Orientation in Objects

Rethinking Software Migration im Zeitalter der Webanwendung

Die Migration unserer Software kann doch kein Problem sein, wenn man dazu schon Lehrbuchlösungen findet. Der Vortrag diskutiert Probleme klassischer Migrationsstrategien wie z. B. Big Bang Approach, Chicken Little Strategy oder Database First Approach auf ihre in der Praxis der Webanwendungsentwicklung auftretenden Probleme. Anhand praktischer Erfahrungen mit dem Versagen klassischer Lösungsstrategien werden alternative Denkansätze und Lösungsstrategien wie z. B. MicroSOA, Modularisierung oder Service-Mashup als Alternativen präsentiert.

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0

Java, XML und Open Source seit 1998

Orientation in Objects

Java und XML

) Software Factory)

- **Schlüsselfertige Realisierung von Java Software**
- **Individualsoftware**
- **Pilot- und Migrationsprojekte**
- **Sanierung von Software**
- **Software Wartung**

) Object Rangers)

- **Unterstützung laufender Java Projekte**
- **Perfect Match**
- **Rent-a-team**
- **Coaching on the project**
- **Inhouse Outsourcing**

) Competence Center)

- **Schulungen, Coaching, Weiterbildungsberatung, Train & Solve-Programme**
- **Methoden, Standards** und **Tools** für die Entwicklung von **offenen, unternehmens-weiten Systemen**

Ihr Sprecher

Orientation in Objects

Christian Dedek

SW-Architekt, Berater,CTO

Enterprise Java Technologien
Software Engineering
Qualitätsmanagement

Gliederung

Orientation in Objects

- Motivation/Problemstellung
- „klassische“ Lösungsstrategien
- Alternative Denkansätze

Motivation

Orientation in Objects

- Software als Altlast
  - Die Dokumentationen sind unübersichtlich, unvollständig, veraltet oder gar nicht mehr vorhanden
  - Die Programme sind selten klar strukturiert, hieraus resultiert eine erhöhte Komplexität
  - Es existieren zahlreiche Schnittstellen zu weiteren Systemen

Stahlknecht, Peter ; Hasenkamp, Ulrich: Einführung in die Wirtschaftsinformatik. 11. Auflage. Berlin : Springer, 2005.

- Vorteile Moderner Softwaresysteme
  - Effizienz
  - Benutzbarkeit
  - Spezialisierung
  - Entkopplung

- Lösungsstrategien
  - Beibehaltung (langfristig Stagnation)
  - Ablösung durch Standardsoftware
  - Migration
  - Neuentwicklung/“Cold Turkey“

Neuentwicklung

Orientation in Objects

Neuimplementierung - Cold Turkey

Orientation in Objects

- Vollständige Neuentwicklung
  - parallel zum Altsystem das neue System entwickeln und testen
- Potentiale:
  - moderne Entwicklungswerkzeuge und –verfahren einsetzbar
  - theoretisch maximale Produktivität erzielbar
  - neue Features erhöhen den Systemnutzen teilweise massiv
  - „saubere“ Entwicklung mit dokumentierten Requirements nach modernen Engineeringmethoden
  - Nutzung von architekturellen und technologischen Blue Prints möglich

Cold Turkey Probleme

Orientation in Objects

- Komplexität durch Größe des Projekts
- vollständige Neuentwicklung benötigt Zeit ~ doppeltes Requirementsmanagement (moving target problem)
- **unklare/undokumentierte Requirements** müssen upfront evaluiert werden
- undokumentierte Abhängigkeiten werden oft erst in späten Phasen entdeckt
- zeitliche Durchführbarkeit einer vollständigen Datenmigration bei konkreter Installation ?
- 4fache Kosten gegenüber Konversionsmigration?
- 60% der Vorhaben scheitern

Cold Turkey Kritik

Orientation in Objects

- “Cold Turkey involves high risk. It has been applied and has failed many times in large organizations. We now turn our attention to the alternative, low-risk and novel strategy”

Brodie M.,Stonebraker M.: „DARWIN: On the Incremental Migration of Legacy Information Systems.. “

  - **„ Cold Turkey “ - ein plötzlicher Entzug von Drogen**
    - https://de.wikipedia.org/wiki/Cold_Turkey

- Bekannte Probleme von “Cold Turkey” führten u.a. zur Entwicklung alternativer Migrationsstrategien
  - Chicken Little
  - Composite Database
  - Butterfly

Definitionsversuch Migration

Orientation in Objects

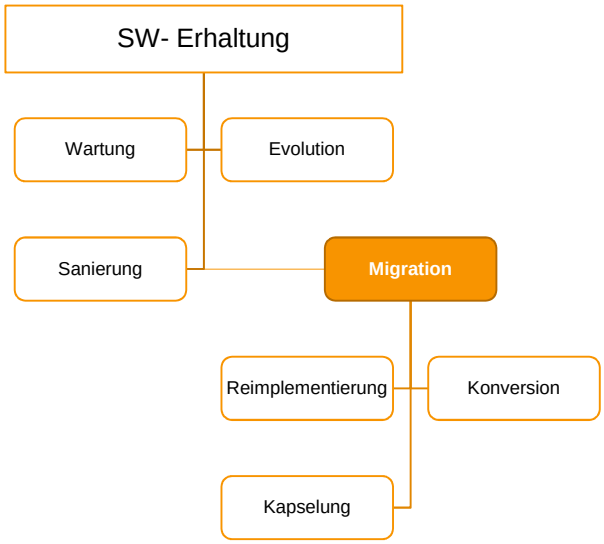
- viele teilweise widersprüchliche Definitionen
  - z.B.https://de.wikipedia.org/wiki/Migrationsstrategien_(Informationstechnik)

- „Eine Software-Migration bezeichnet die Überführung eines Softwaresystems in eine andere Zielumgebung oder in eine andere Form ohne Veränderung der fachliche Funktionalität. “

Ackermann, E. „Ein Referenz-Prozessmodell zur Software-Migration“

- Abgrenzung
  - keine Sanierung
  - keine Nachdokumentation/Reverse Engineering
  - keine Wartung
  - keine Evolution
    - **keine neuen/geänderten Funktionen**

Begriffsbestimmung



Grundideen

- ununterbrochener, sicherer, zuverlässiger Betrieb
- Migration auf eine zukunftsfähige Basis
  - Nach der Migration ist nicht der Beginn des nächsten Projektes
- Maximale Senkung des Migrationsrisikos
  - Altsourcen wenn möglich 1:1 übernehmen
- Legacysystemanpassungen sind meist ökonomisch günstiger und weniger riskant als vollständige Neuentwicklungen
- Starke analytische QM-Massnahmen
  - Erkennen von fehlerhaften Entwicklungen vor Produktionsstart
- Nutzen der Migrationszielarchitektur früh herausarbeiten
  - Vorteile des Zielarchitektur können helfen negative Effekte erträglich zu gestalten
- Bessere Flexibilisierung in der neuen Architektur durch Kapselung und sauberes API Design forcieren
  -

Abstrakte Migrationsprojektphasen

Orientation in Objects

- **Bewertung**
  - Projektantrag
  - Kostenschätzung
  - Portfolioanalyse
  - Kosten/Nutzen-Analyse
  - (Contracting)
- Legacy System Understanding
- Target System Development
- Testing
- Migration

Fallstudie Schätzung Green Field COCOMO II

Orientation in Objects

- 75 KLoC
- Basiswert 258 PM
- Skalierbarkeit der Produktivität
- Umfeld Klassifizierung
  - Einschätzung Berater
    - **66 PM**
  - Selbsteinschätzung
    - **53 P**
  - Best Case ?
  - Worst Case ?

Fallstudie Schätzung Brown Field COCOMO II

Orientation in Objects

- Wiederverwendung/Modification
  - Prozentuale Kalkulation von Source Code als Input
  - Komplexere Rechenmodelle
- In der Regel bei 100% Modification ist der Greenfield Schätzwert eine **untere Schranke des Aufwands**
  - Geringeres Risiko gegen höheren Aufwand
  - Inkrementelle Paketumstellung
    - **Migrationsziel wird in der Regel deutlich verzögert erreicht**

Gliederung

Orientation in Objects

- Motivation/Problemstellung
- „klassische“ Lösungsstrategien
- Alternative Denkansätze

Auswahl geeigneter Migrationsvorgehensmodelle/strategie aus Literatur ? (Auszug)

Orientation in Objects

- Gesamt-Projekt
  - V-Modell XT Variante
  - ReMiP
  - Renaissance
- Planung
  - Sneed
- **Umstellungsverfahren**
  - Bing Bang
  - Inkrementelle Paketumstellung
- **Daten**
  - Mikado
  - Butterfly
- **Konkrete technisches Umsetzung**
  - Chicken Little, Database First, Database Last, Composite Database

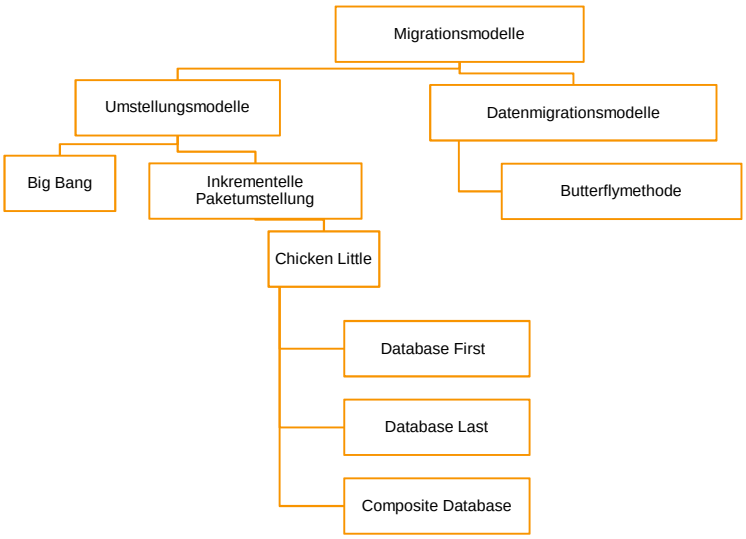
Kriterien bei Migrationsvorhaben

Orientation in Objects

- Migrationsebene
  - Architektur/Sprache/UI/Daten
- Strategie
- Umstellungsverfahren
  - Einstufig
  - Mehrstufig
- Budget
  - Kostenschätzung
- Team
  - Kernteam/Stakeholder/weitere Beteiligte der Umstellung
- Geplante Dauer

Strategien in der Literatur

Orientation in Objects

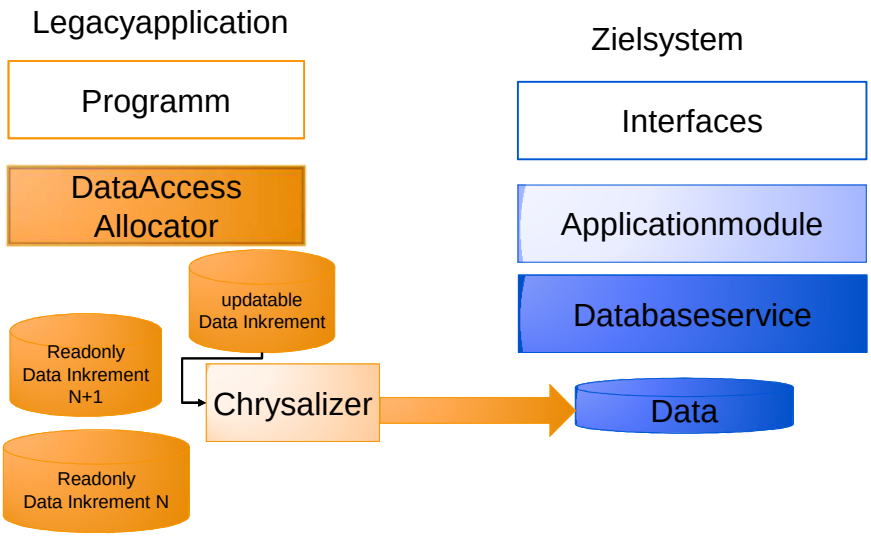


Butterfly

Orientation in Objects

- Reine Datenmigration
- Keine Kooperation zwischen Ziel-/Altssystem
- Trennung der Datenspeicherung im Altsystem in lesend/ändernd notwendig
  - Data Access Allocator
- Chrysalizer zur Transformation der Daten
- Vorteile
  - Migrationsfenster steuerbar
- Nachteile
  - Machbarkeit/Kosten
    - **Data Access Allocator**
    - **Chrysalizer**

Butterflyarchitektur



Big Bang – Cut-and-Run

- Potentiale:
  - siehe Neuentwicklung
  - kürzere Durchführungszeit des Vorhabens
  - Geringerer Abnahmetestaufwand
- Probleme:
  - siehe Nachteile „Cold Turkey“
  - Einsatz eines potentiell instabilen Systems
- Empfehlung
  - kleinere Systeme
  - unkritischere Systeme
  - geringe Zahl von Umstellungen
  - **starkes Management des „Cold Turkey“ !**

Starkes (Sucht)management

Orientation in Objects

- Offene Kommunikation zu Stakeholdern
  - Ökonomische Dimensionen verdeutlichen
  - Entzug als Prozess verstehen
  - Perspektiven aufzeigen
  - Optionen bilden
- Portfolio genau analysieren !

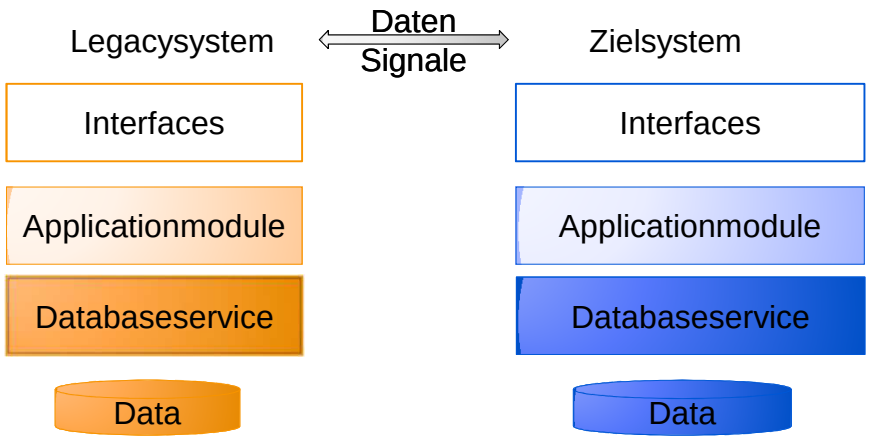
- Alternatives Produkt/System bewußt einsetzen
  - anderes Team
  - Schwächen des Altsystems fokussieren
  - Gefahr komplementärer/divergierender Systeme

Chicken Little

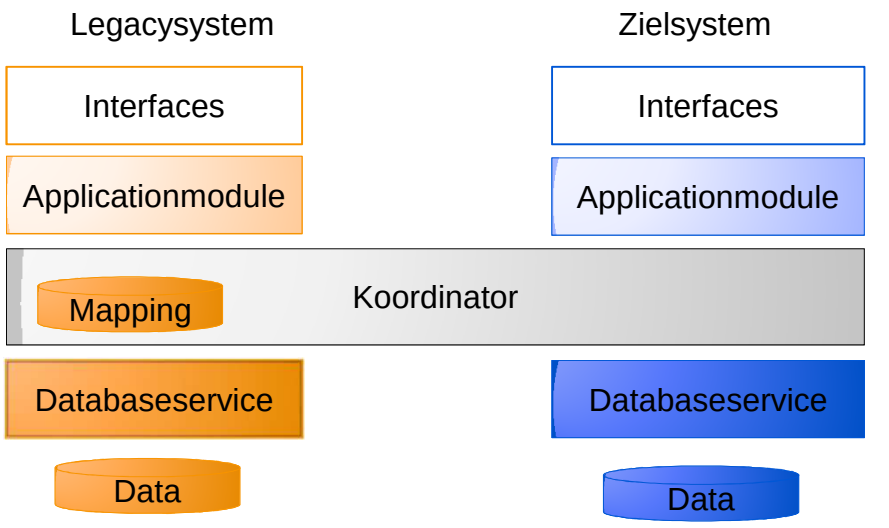
Orientation in Objects

- 11 Schritte Programm (divide & conquer)
  - Analyse des Altsystems
  - Zerlegen des Altsystems
    - Entwickeln der Schnittstellen des Zielsystems !!!
  - Entwickeln der Zielbenutzeroberflächen
  - Entwickeln der Zielanwendungen
  - Entwickeln des Datenbankbackends.
  - Installation der Zielumgebung
  - Entwicklung und Installation von Gateways
  - Migration der Datenbank des Altsystems
  - Migration der Altanwendungen
  - Migration der Benutzeroberfläche
  - Umschalten vom Altsystem auf das Zielsystem

Chicken Little Strategie



Chicken Little Basisarchitektur



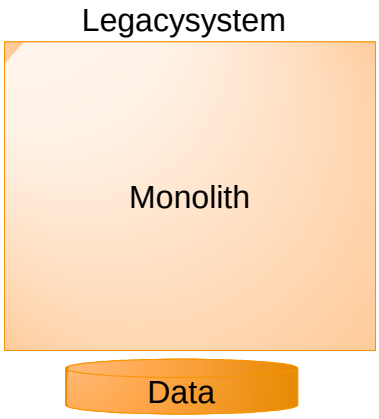
Zentrale Problemstellung bei Chicken Little

Orientation in Objects

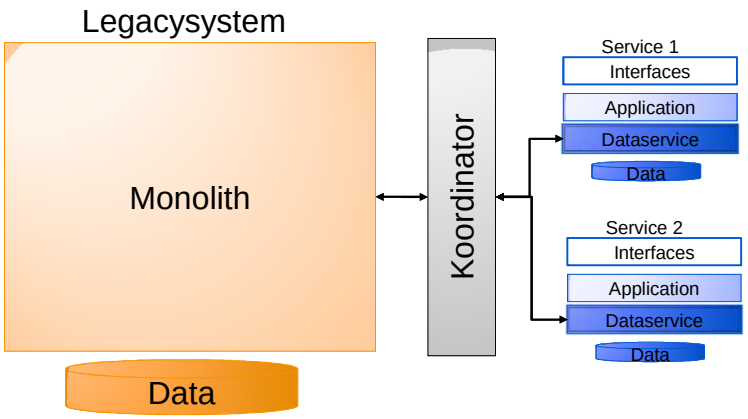
- Zerlegbarkeit des Legacysystems ?
  - Zerlegbares System
  - Teilzerlegbares System
  - Nicht zerlegbares System

Nicht zerlegbares System

Orientation in Objects



Nicht zerlegbares System ?



Chicken Little

- Potentiale:
  - Risikominimierung
  - Schnelles Produktionsfeedback
  - Im Verlauf flexible und anpassbare Strategie
- Probleme:
  - **Kopplung von Legacy/Zielsystem**
  - Ständige QM-Maßnahmen bei jedem Inkrement
  - **Machbarkeit/Kosten**
    - **Coordinatoren(Gateways (forward/reverse)**

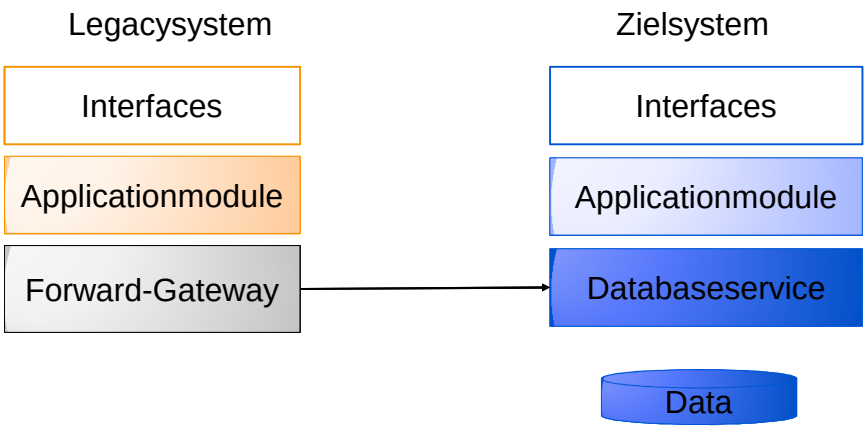
Database First(Forward Migration)

Orientation in Objects

- Vorwärtsmigration
  - Umstellung der Legacysoftware
    - **alle Datenbankzugriffe auf ein Forward Gateway**
    - **Big Bang der Datenbankzugriffe**
  - In der neuen Datenbank kann ein anderes Datenmodell implementiert werden
    - **Forward Gateway muss dann ein Mapping implementieren**
  - Umstellung der Module/UI/Schnittstellen nicht definiert

Database First(Forward Migration)

Orientation in Objects



Database First(Forward Migration)

Orientation in Objects

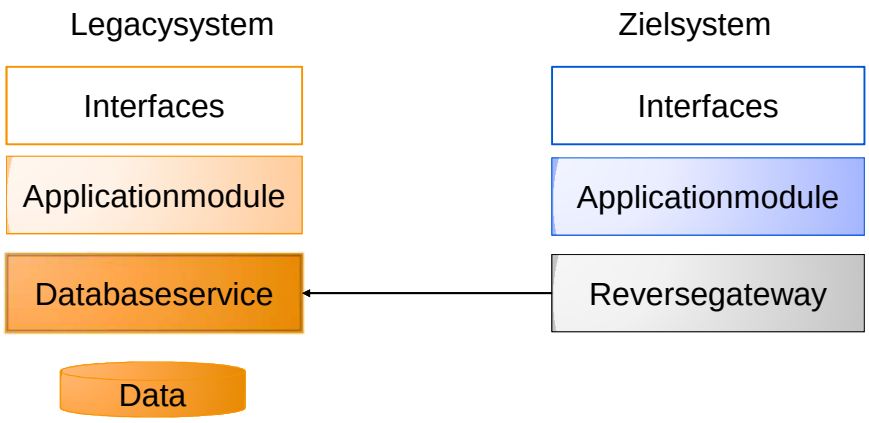
- Potentiale:
  - Nutzung der Technologie der neuen DB
    - **Produktivität, Tools, Konzepte, Sprachen**
  - ev. bessere Performance der neuen Storage
  - Entkopplung von Altsystem und späterer Neuentwicklung
- Probleme:
  - **(Nicht)Existenz eindeutiger Datenzugriffsschnittstellen**
  - **Aufwand Entwicklung Forward Gateway ?**
  - Ev. Performanceprobleme Gateway
  - Externe Zugriffe auf Legacy DB

Database Last (Reverse Migration)

Orientation in Objects

- Rückwärtsmigration
  - Datenbank bleibt im Altsystem
  - Neuentwicklung im Zielsystem greift über ein Reverse Gateway zu
  - Eigentliche Datenmigration wird in dieser Strategie nicht definiert

Database Last (Reverse Migration)



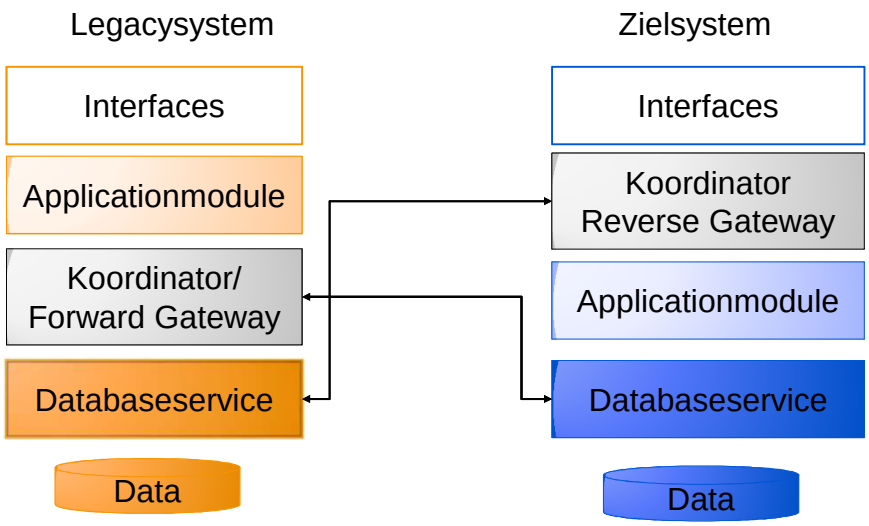
Database Last (Reverse Migration)

- Potentiale:
  - Entkopplung von Altsystem und späterer Neuentwicklung
  - Altssystem sicher verwendbar bis zur DB Migration
  - geringeres Risiko nicht erkannter Requirements
  - Abhängigkeiten können langsamer exploriert werden
  - Nutzung kommerzieller Gatewayprodukte
- Probleme:
  - Datenmodel der Legacyanwendung im Zielsystem weiter nutzen ?
    - **Produktivität**
    - **Performance**
  - **Aufwand Entwicklung Reverse Gateway falls ein neues Datenmodell in der Zielarchitektur umgesetzt wird**
  - Ev. Performanceprobleme Gateway

Composite Database

- Parallel Database Strategy
  - Neue Anwendungen auf neuer Datenbank
    - **Datenduplikation**
  - Koordinator mit Forward und Reverse Gateway

Composite Database Basisarchitektur



Composite Database

Orientation in Objects

- Potentiale:
  - Vorteile von Forward und Reverse Migration
  - Entkopplung von Altsystem und späterer Neuentwicklung
  - Altssystem sicher verwendbar bis zur DB Migration
  - geringeres Risiko nicht erkannter Requirements
  - Abhängigkeiten können langsamer exploriert werden
  - Nutzung der Technologie der neuen DB
    - **Produktivität, Tools, Konzepte, Sprachen**
    - **ev. bessere Performance der neuen Storage**
- Probleme:
  - **Aufwand/Machbarkeit Forward/Reverse Gateway/Koordinator da Synchronization/Konsistenz**

Gliederung

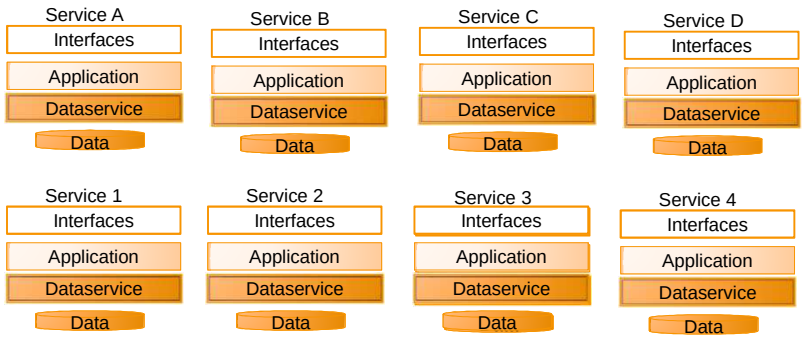
Orientation in Objects

- Motivation/Problemstellung
- „klassische“ Lösungsstrategien
- Alternative Denkansätze

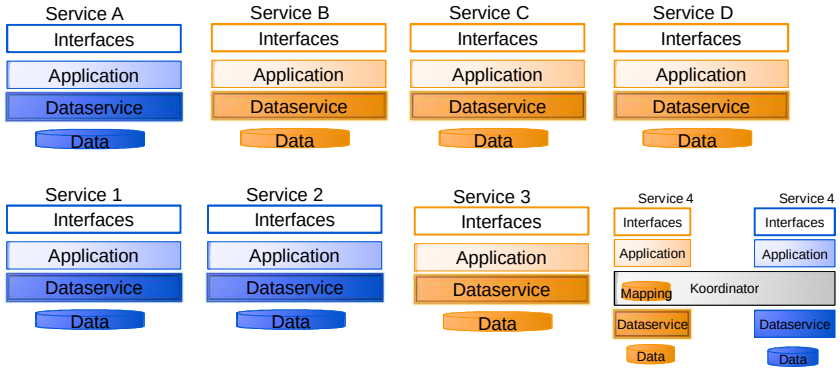
Probleme

- **Chicken Little (alle Ansätze)**
  - **Machbarkeit/Kosten**
    - **Coordinatoren(Gateways (forward/reverse)**
    - **Kopplung von Legacy/Zielsystem**
    - **Ständige QM-Maßnahmen bei jedem Inkrement**
    - **Ev. Performanceprobleme Gateway**
- **Forward Migration**
  - **Nicht-Existenz eindeutiger Datenzugriffsschnittstellen**
  - Externe Zugriffe auf Legacy DB

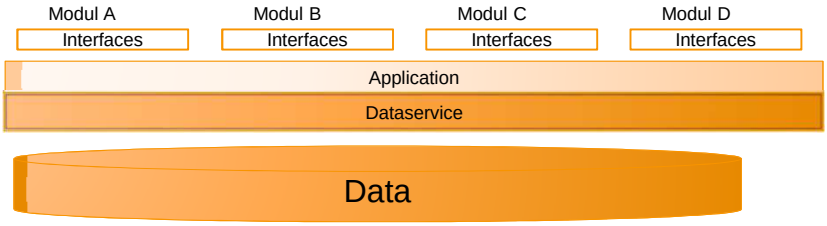
Vollständig zerlegbares System (MicroSOA ?)



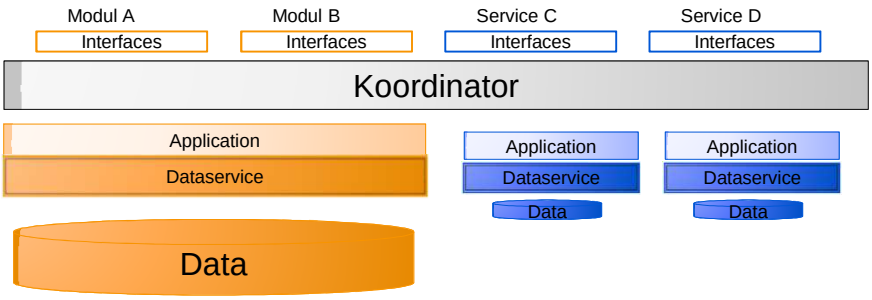
n-tes Migrationsinkrement vollständig zerlegbares System



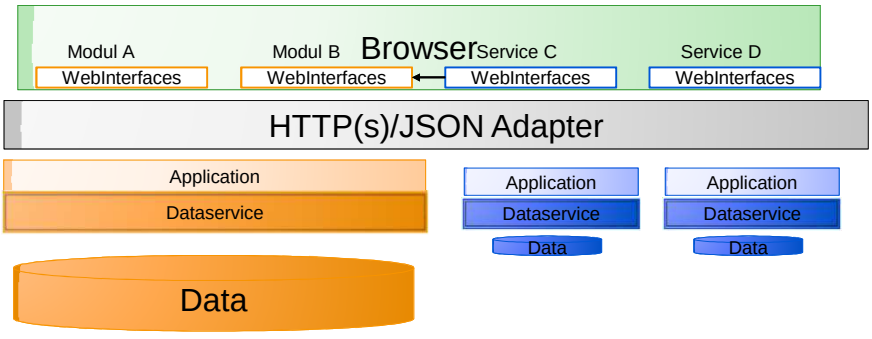
Semi-zerlegbares System



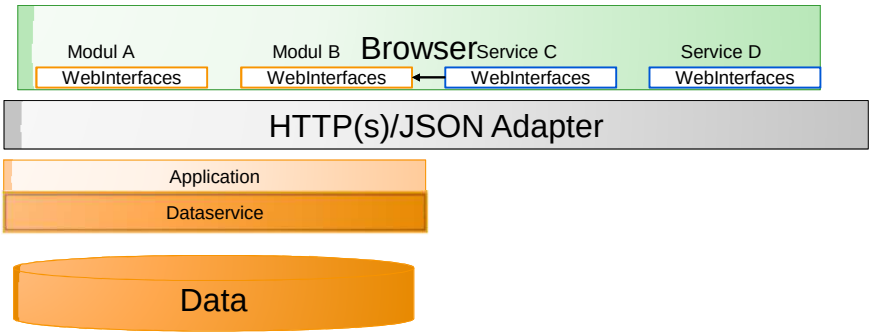
Chicken Little Semi-zerlegbares System



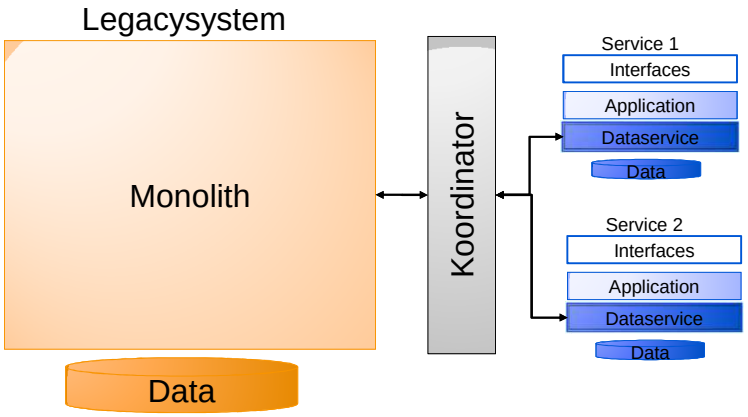
Webservice_Mashup



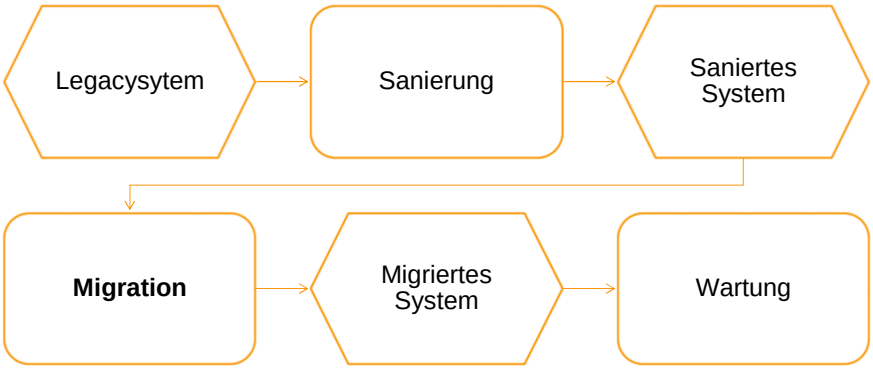
Frontend Migration ?



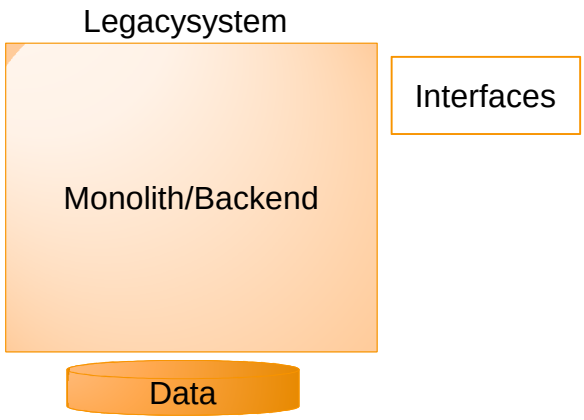
Nicht zerlegbares System ?



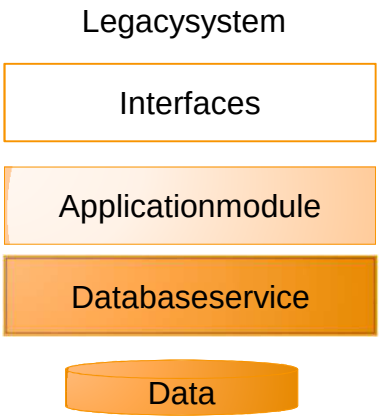
Migration im Wartungsprozess



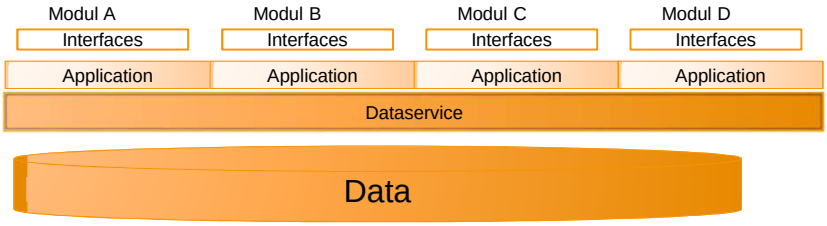
Sanierung Monolith – Macro(web)service



Sanierung Monolith Schichtenbildung

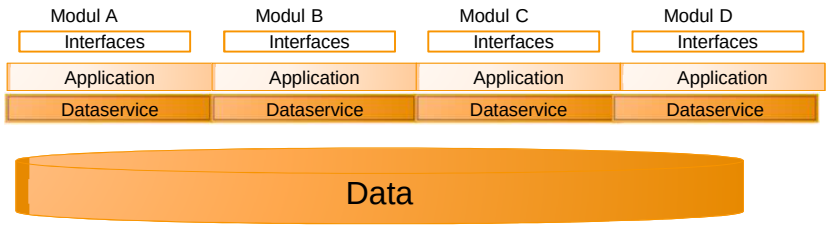


Sanierung Semi-zerlegbares System I



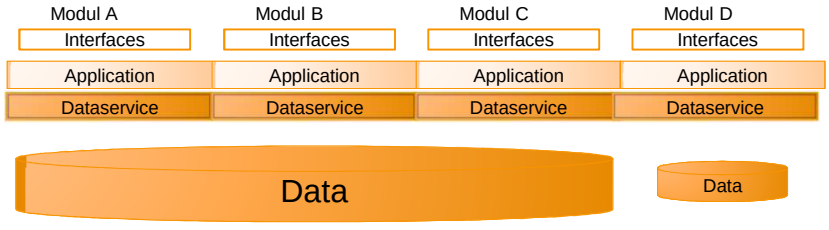
Sanierung Semi-zerlegbares System II (Modularisierung)

Orientation in Objects



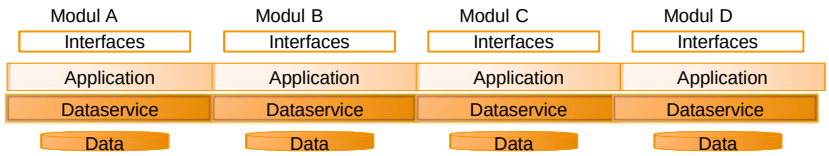
Sanierung Semi-zerlegbares System IIa (Modularisierung)

Orientation in Objects



Sanierung Semi-zerlegbares System III (MicroSOA)

Orientation in Objects



Orientation in Objects

Fragen ?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Orientation in Objects

Vielen Dank für Ihre Aufmerksamkeit !

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Links

Orientation in Objects

- Brodie M., Stonebraker M
  - http://db.cs.berkeley.edu/papers/S2K-93-25.pdf
  - http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Integration-und-Migration-von-IT-Systemen/Ablosung-von-Altsystemen
  - https://de.wikipedia.org/wiki/Migration_(Informationstechnik)
- Ackermann 2005
  - http://www.uni-koblenz.de/ist/documents/ackermann2005da.pdf
- Software Evolution and Maintenance (Englisch) Gebundene Ausgabe – 20. Januar 2015 von Priyadarshi Tripathy (Autor), Kshirasagar Naik (Autor)
  - **http://aim42.github.io/#improve-approaches**