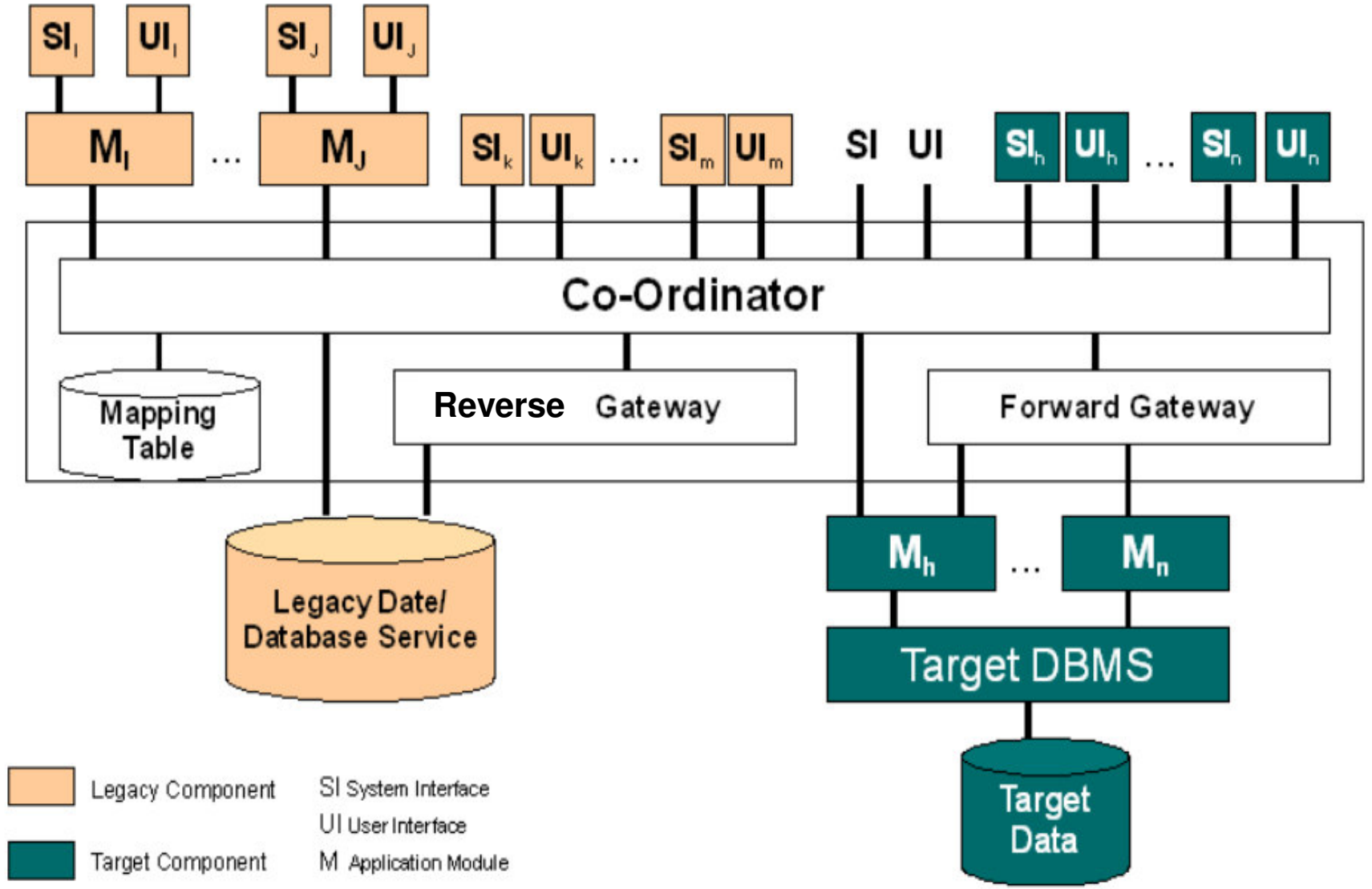
Probleme



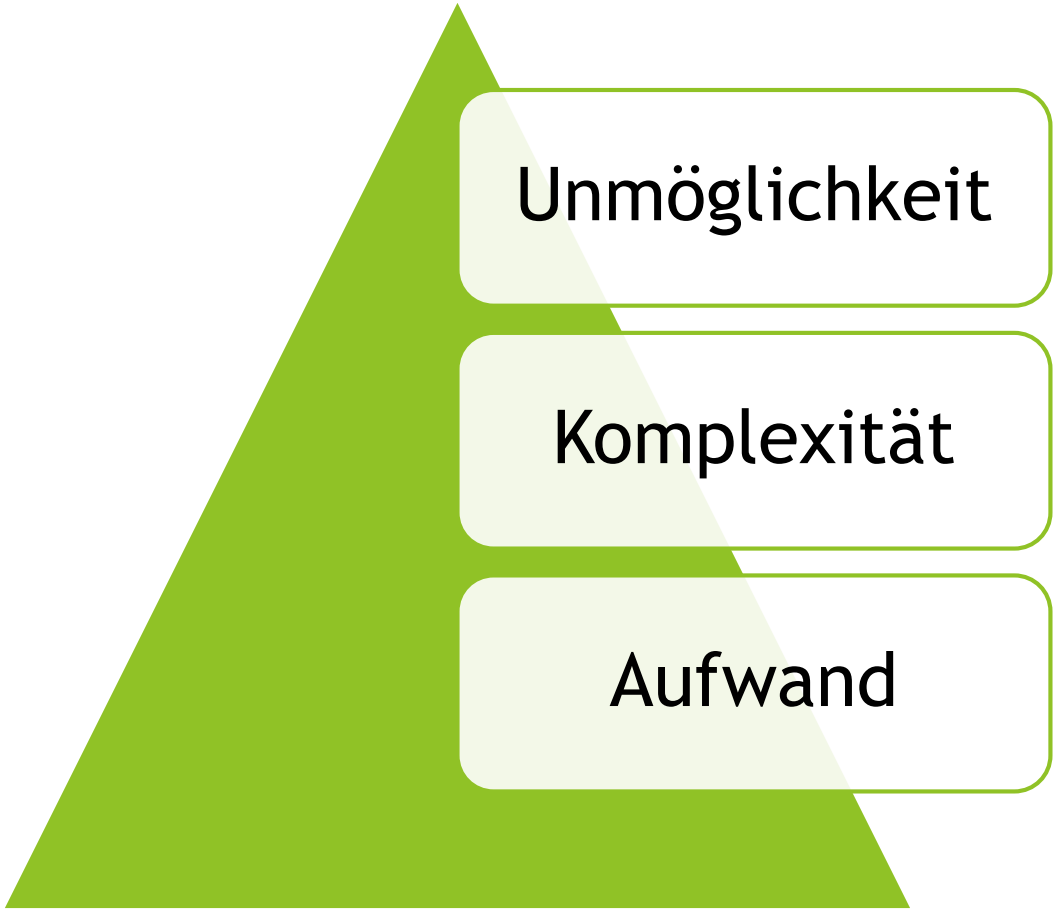
Probleme



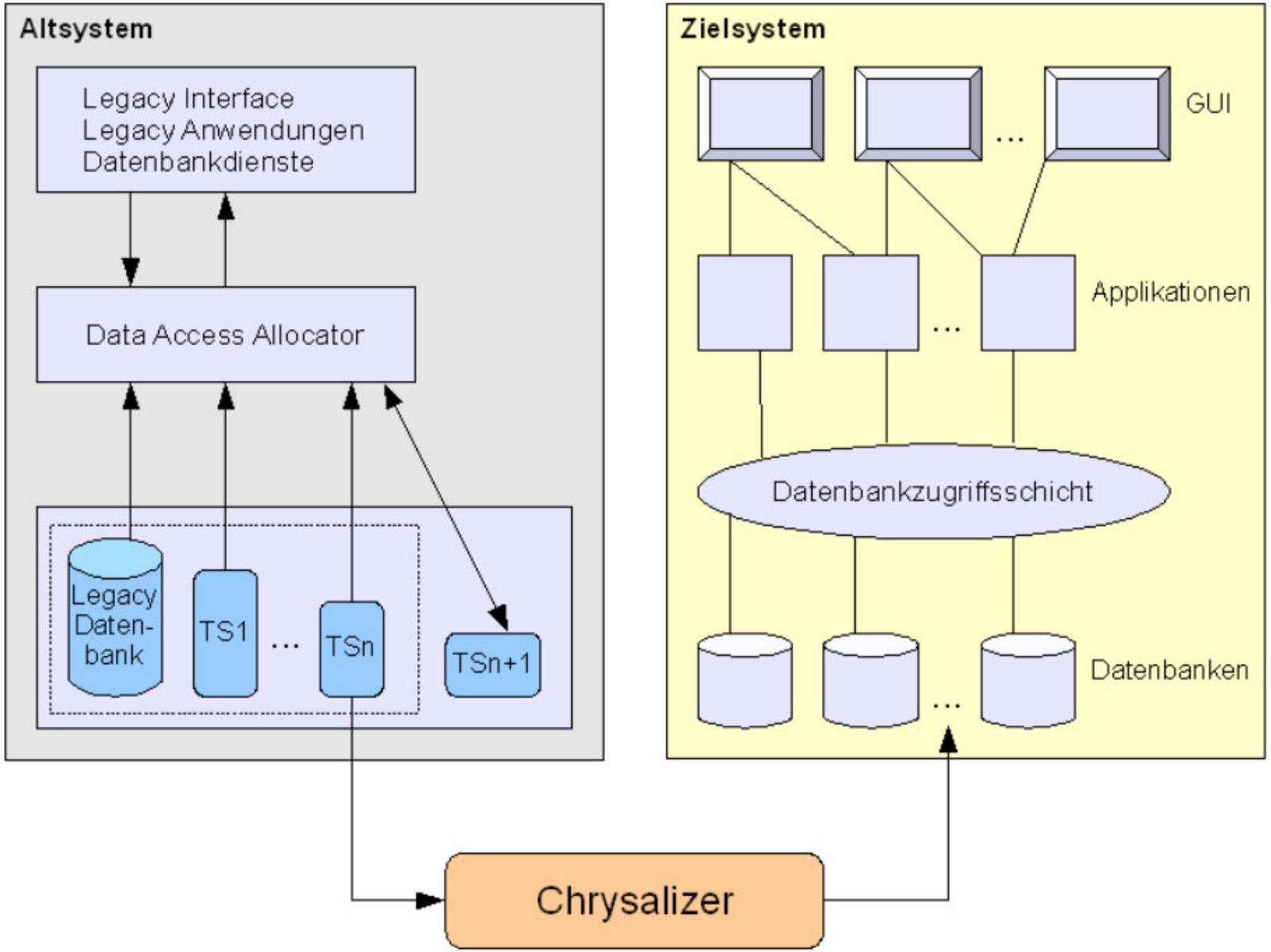
Composite Database Approach


SI I
UI I
SI J
UI J
M I
M J
SI k
UI k
SI m
UI m
SI
UI
SI h
UI h
SI n
UI n
Co-Ordinator
Mapping Table
Reverse Gateway
Forward Gateway
Legacy Date/ Database Service
M h
M n
Target DBMS
Target Data
Legacy Component
Target Component
SI System Interface
UI User Interface
M Application Module

Probleme



Butterfly Methodology


Altsystem
Legacy Interface
Legacy Anwendungen
Datenbankdienste
Data Access Allocator
Legacy Daten-bank
TS1
...
TSn
TSn+1
Zielsystem
GUI
Applikationen
Datenbankzugriffsschicht
Datenbanken
Chrysalizer

Funktionsweise

1. Sobald Datenmigration beginnt, wird Bestand des Altsystems eingefroren und mit Schreibschutz versehen

Ein Data-Access-Allocator (DAA) leitet von nun an alle Anfragen um. Änderungen werden in temporären Speichern verwaltet. Bei dem Aufruf bereits geänderter Informationen, werden diese aus dem temporären Speicher gelesen.

Funktionsweise

2. Überführung des Datenbestandes des Altsystems in das neue System mit Hilfe des „Chrysalizers“

Dies geschieht auf Grundlage definierter Transformationsregeln, die im Chrysalizer implementiert sind. Dies geschieht ohne Datenverlust, da zwischenzeitliche Änderungen im ersten Temporärspeicher abgelegt werden.

Funktionsweise

3. Migration der Daten des Temporärspeichers 1

Zwischenzeitliche Änderungen werden in Temporärspeicher 2 geschrieben und der erste wird mit Schreibschutz versehen.

Funktionsweise

4. Das geht so lange, bis die Dauer der Migration des letzten Temporärspeichers so kurz ist, dass keine Änderungen mehr stattfinden konnten

In diesem Stadium kann das Altsystem abgeschaltet, der letzte Temporärspeicher überführt und das neue System in Betrieb genommen werden.

Vorteile

- Ermöglicht eine Migration ohne Downtime
- Man kann jederzeit abbrechen (Temporärspeicher wieder zurückführen)
- Parallele Neuentwicklung kann umfangreiche Tests mit aktuellen Daten durchführen

Nachteile

- Bei großen Datenbeständen ist ein hoher Hardwareeinsatz notwendig (für eine große Zahl an Temporärspeichern)
- Die Entwicklung des DAA kann sehr aufwändig werden

v / u

- Ob die Butterfly-Methode erfolgreich ist, hängt vor allem von dem Faktor v / u ab
  - u ist die Geschwindigkeit des Chrysalizers
  - v die Geschwindigkeit des DAA, neue Temporärspeicher anzulegen.
  - Bei v = 0 ähnelt die Vorgehensweise der Big-Bang-Strategie
  - bei v > u wird die Migration nie enden

Erfolgsfaktoren

- ein gründliches Verständnis des Alt- und Zielsystems
- eine fehlerfreie Testdatenbank (Initialmigration)
- ein schneller Chrysalizer
- ein effizienter Data-Access-Allocator

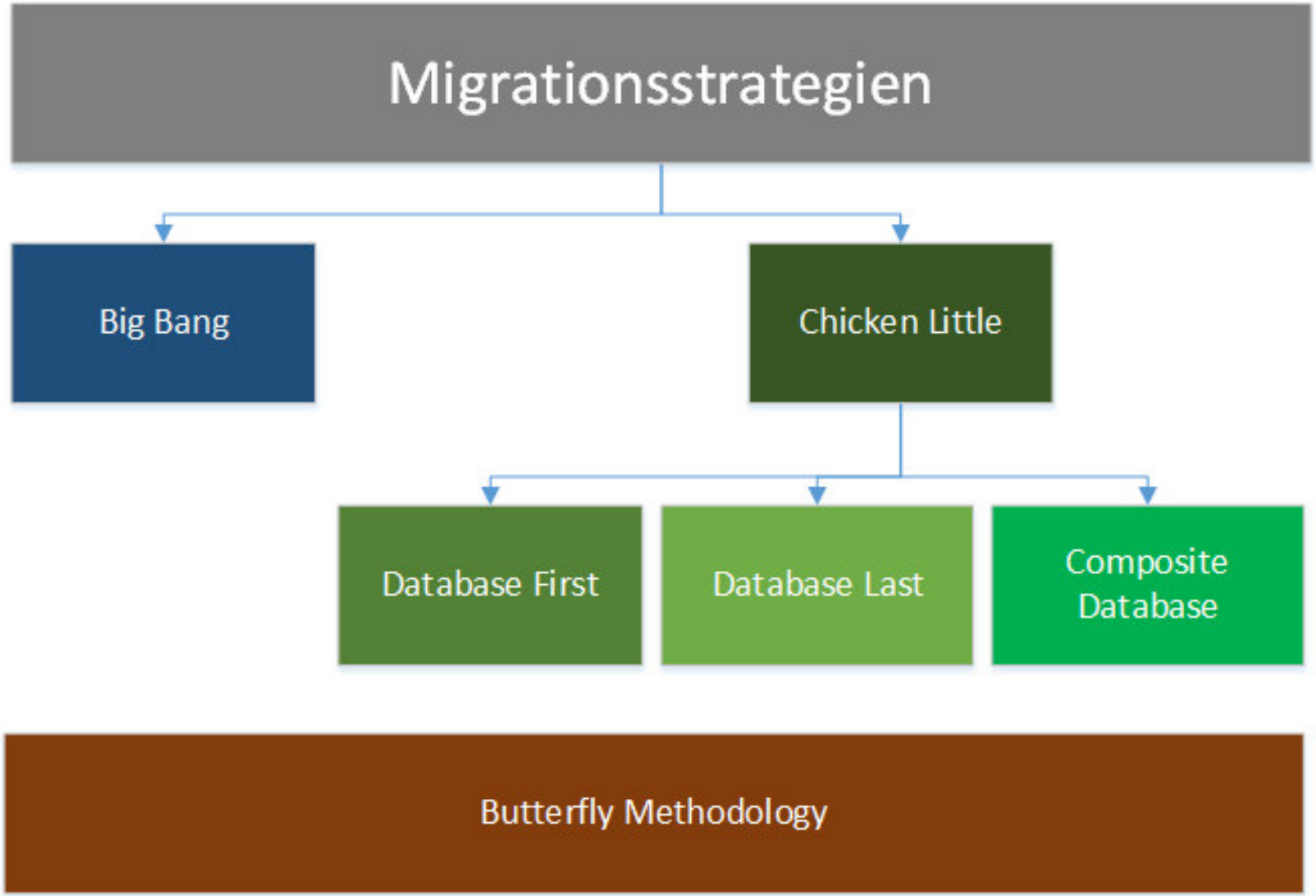
Weitere Ansätze

Weitere Ansätze

- Frontend-Switch
- Change-On-Copy
- Change-Via-Split
- Change-By-Extension
- Branch-For-Improvement
- Branch-By-Abstraction

  (Anticorruption Layer / Isolate Change)

Zusammenfassung


Migrationsstrategien
Big Bang
Chicken Little
Database First
Database Last
Composite Database
Butterfly Methodology

Zusammenfassung

Chicken Little, Cold Turkey

… oder was?

Entscheidende Fragen

- Umfang der Daten
- Ist eine Downtime akzeptabel?
- Kritikalität des Gesamtsystems
- Verfügbares Personal
- Ist das System in unabhängige Module zerlegbar?

aim42.org

- Architecture improvement method
- OpenSource
- Software evolution und optimization
- Sammlung von Pattern und Best Practices für die Phasen analyze, evaluate und improve

Mal was anderes

Continuous Delivery =

Continuous Migration?

What about Feature Toggles?

Or Database Migration Tools?

work just simply smarter

- Continuous Delivery
- Application Lifecycle Management
- BizDevOps

- JIRA
- Jenkins
- Confluence
- Schulung
- Coaching
- Training
- Vorträge
- Artikel
- PoCs

Vielen Dank

Noch Fragen?

http://www.kitenco.de

info@kitenco.de | @kitenco1