**Vorlesung am 7. Juli 2009**

Serviceorientiertes E-Government

# SOA Governance

**Dr. Frank Sarre**
Lehrbeauftragter der LMU München

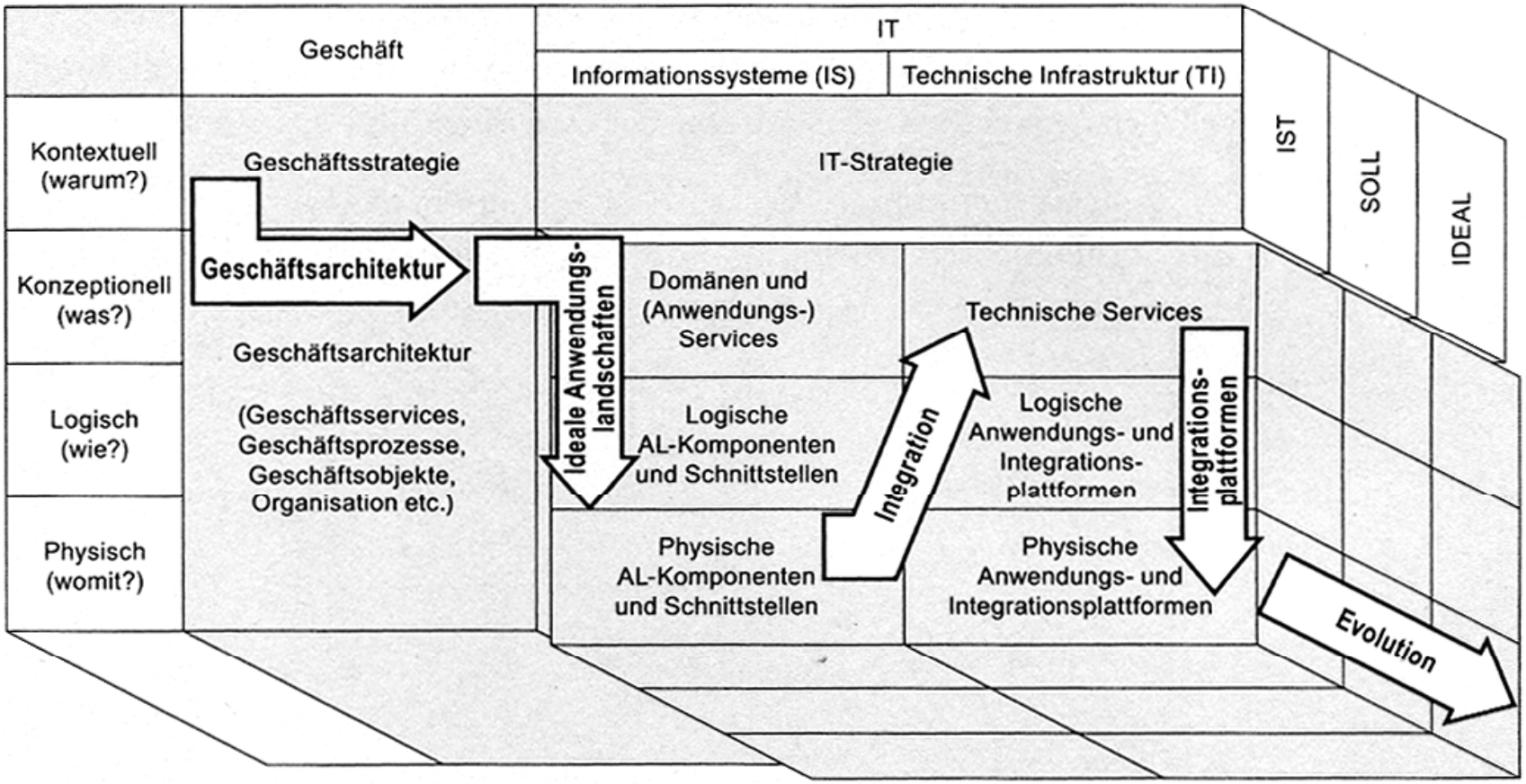
# Wiederholung [1]

## Ziele einer Serviceorientierten Architektur

- Kosteneffizienz
- Effektivität
- Agilität
- Innovation
- Quality of Service

# Wiederholung [2]

## Lösungsweg




Quelle: „Quasar Enterprise“

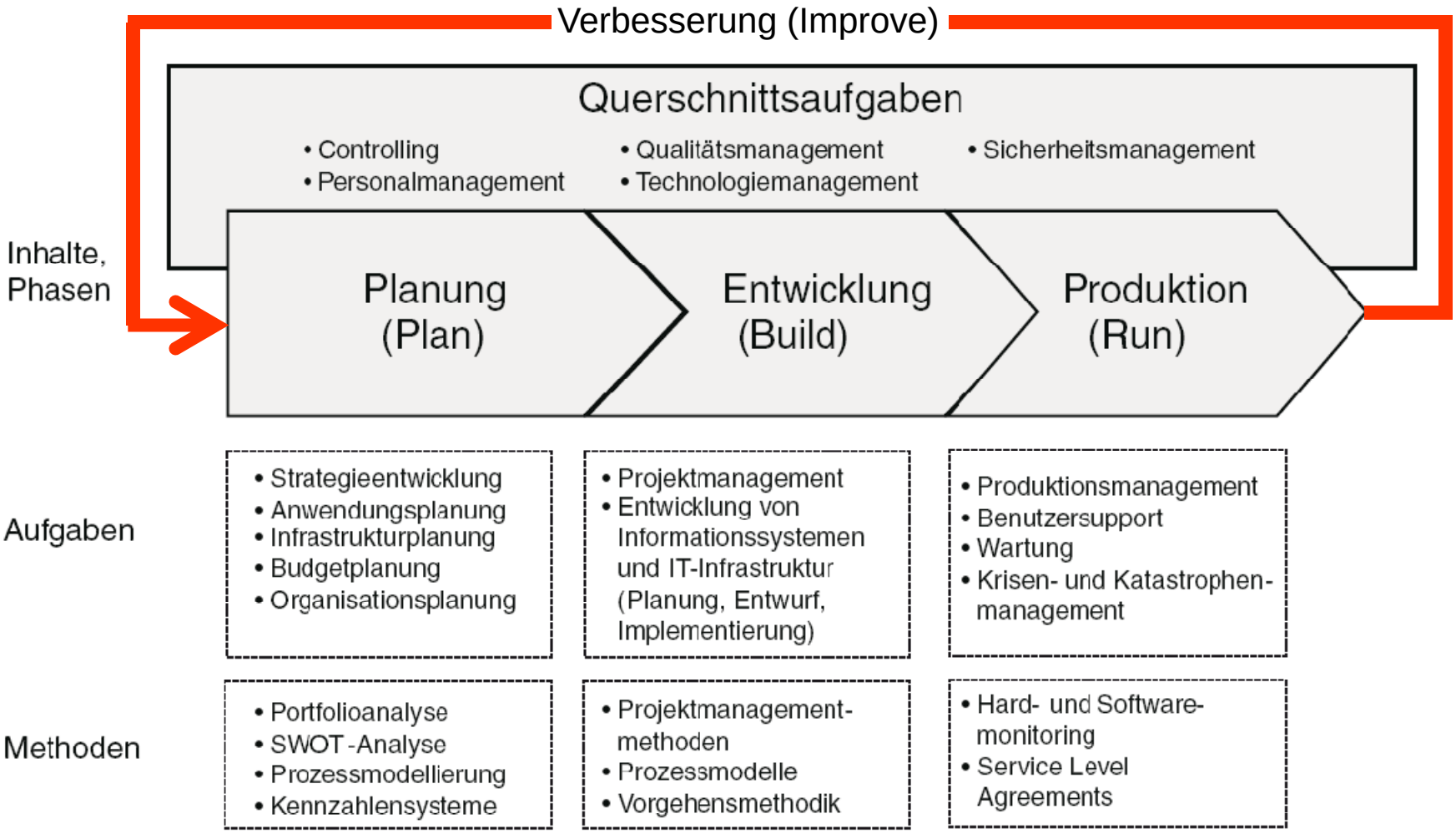
# SOA Governance

- **Kein** eindeutig definierter Begriff !
- SOA Governance betrifft sowohl **organisatorische** als auch **technische Themen**
- Stark vereinfacht beschreibt eine SOA Governance:
  - **was** zur Einführung und Aufrechterhaltung einer SOA zu tun ist und
  - **wie** die einzelnen Beteiligten dies zu tun haben.
- SOA Governance ist ein **Teil des Informationsmanagements** einer Behörde (oder eines Unternehmens)

Interessante Definition der Software AG (für Manager)

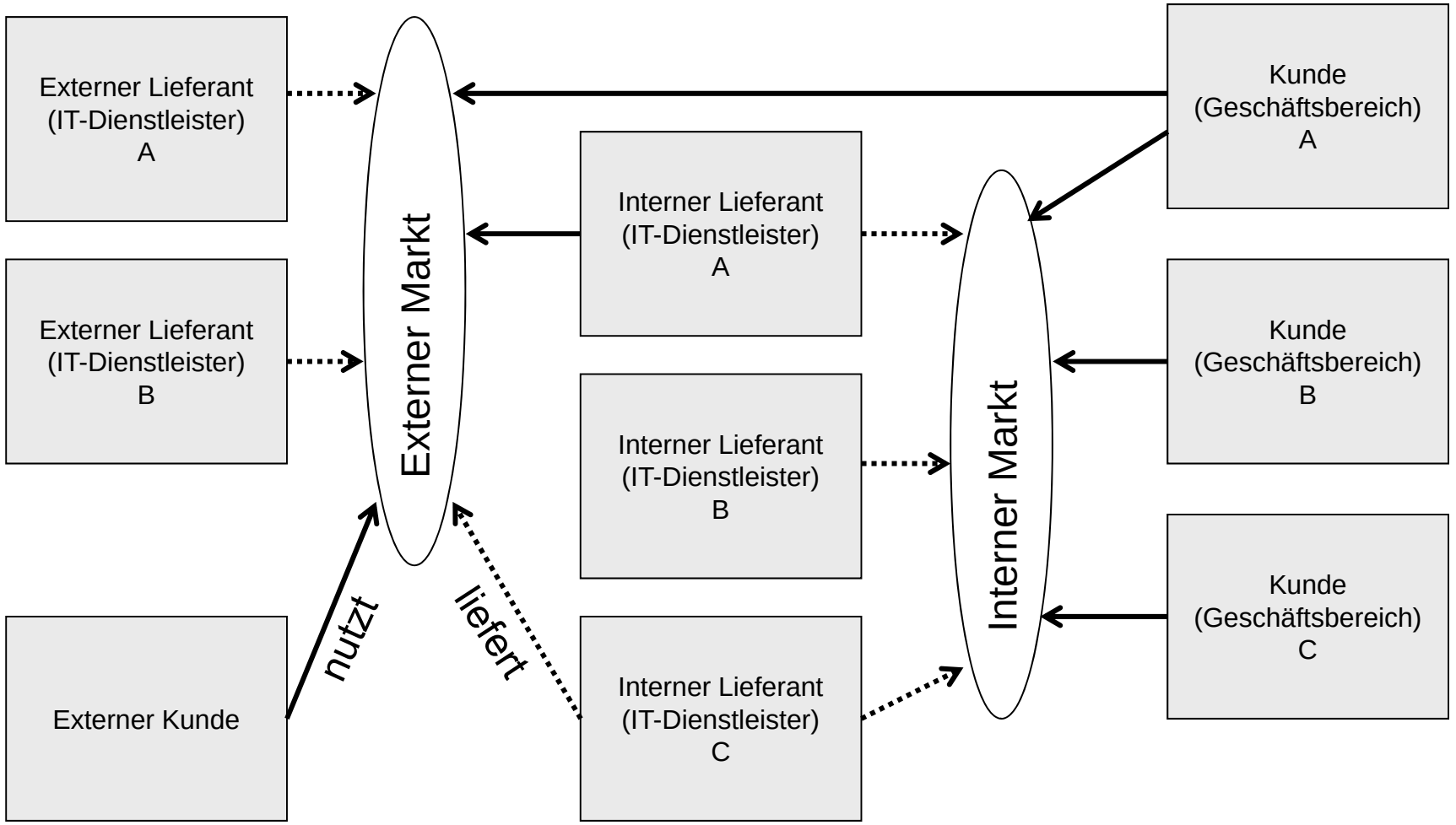
- Eine **SOA** ist wie ein LEGO-Baukasten. Die Blöcke haben verschiedene Größen, Formen und Farben, lassen sich aber in vielfältiger Weise kombinieren.
- **SOA Governance** ist das Verfahren, die **richtigen LEGO-Blöcke** zu produzieren und diese in geeigneter Weise anzubieten.

# Informationsmanagement [1]



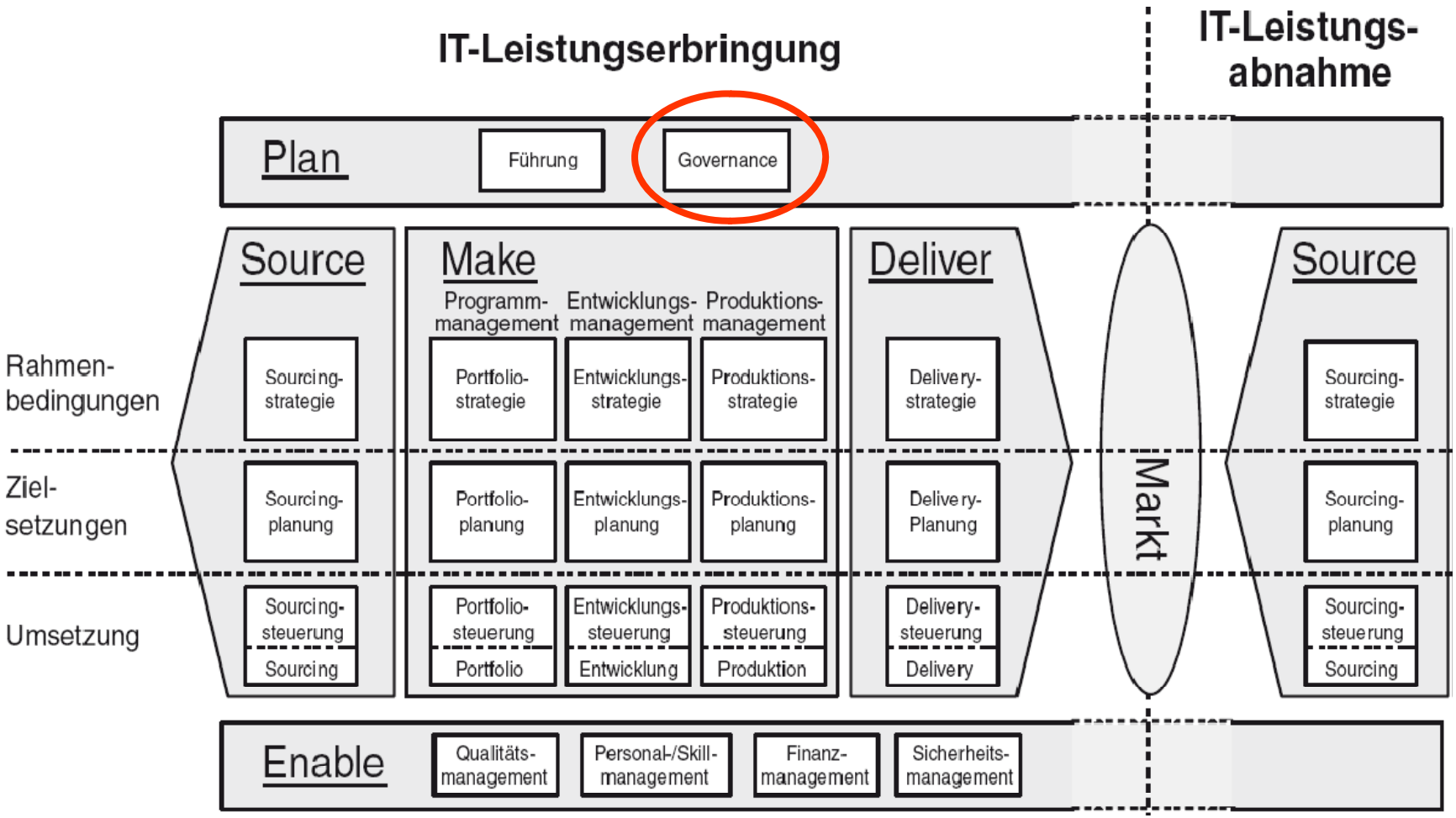

Quelle: Zarnekow, Integriertes Integrationsmanagement – Vom Plan, Build, Run zum Source, Make, Deliver

## Kunden-/ Lieferantenbeziehungen




Angelehnt an: Zarnekow, Integriertes Integrationsmanagement – Vom Plan Build Run zum Source, Make, Deliver

# Informationsmanagement [3]




Quelle: Zarnekow, Integriertes Integrationsmanagement – Vom Plan Build Run zum Source, Make, Deliver

# Beispielhafte Probleme in der Praxis

- Der Modellierungsprozess lässt existierende Services außer Acht
  → zu **geringe Wiederverwendung**
- Fehlende Qualitätssicherungsprozesse, bevor ein Service für den Betrieb freigegeben wird
  → **mangelhafte Servicequalität**
- Fehlender Gesamtblick, wie das Geschäft und die EDV verzahnt sind
  → **ineffiziente Organisation und Servicenutzung**
- Keine Berücksichtigung von Auswirkungen, wenn Dienste geändert werden
  → **Versionskonflikte, unzureichende Servicequalität**
- Fehlende Verantwortlichkeiten bezüglich der Schaffung von Diensten und ihrer Nutzung
  → **unkoordinierter Software Lifecycle**
- Die Einhaltung von Richtlinien wird nur manuell sporadisch und unstrukturiert kontrolliert
  → **unzureichende Sicherheit, erhöhte Aufwände**

# Beispielservice: Adressdatenmanagement

Ein Service bietet Methoden für die **Pflege und Verwendung von Adressdaten** in einer Behörde.

Dieser Service wird von folgenden Anwenderkreisen genutzt:

- Finanzexperten
- Rechtsabteilung
- Zentraler Einkauf

→ Der Service wird von verschiedenen Stellen zu unterschiedlichen Zwecken genutzt

→ Die Nutzung unterliegt unterschiedlichen Richtlinien und Rahmenbedingungen

→ Für den effizienten Einsatz des Adressdatenmanagements muss eine Abstimmung hinsichtlich Inhalt, Verwendbarkeit und Zulässigkeit der Verwendung stattfinden.

# Change Management

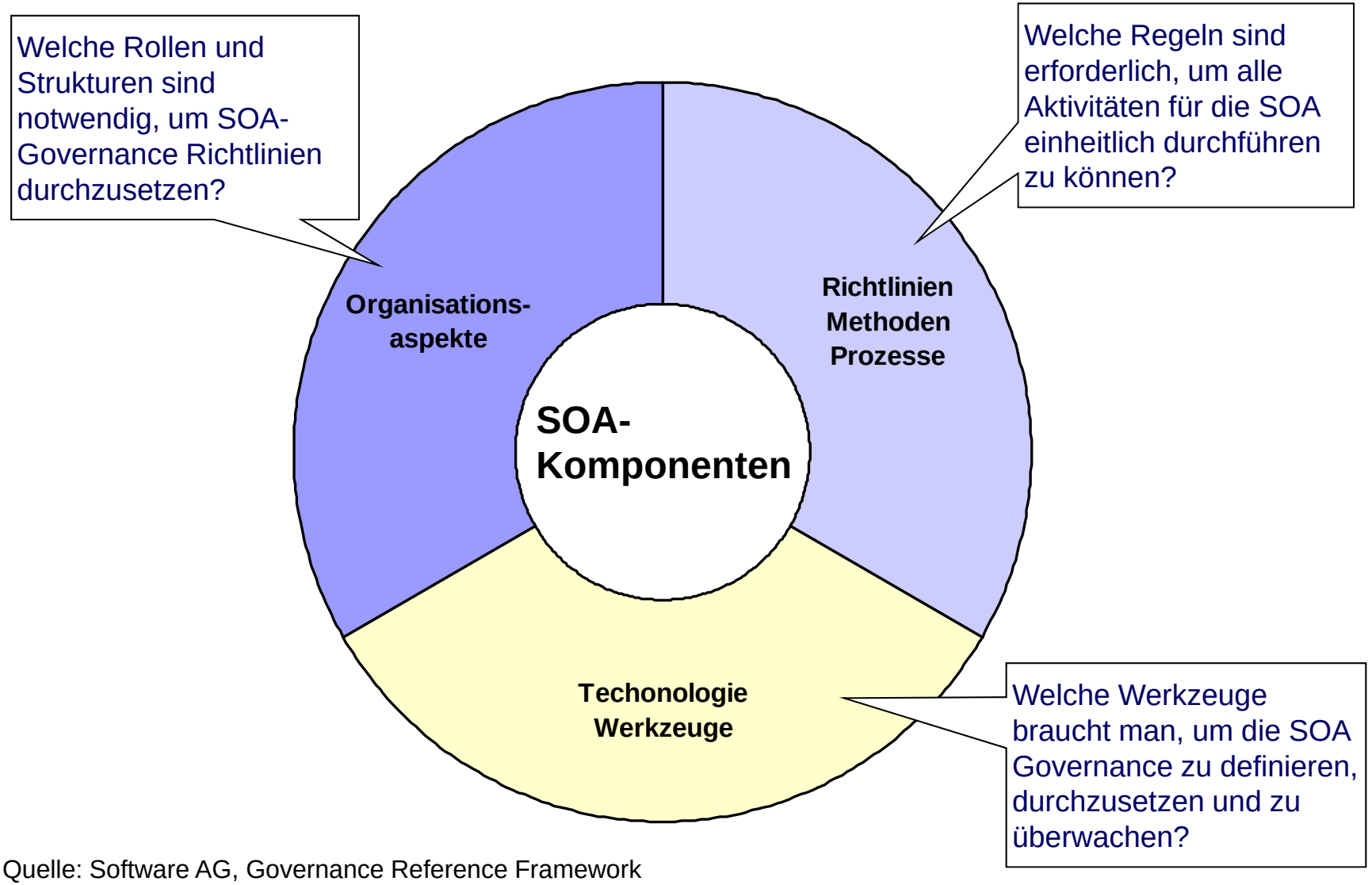
Informationsmanagement beinhaltet auch Methoden und Prozesse, um Veränderungen an bestehenden Strukturen vorzunehmen.

a) Organisatorische Veränderungen
   - Abteilungsstrukturen
   - Aufgabenverteilungen
   - Geschäftsprozesse
   - …

b) Technische Veränderungen
   - Austausch veralteter Systeme
   - Hinzufügen zusätzlicher Systeme
   - Einführung neuer Technologien
   - …

→ **Change-Management**

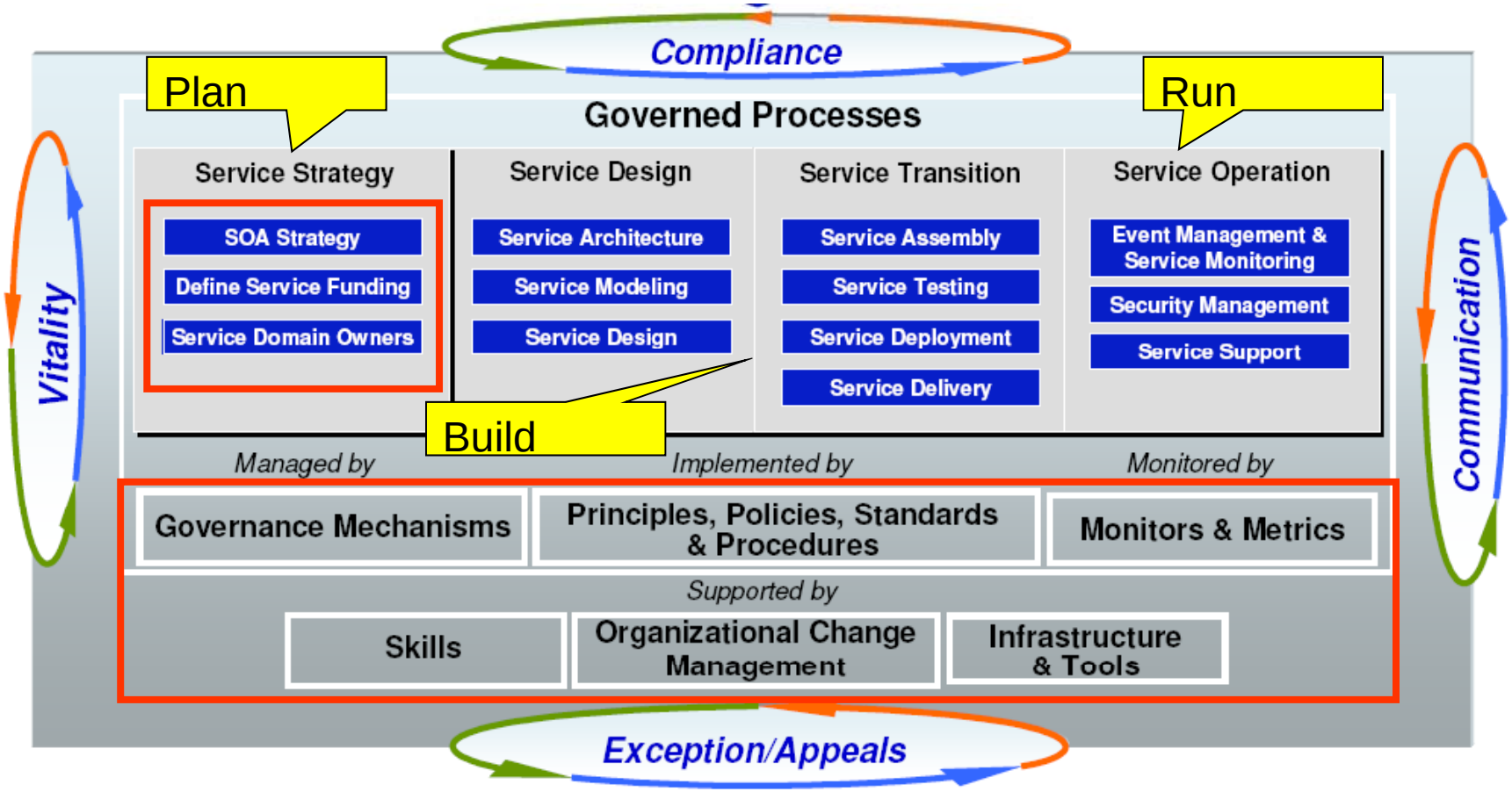
# SOA Governance [1]




Quelle: Software AG, Governance Reference Framework

# SOA Governance [2]

## Governance für einen Service Lifecycle (Vorschlag von IBM)




Quelle: IBM, Advancing the Business/IT linkage with SOA Governance and Service Lifecycle Management

# SOA Governance [3]

## SOA-Strategie

- **Führungsentscheidung**, dass eine SOA umgesetzt werden soll
- Definition **globaler Zielsetzungen** (auch, was **nicht** Ziel sein soll)
- Definition der **Rahmenbedingungen** für eine SOA
- Einrichtung von **Kontroll- und Entscheidungsgremien**
- Festlegung von **Kompetenzen**
- Festlegung von Grundsätzen zur **Finanzierung**

→ **Top-Level-Management**

# SOA Governance [4]

## Domain-Verantwortung

- Zentraler Aspekt für die Umsetzung
- Festlegung, welche Organisationseinheit für die jeweiligen Services verantwortlich ist
- Häufig problematisch, da Geschäftsprozesse bei der SOA-Betrachtung mehrere Organisationseinheiten betreffen
- Erfordert die Bereitschaft der höheren Führungsebenen, sinnvolle Lösungen auch durchzusetzen
- Schlüsselaspekt: Der **Aufwand** und die damit verbundene **Finanzierung** bzw. **personelle Ausstattung**

# SOA Governance [5]

## Kontroll- und Entscheidungsgremien

Aufgaben

- Festlegung von **Detailzielen**
- Festlegung von **Regeln, Methoden und Verfahren**
- Verabschiedung von **Architekturleitlinien**
- Festlegung und Überwachung von **Qualitätskriterien**
- Abstimmung der **Finanzierung**
- **Change Management**

Voraussetzungen

- Hinreichende Kompetenz
- Berechtigung zur Durchsetzung von Regeln (Policy Enforcement)

Problem

- Nicht immer vorhandenes technisches Know-how
- Unzureichende Methodenkenntnisse bei den Verantwortlichen

# SOA Governance [6]

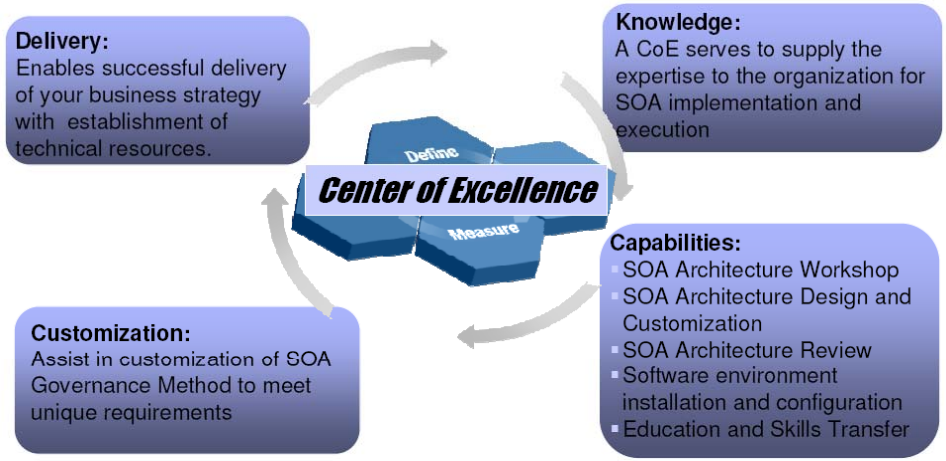
## Grundsätzliche Probleme und beispielhafte Fragestellungen

- Welche **Services** stehen zur Verfügung?
- Wie sind bestimmte **technische Lösungen** zu bewerten?
- Was kann **wiederverwendet** und was muss neu gemacht werden?
- Was ist zu tun, damit **Services wiederverwendet** werden können?
- Wie sieht eine **optimale Architektur** aus?
- Was wären **sinnvolle Schritte** auf dem Weg dorthin?
- Welche Services sind bereits in **Planung**?
- Wie wird die **Leistung/Nutzung** eines Services gemessen?
- Welche **Qualitätsanforderungen** gibt es?

→ **SOA-Kompetenz-Center**

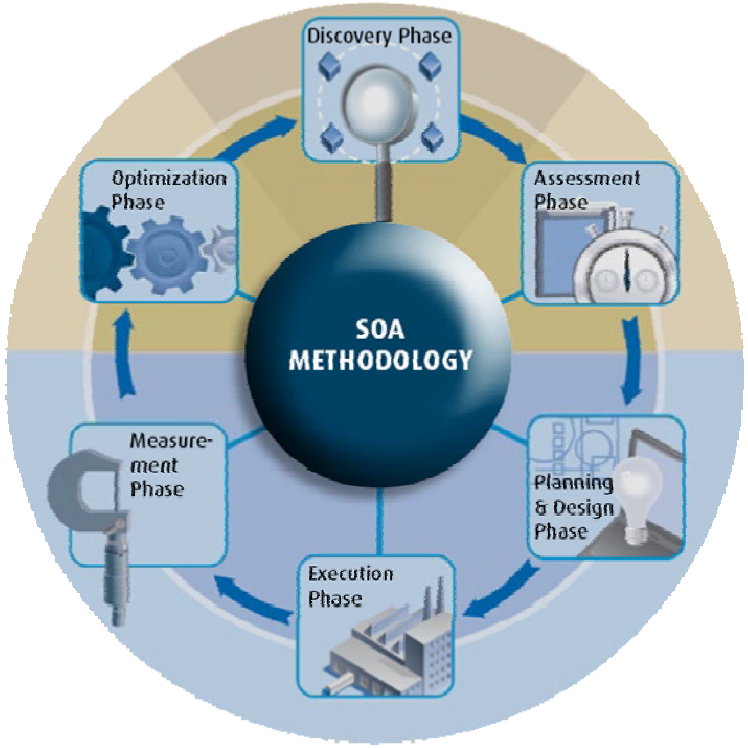
# SOA-Governance [7]

## Kompetenz-Center zur Unterstützung aller Beteiligten




IBM: „Center of Excellence“

Software AG:
SOA Competency Center™



# SOA-Governance [8]

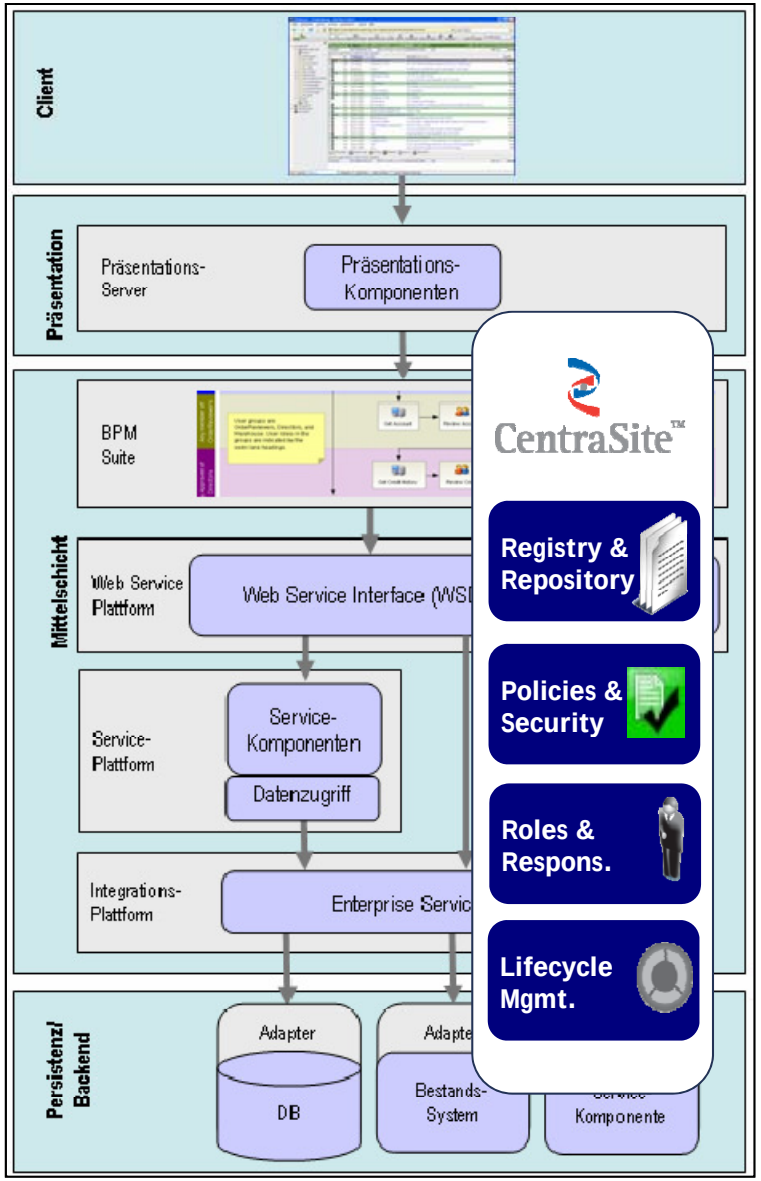
## Typische Aufgaben des SOA-Kompetenz-Centers

- Auswahl und Erarbeitung von **Methoden und Richtlinien**
- **Analyse von Geschäftsprozessen**
- Entwicklung von **Architekturvorschlägen**
- Definition von **Metriken**
- Auswahl von **Werkzeugen**
- **Unterstützung bei der Umsetzung**
- **Unterstützung des Systembetriebs**
- **Ausbildung** aller Beteiligten
- **Beratung** der Verantwortlichen
- …

# Tools zur Unterstützung für SOA Governance

- Top- Anbieter für Tools zur SOA-Governance laut Forrester und Gartner :
  - IBM (Rational Asset Manager, WebSphere Service Registry and Repository)
  - HP (Systinet)
  - Software AG (CentraSite)
- In der Regel **erweiterte Service-Repositorys**
- Keine voll integrierten und vollständigen Produkte, sondern **Tool Suiten**

→ **Software-/Service-Lifecycle Mgmt**



# Zusammenfassung

a) Die unkontrollierte Entwicklung einer SOA verfehlt regelmäßig die grundlegende Zielsetzung einer SOA

b) Die SOA Governance steuert und überwacht die Entwicklung

c) Zentrale Aspekte sind dabei Planung, Umsetzung, Überwachung und kontinuierliche Verbesserung

d) Die SOA-Governance umfasst sowohl die Unternehmens- als auch die IT-Organisation

e) Der Einsatz diverser Methoden des Informations- und Change-Managements ist wichtig!

f) Die anspruchsvollen Aufgaben müssen durch ein Kompetenz-Team unterstützt werden

g) Unterstützung durch Tools ist erforderlich:
In der Regel ein zentrales Repository mit einem darauf aufbauenden Werkzeugkasten

**Vorlesung am 7. Juli 2009**

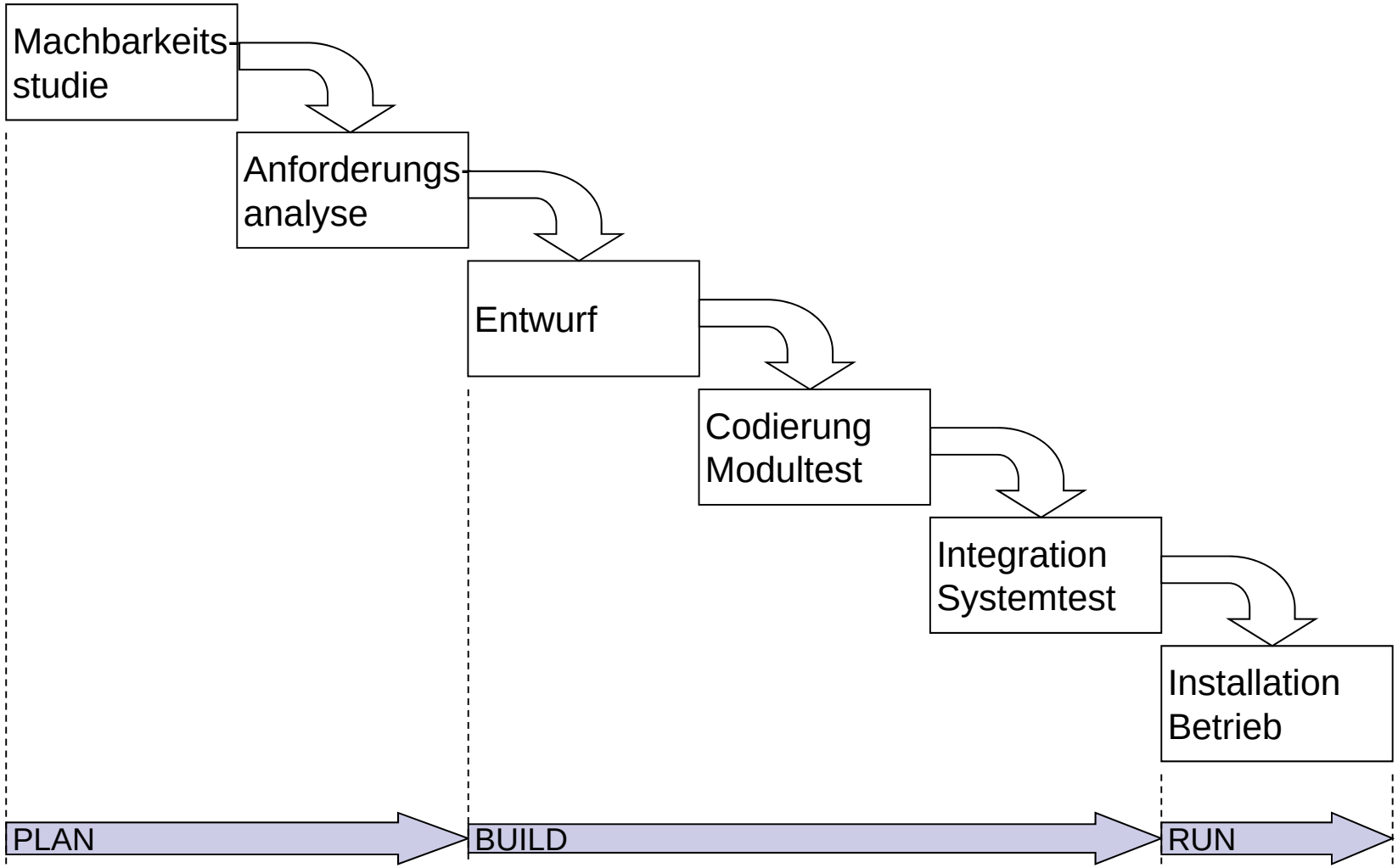
Serviceorientiertes E-Government

# Software Lifecycle mit SOA

**Dr. Frank Sarre**
Lehrbeauftragter der LMU München

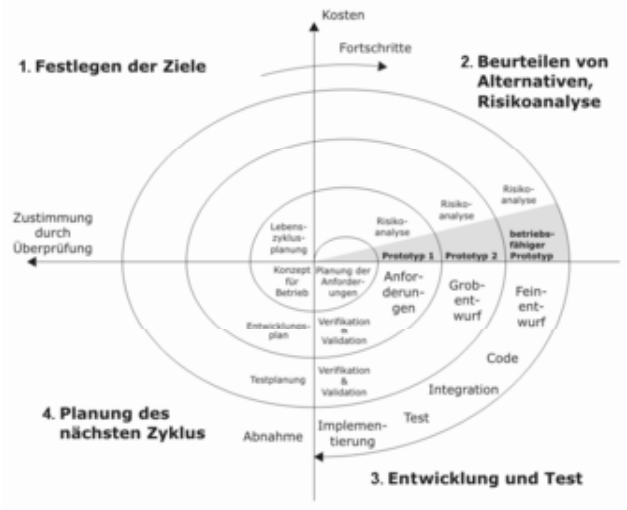
# Software Lifecycles [1]

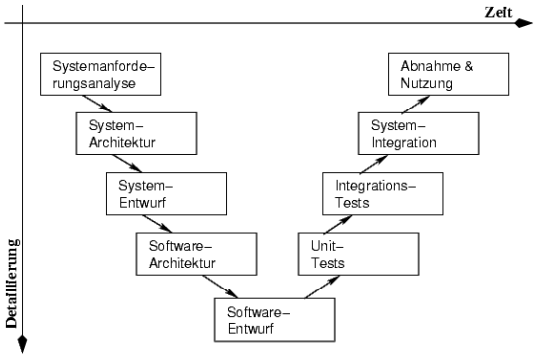
## Klassisches Wasserfallmodell



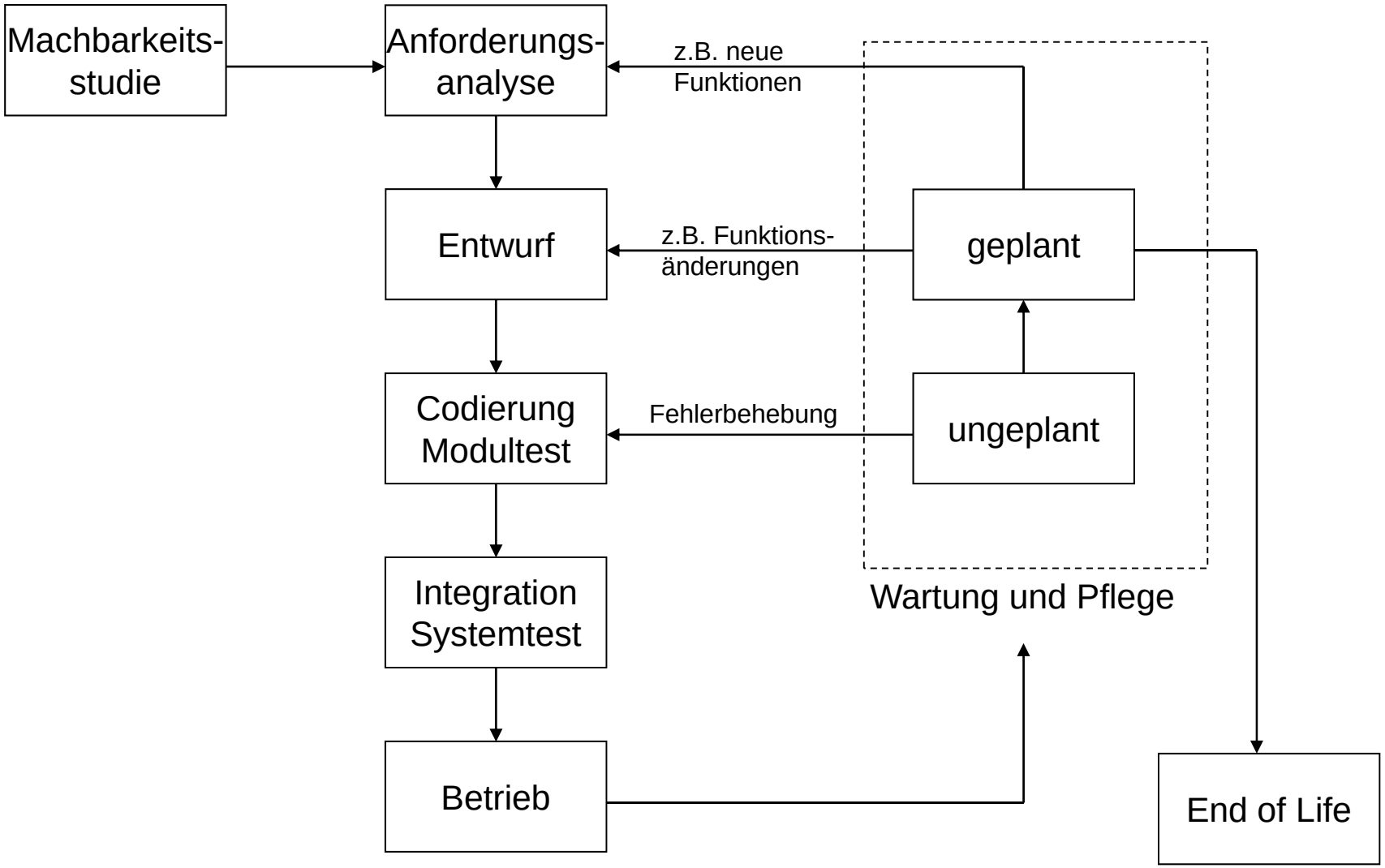
# Software Lifecycles [2]

- (Rapid) Prototyping
- Spiralmodell
- Extreme Programming (XP)
- Agile Software Development
- V-Modell XT
- Capability Maturity Model (CMM)
- Test Driven Development
- Rational Unified Process (RUP)
- …



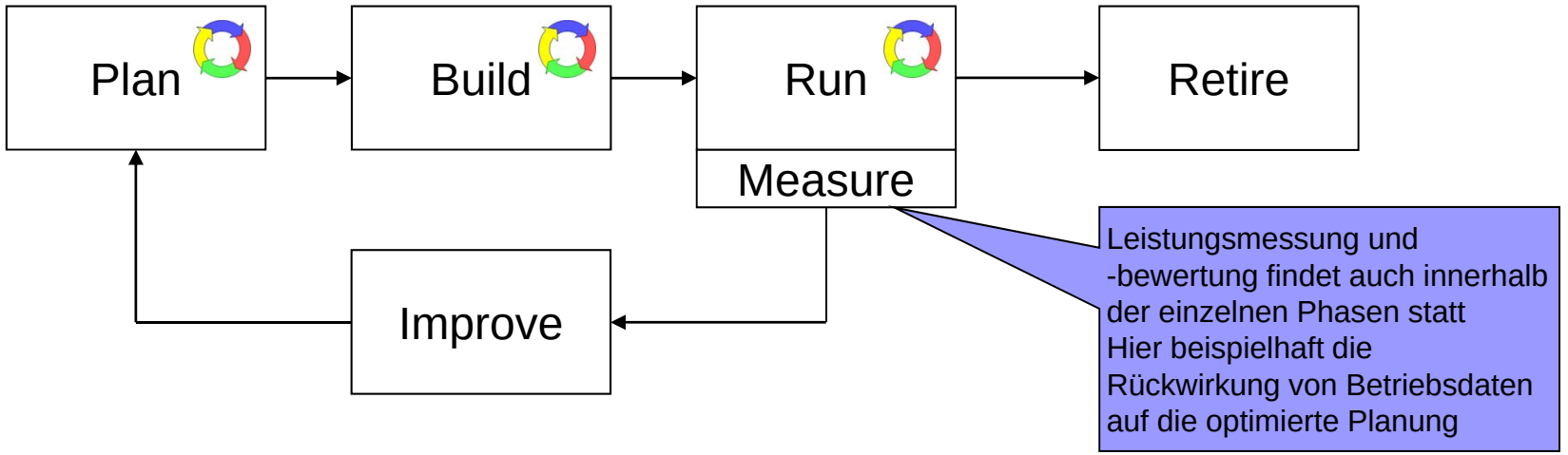

# Software Lifecycles [3]



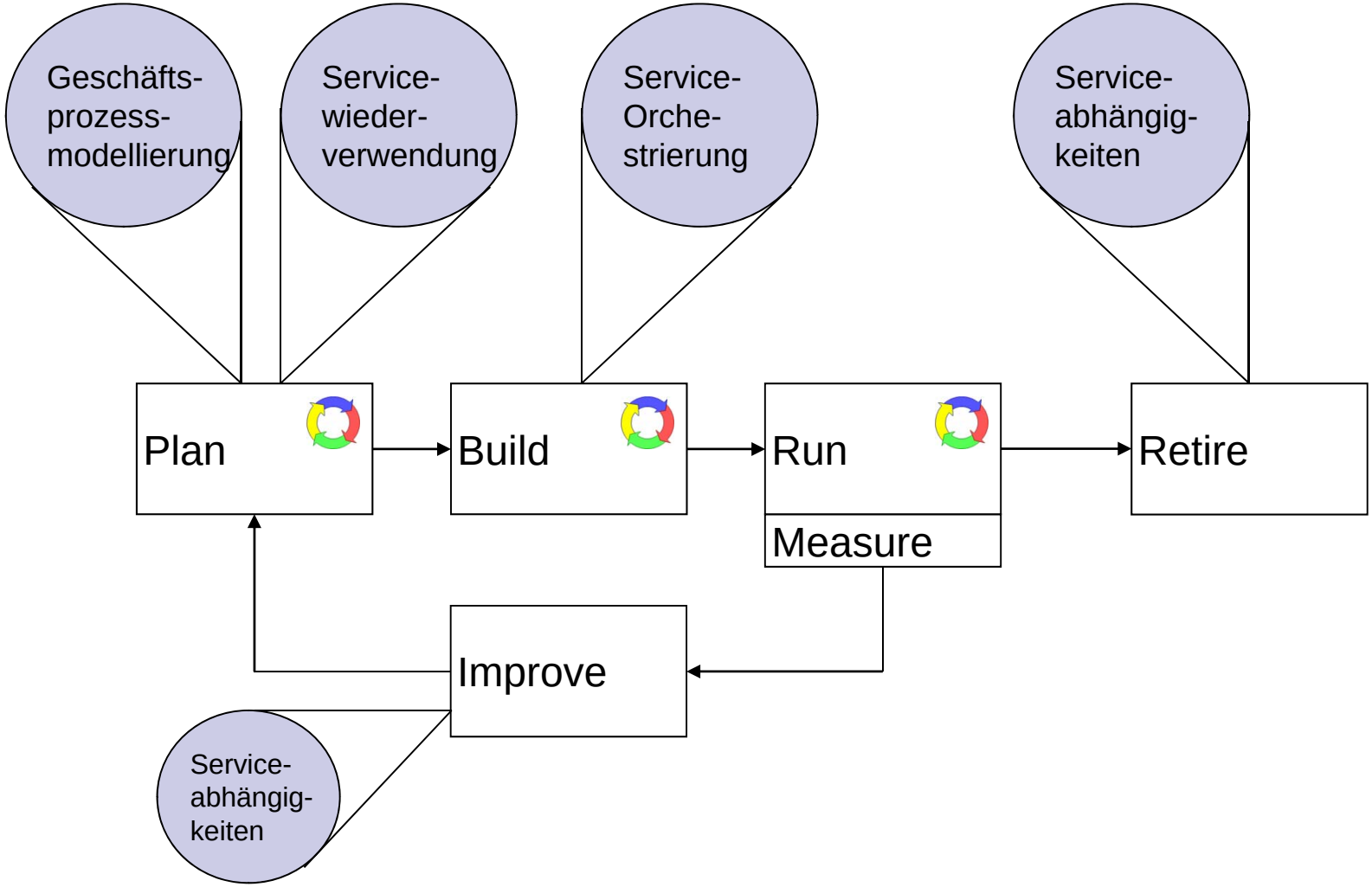
# Software Lifecycles [4]

- In den meisten Entwicklungsmodellen gibt es in der Regel immer gleiche **Phasen**, auch wenn diese im Detail anders ausgeprägt sind
- Die meisten Modelle berücksichtigen **nicht** die Aktivitäten nach der Fertigstellung und Inbetriebnahme (Betrieb, Wartung, …)

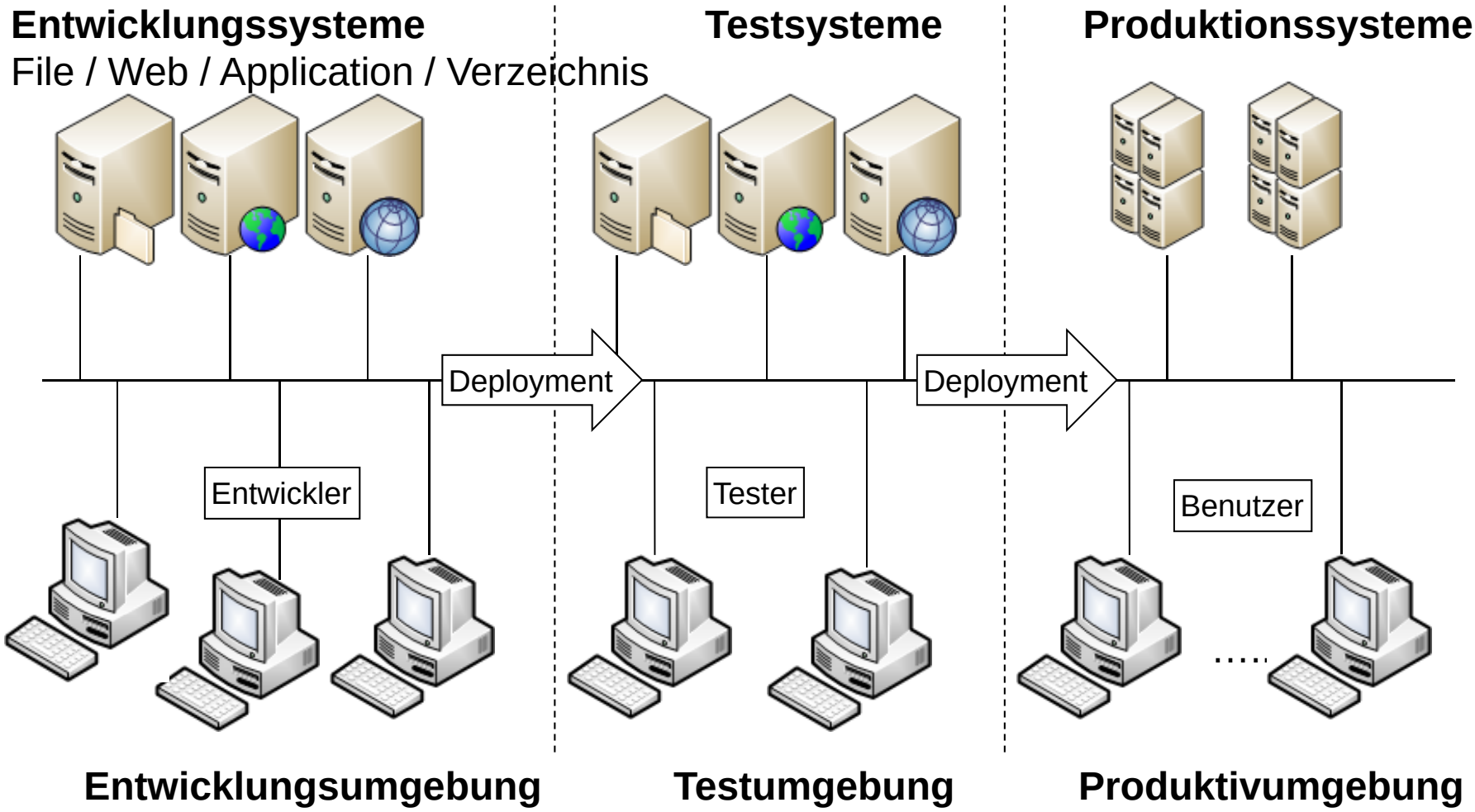
  → Einsatz ergänzender Methoden, z.B. ITIL



## SOA-Einflüsse



## Von der Entwicklung zum produktiven Einsatz



# Software Lifecycle – Produktivsetzung [2]

**Die Implementierung und das Deployment von Software durchlaufen üblicherweise folgende Stufen:**

(1) Entwicklung und Modultest in einer Entwicklungsumgebung

(2) Integrationstests in einer Testumgebung

(3) Produktivsetzung in der Produktionsumgebung

(4) Fallweise wird auch noch eine weitere Stufe, ein sog. *Staging System* vor der Produktivsetzung eingeführt, in der ein möglichst realitätsnahes Abbild der Produktivumgebung für den Integrationstest verwendet wird

## Übliche Probleme

- Verfügbarkeit realistischer Dummys für die Nachbarsysteme bzw. Schnittstellen
- Definierter „Transport“ der entwickelten Komponenten von einer Umgebung in die andere

# Service Lifecycle in einer SOA

## Idealisierte Ausgangslage

- Anwendungslandschaft ist in fachliche Komponenten zerlegt
- Basisservices behandeln Geschäftsobjekte und einfache fachliche Funktionen
- Funktionsservices bilden spezielle Fachanwendungslogik ab
- Prozessservices steuern aufgabenübergreifende Abläufe
- Services werden von unterschiedlichen Betreibern zur Verfügung gestellt
- Services sind teilweise intern und teilweise extern angesiedelt
- Servicenutzung unterliegt definierten Regeln und Kosten
- Services haben eine definierte Leistungsfähigkeit und Verfügbarkeit

# Service Lifecycle in einer SOA – Plan [1]

## Planung neuer Geschäftsprozesse

- Gibt es schon ähnliche Geschäftsprozesse?
- Können diese als Muster verwendet werden?

  → Verwendung von Templates oder „Best Practice“ Ansätzen
- Gibt es Überschneidungen mit anderen Geschäftsprozessen?
- Welche Services werden dort verwendet?

  → Wiederverwendung bestehender Services
- Kann ein Geschäftsprozess komplett auf Basis bestehender Services abgebildet werden oder müssen neue bzw. modifizierte Services geschaffen werden?

Die Abstimmung zwischen der Geschäftsprozessmodellierung und vorhandenen IT-Services ist sinnvoll und notwendig.

# Service Lifecycle in einer SOA – Plan [2]

## Planung neuer Services

- Gibt es schon ähnliche Services?
- Wem gehört ein bestehender Service?
- Zu welchen Konditionen kann man diesen nutzen?

  → Wiederverwendung bestehender Services,
  evtl. durch Modifikation / Erweiterung eines bestehenden Services

- Kann ein neuer Service evtl. für andere nützlich sein?

  → Berücksichtigung erweiterter Anforderungen

- Wer darf neue Services zu welchen Konditionen nutzen?

  → Berücksichtigung des Berechtigungskonzepts

- Welche Service Levels sind erforderlich?

  → Definition eines Quality of Service

# Service Lifecycle in einer SOA – Plan [3]

## Planung neuer Services

- Gibt es Services **in Entwicklung**, die der neue Service bald nutzen kann?
- Gibt es **geplante** Services, die der neue Service in absehbarer Zeit nutzen kann?
- Soll ein potenziell genutzter Service evtl. bald wieder abgeschaltet werden?

  → Wiederverwendung von Services

  → Status der Services muss bekannt sein

# Service Lifecycle in einer SOA – Build [1]

## Entwurf, Implementierung und Test von Services

- Entwickler brauchen Zugriff auf die Serviceinformationen inklusive aller Randbedingungen für die Nutzung
- Wie in der klassischen Softwareentwicklung wird auch hier ein CVS benötigt, das die Entwicklung in einem Team ermöglicht
- Für Modultest sind geeignete Testframeworks erforderlich
- Für Integrationstests müssen Service-Dummys bzw. Kopien der echten Services zur Verfügung stehen
  - → Wer liefert die Dummys?
  - → Müssen mit jedem Service immer auch Testschnittstellen entwickelt werden?

# Service Lifecycle in einer SOA – Build [2]

## Entwurf, Implementierung und Test von BPEL-Prozessen

- Bei der Entwicklung von BPEL-Prozessen werden die Servicebeschreibungen aller genutzten Services benötigt

  → Zugriff auf Verzeichnisdienste

- Beschreibungen müssen beim Entwurf und bei der Implementierung noch nicht vollständig sein, z.B. muss binding und port erst zur Laufzeit verfügbar sein

  → Berücksichtigung von Services im Planungs- und Entwicklungsstadium

- Für den Funktionstest müssen binding und port auf Testservices „gemapt“ werden

  → Dummy-Services bzw. Kopien der echten Services erforderlich

  → Gibt es Testaccounts? Wer verwaltet diese?

# Service Lifecycle in einer SOA – Build [3]

## Identity- und Rechtemanagement

- Im Rahmen der Entwicklung müssen **berechtigte Nutzer** und die entsprechenden Zugriffsrechte berücksichtigt werden
- In **Entwicklungsumgebungen** haben meistens alle Benutzer alle Rechte, die es gibt
- **Testsysteme** spiegeln häufig die reale Produktionsumgebung nicht vollständig wider
- Häufig fallen erst in der Produktion Fehler auf, die aufgrund falscher Berücksichtigung von Rollen und Rechten entstehen

  → Zugriff auf das zentrale Identitätsmanagement

  → Verwaltung von Rollen und Rechten im Entwicklungssystem!

# Service Lifecycle in einer SOA – Run / Measure

- Automatisiertes Deployment von der Testumgebung in die Produktivumgebung
  - → In der Regel skriptgesteuerte Lösungen, z.B. auf Basis von ANT
- Überwachung der Services zur Laufzeit:
  - Antwortzeit
  - Durchsatz
  - Latenzzeit
  - Verfügbarkeit
  - Häufigkeit der Serviceaufrufe
  - Häufigkeit einzelner Prozessschritte
  - Dauer einzelner Prozessschritte
  - …
- Rückkopplung der Ergebnisse zur Planung und zurück in die Entwicklungsumgebung

# Service Lifecycle in einer SOA – Improve

## Modifikationen an Services

- Optimierung von Abläufen
- Korrektur von Fehlern
- Einbringen neuer Funktionen

Wichtige Fragestellungen dabei:

- Welche anderen Services nutzen den zu ändernden Service?
- Was passiert, wenn Schnittstellen geändert werden?
- Was passiert, wenn die Verfügbarkeit geändert wird?
- Was passiert, wenn Nutzungsberechtigungen geändert werden?

→ Servicenutzer und Abhängigkeiten müssen bekannt sein

# Service Lifecycle in einer SOA – Retirement

## Abschaltung von Services

- Kann ein Service überhaupt abgeschaltet werden?
- Welche Services nutzen den Service sonst noch?

  → Servicenutzer und Abhängigkeiten müssen bekannt sein

- Ein Service kann auch durch einen neuen Service ersetzt werden

  → Auch hier müssen die Abhängigkeiten klar sein,
  wenn sich z.B. die Schnittstelle ändert

# Repository [1]

**Service Repository zur Verwaltung von**

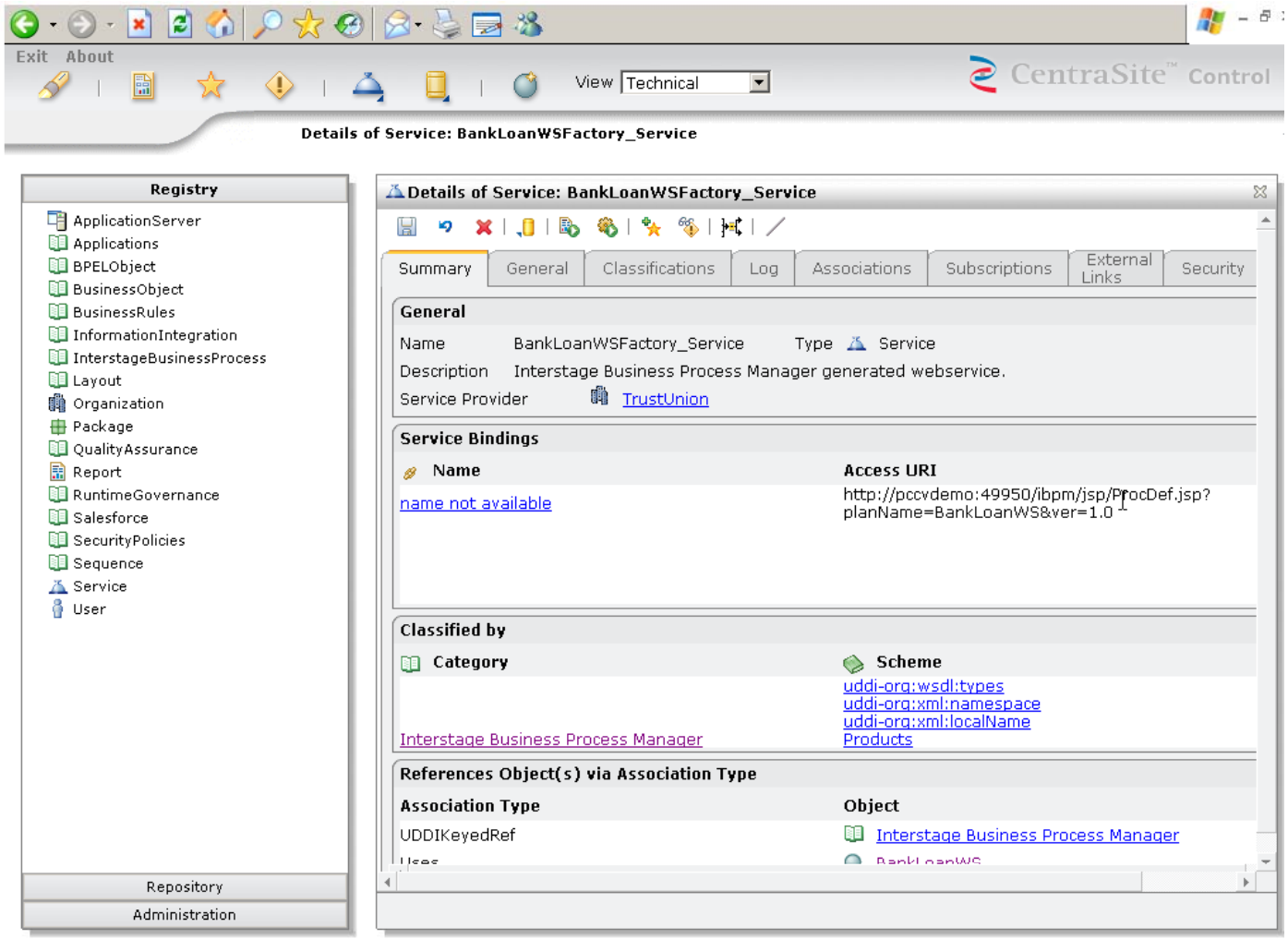
- Servicedefinitionen
- erweiterte Serviceattribute (Metadaten)
- Serviceabhängigkeiten
- Benutzer
- Rollen / Rechte
- Service Level Agreements
- Security Policy
- Monitoringdaten
- …

Repository allein ist nicht hinreichend!


Beispiel:
CentraSite der Software AG

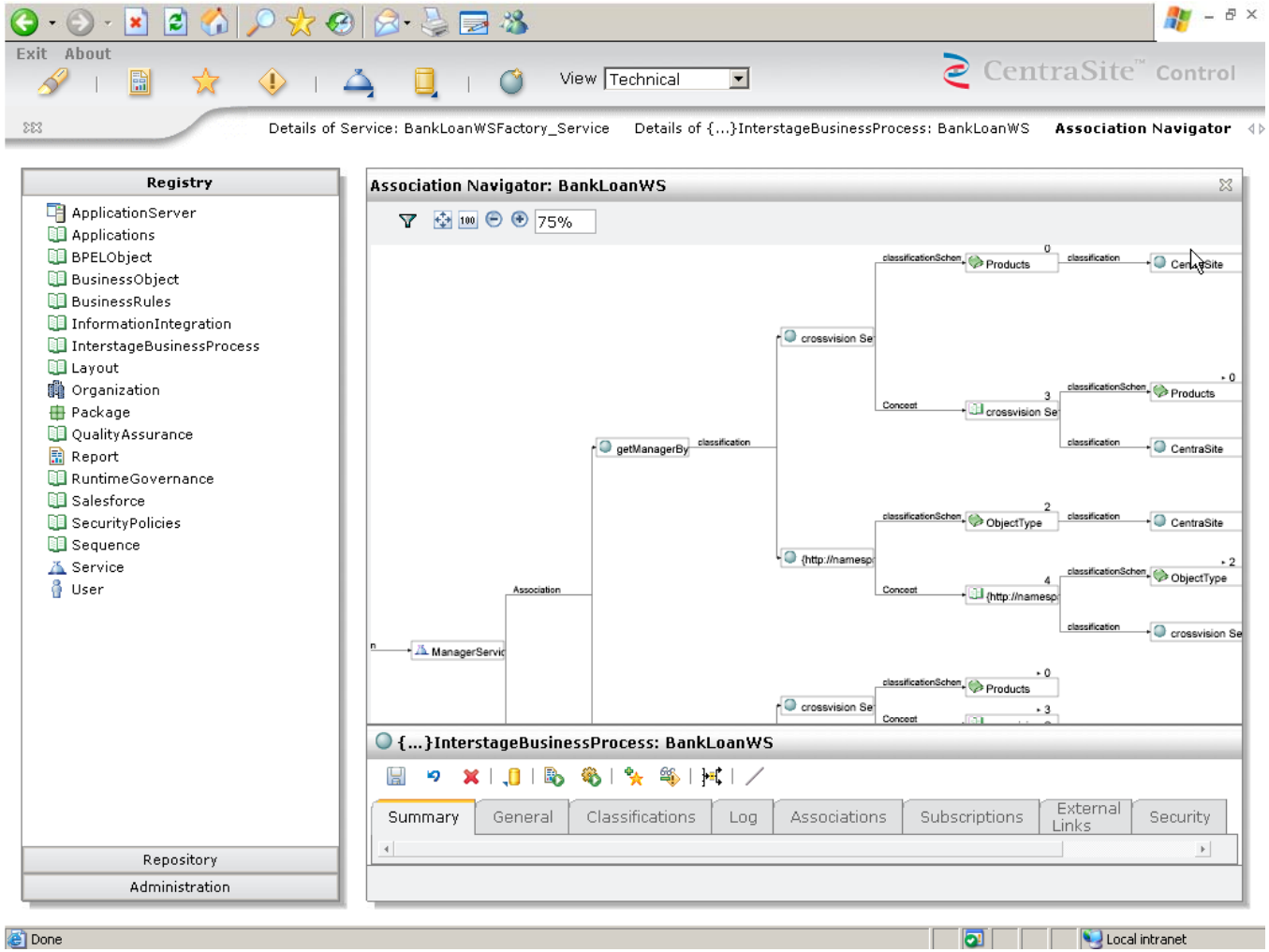
# Repository [2]

## Repository liefert Informationen für SOA-Architekten



# Repository [3]

## Beispiel: Darstellung von Service-Abhängigkeiten



# Werkzeuge [1]

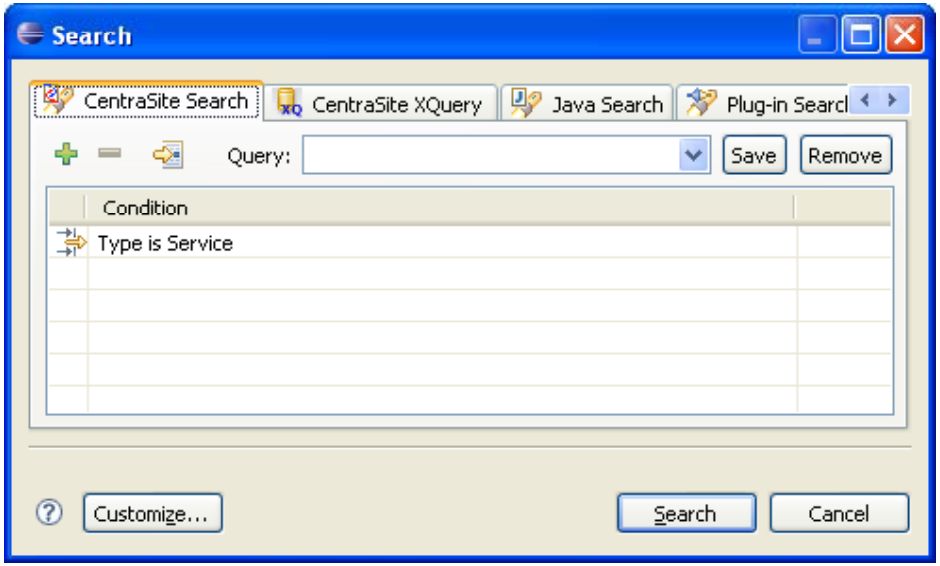
**Was wird neben dem Repository noch benötigt?**

- Das Repository ist das Werkzeug für SOA-Architekten und Planer
- Entwickler, Tester und Betreiber der Services und Prozesse brauchen weitere Werkzeuge
  - Programmierumgebungen
  - SOA-Test-Frameworks
  - Management- und Monitoring-Tools
  - …

# Werkzeuge [2]

## Programmierumgebungen

- → Eclipse, JDeveloper, Visual Studio, …
- → Integration mit dem Repository erforderlich, um die Informationen im Entwicklungsprozess abgreifen zu können
- → Aktualisierung der entwickelten Services im Repository

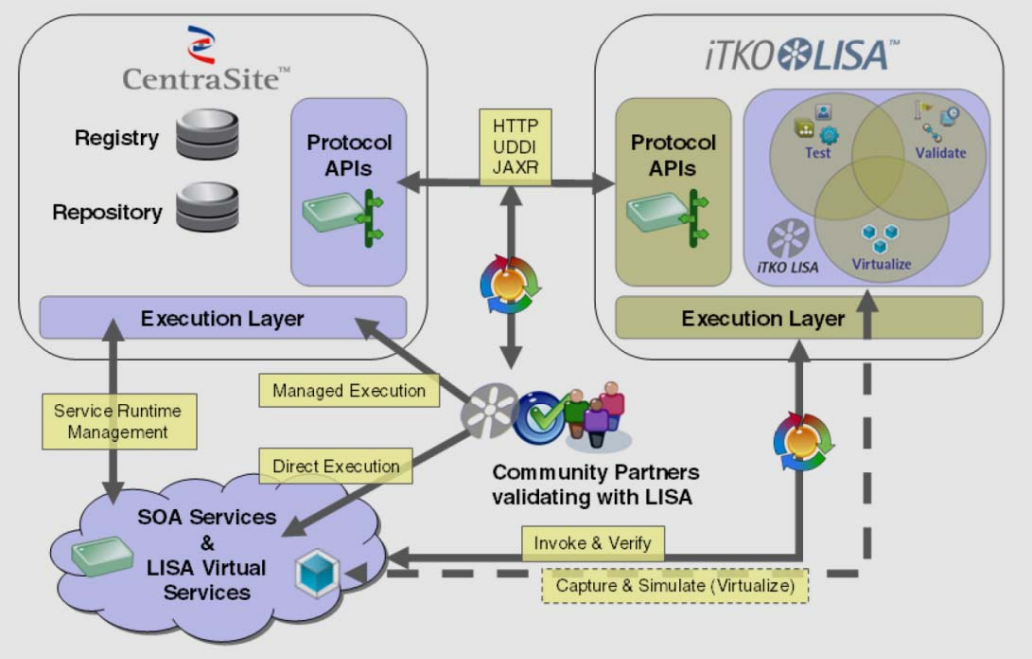



Beispiel Service Query in Centrasite über Eclipse

# Werkzeuge [3]

## SOA-Test-Framework

- → iTKO LISA, IONA Interface Simulation and Testing Framework …
- → Besonderheit gegenüber herkömmlichen Test-Suites ist der intensive Integrationstest mit Serviceschnittstellen
- → Integration zum Service Repository für die Nutzung vorhandener Serviceinformationen

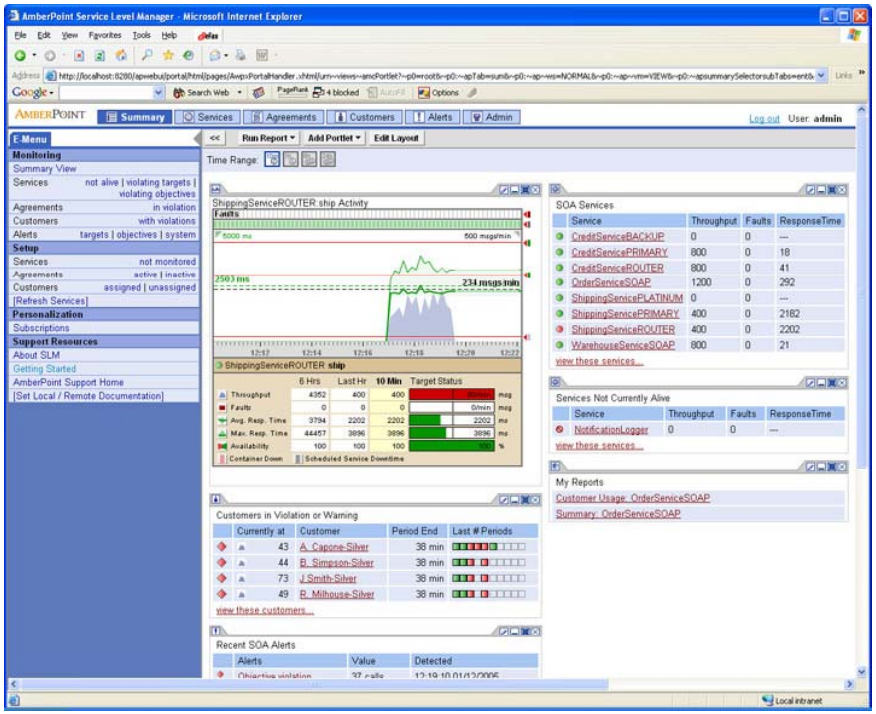



Beispiel:
Integration von Centrasite und LISA

# Werkzeuge [4]

## Management- und Monitoring-Tools

- AmberPoint, Actional, …
- Überwachung der Services im Betrieb (Policys, …)
- Monitoring-Ergebnisse müssen an das Repository zurückgeliefert werden oder über das Repository verlinkt sein



# Werkzeuge [5]

## Integration von Repository und Tools

- Repositorys liefern Integrationstools für gängige Programmierumgebungen wie z.B. Eclipse gleich mit
- Für Servicedefinitionen und Daten aller Art gibt es einen standardmäßigen XML-Datenaustausch
- Problem bei Daten und Informationen, die nicht über WS-Standards definiert werden!

  → Erweiterbarkeit des Repositorys über eigene XML-Schemata und Plugin-Technologien (proprietäre Herstellererweiterungen)

# Zusammenfassung

- Zentrale Punkte eines SOA-Software-Lifecycles sind:
  - flexible **Orchestrierung**
  - **Wiederverwendung von Services**
  - **kontrollierbarer und effizienter Betrieb**
- Effizientes Life Cycle Management muss diese Anforderungen in geeigneter Weise unterstützen
- Daraus ergeben sich hohe Anforderungen an die **Gesamtsicht** über alle verfügbaren Services und Prozesse
- Mit Hilfe von **Service Repositorys** werden alle technischen und organisatorischen Informationen zentral verwaltet
- **Alle Werkzeuge für Planung, Entwicklung und Betrieb müssen an dieses Repository angebunden sein**!