Technische Hochschule Ingolstadt

Fakultät für Elektrotechnik und Informatik

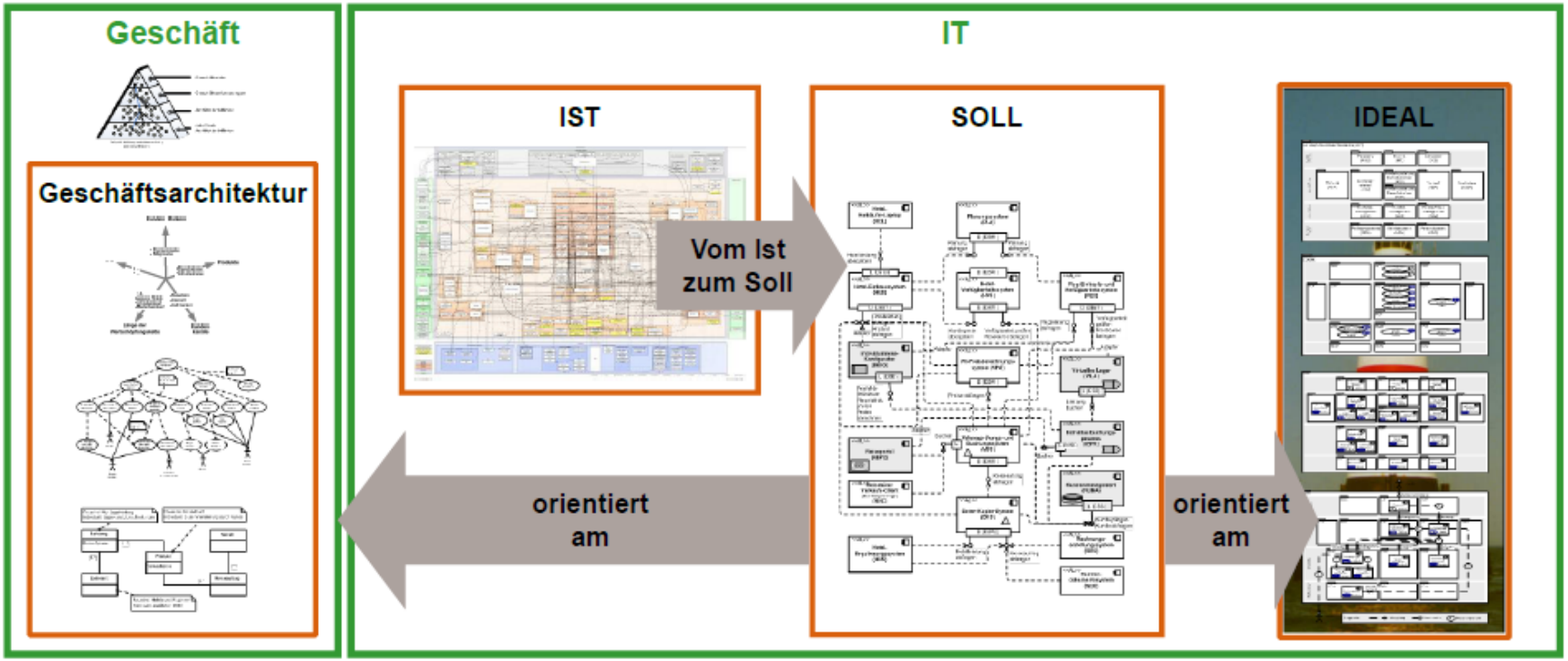
Zukunft in Bewegung

Umsetzung einer Anwendungslandschaft

Integrations- und Migrationstechnologien

Prof. Dr. Bernd Hafenrichter 01.10.2014

Anwendungslandschaften



Schritt 1:
Das Geschäft verstehen

Schritt 3:
Das Ist erheben undbewerten

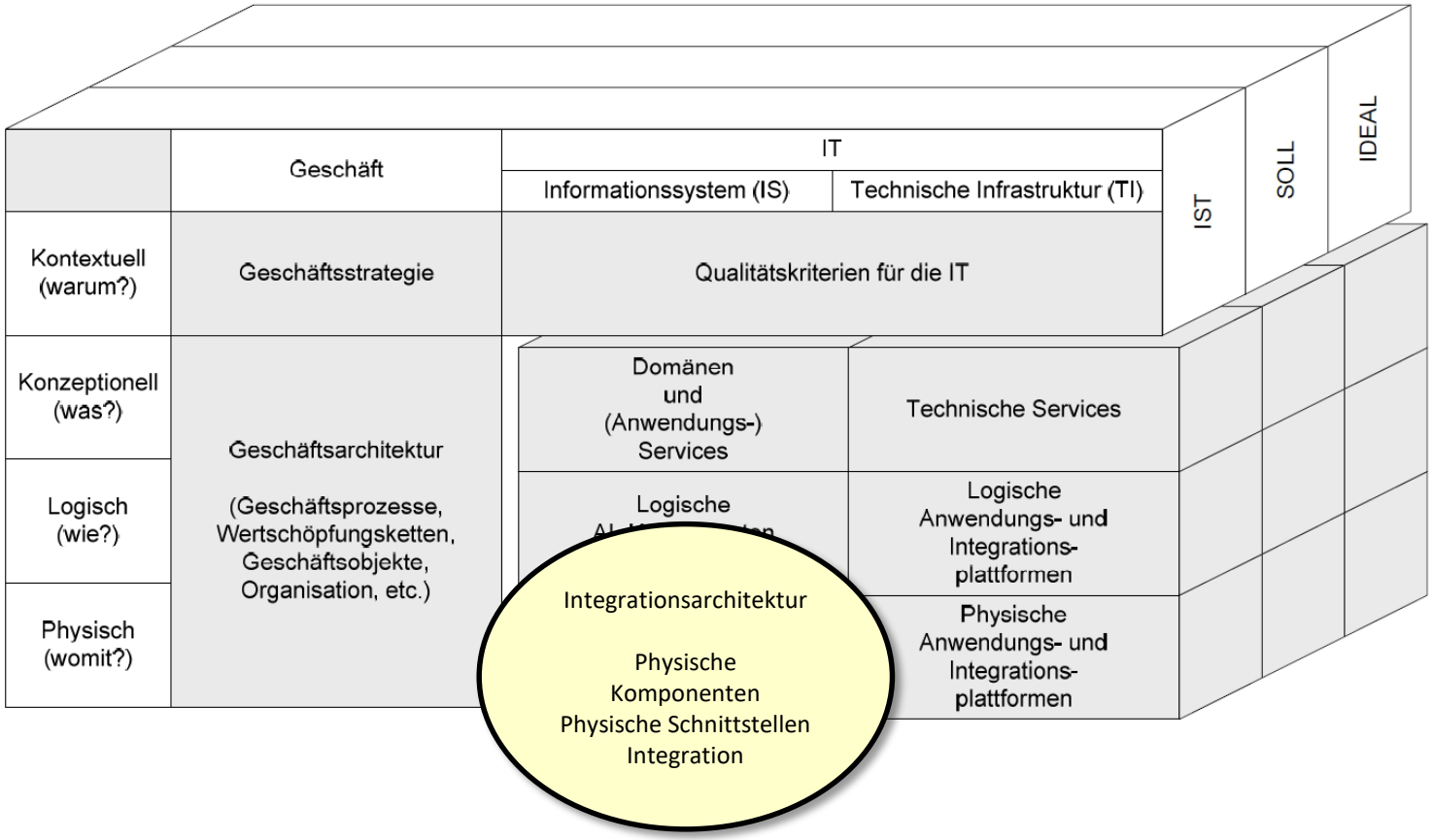
Schritt 4:
Die Soll-Achitektur erstellen

Schritt 2:
Das Idealerstellen

Motivation

- Die Betrachtung von logischen Anwendungslandschaften ist ein wertvolles architektonisches Hilfsmittel
- Sie dient der Orientierung und als Richtschnur bei der Umsetzung
- Für den Aufbau der IST-Landschaft ist es notwendig die Prinzipien der Serviceorientierung auf der physischen Ebene zu konkretisieren

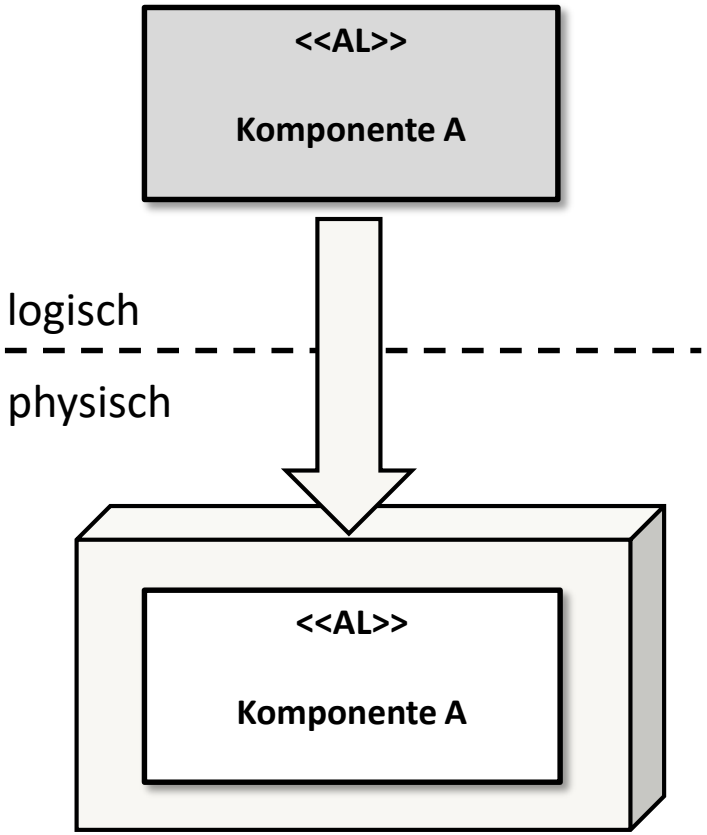
Motivation


Geschäft
IT
Informationssystem (IS)
Technische Infrastruktur (TI)
IST
SOLL
IDEAL
Kontextuell (warum?)
Geschäftsstrategie
Qualitätskriterien für die IT
Konzeptionell (was?)
Logisch (wie?)
Physisch (womit?)
Geschäftsarchitektur
(Geschäftsprozesse, Wertschöpfungsketten, Geschäftsobjekte, Organisation, etc.)
Domänen und (Anwendungs-) Services
Technische Services
Logische
Logische Anwendungs- und Integrations-plattformen
Physische Anwendungs- und Integrations-plattformen
Integrationsarchitektur
Physische Komponenten
Physische Schnittstellen
Integration

Physische Komponenten

Physische Komponenten

- Eine physische Komponente implementiert eine logische AL-Komponente
- Die Implementierung kann konventionell oder durch Orchestration erfolgen
- Sie implementiert die logischen Schnittstellen auf Basis von technischen Schnittstellen
- Die verwendete Technik muss die vordefinierte Kopplungsstufe erfüllen



Physische Komponenten

Physische Komponenten – Konventionelle Implementierung

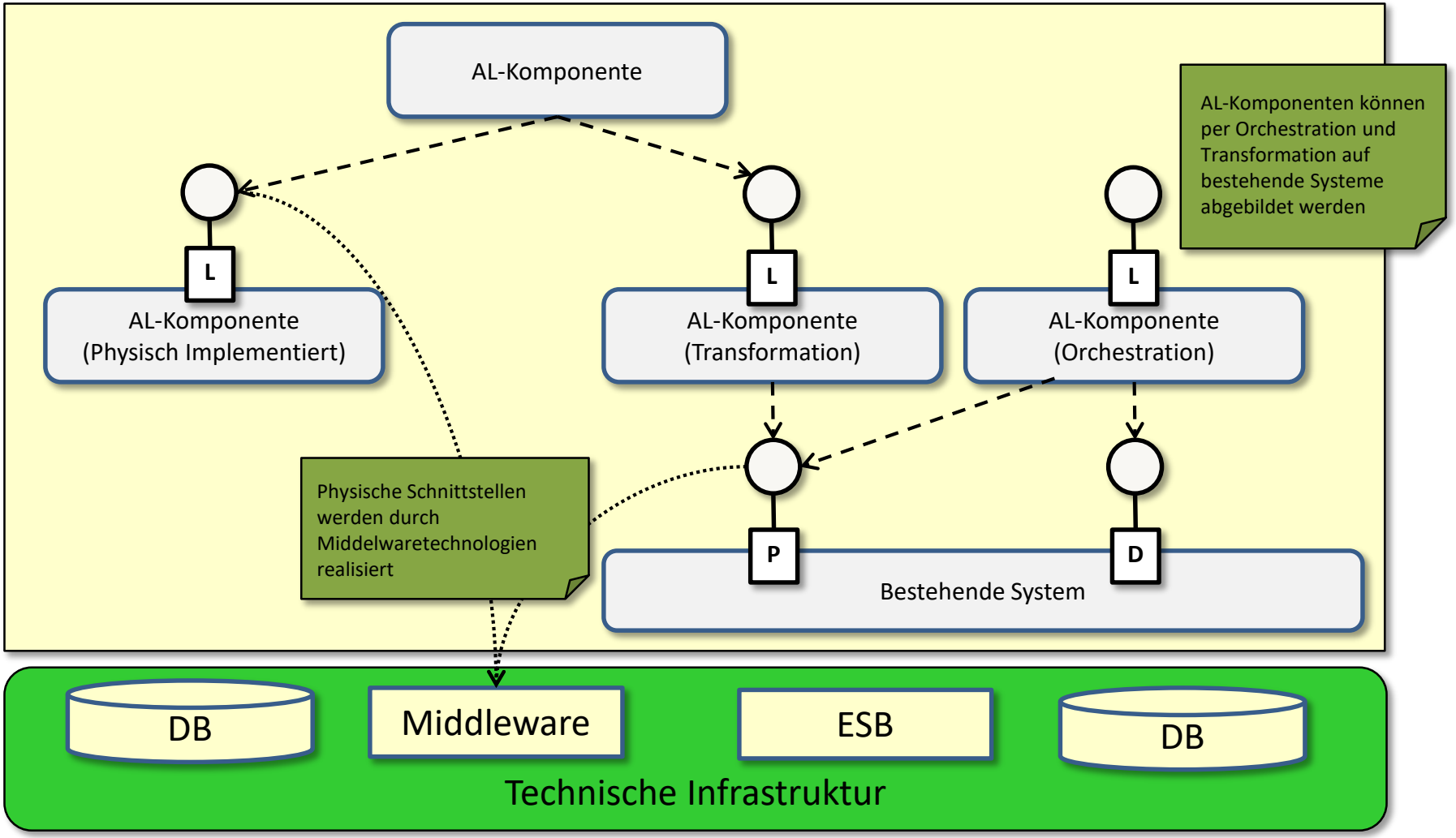
- Implementierung erfolgt durch Neuentwicklung, Anpassung existierender Anwendungen oder Kauf von COTS-Produkten
- Die Anwendung wird auf Basis einer Anwendungsplattform realisiert (Betriebssystem, Container/techn. Stack, Programmiersprache)

Physische Komponenten

Physische Komponenten – Orchestrierung/Integration

- Implementierung von gegebenen Anwendungsservices mit Hilfe einer übergreifenden Ablaufsteuerung
- Diese Ablaufsteuerung realisiert dem Mehrwert des implementierten Komponente
- Die Implementierung erfolgt mit Hilfe von technischen Services die von einer Integrationsplattform zur Verfügung gestellt wird

Technische Infrastruktur einer SOA



Integrationsmuster

- In Anwendungslandschaften sind oft ähnliche Aufgabenstellungen bei der Integration zu lösen
- Mit Hilfe von Mustern sollen allgemeine Lösungsansätze definiert werden, welche auf konkrete Situationen Anwendung finden
- Nachfolgend werden zwei Muster näher erläutert.
- Weitere Muster zur Integration finden sich z.B. in Hohpe G. und Woolf, B.: Enterprise Integration Patterns, Designing and Deploying Messaging Solutions

Integrationsmuster - Orchestration

Aufgabe:

- Die Architektur sieht eine neue Prozesskomponente vor.
- Ein logischer Geschäftsprozess definiert die Logik in welcher Reihenfolge die Schnittstellen/Services aufzurufen sind
- Die benötigten Services stehen in der AL bereits zur Verfügung
- Der realisierte Prozess muss Teilergebnisse propagieren und Fehlersituationen behandeln

Integrationsmuster - Orchestration

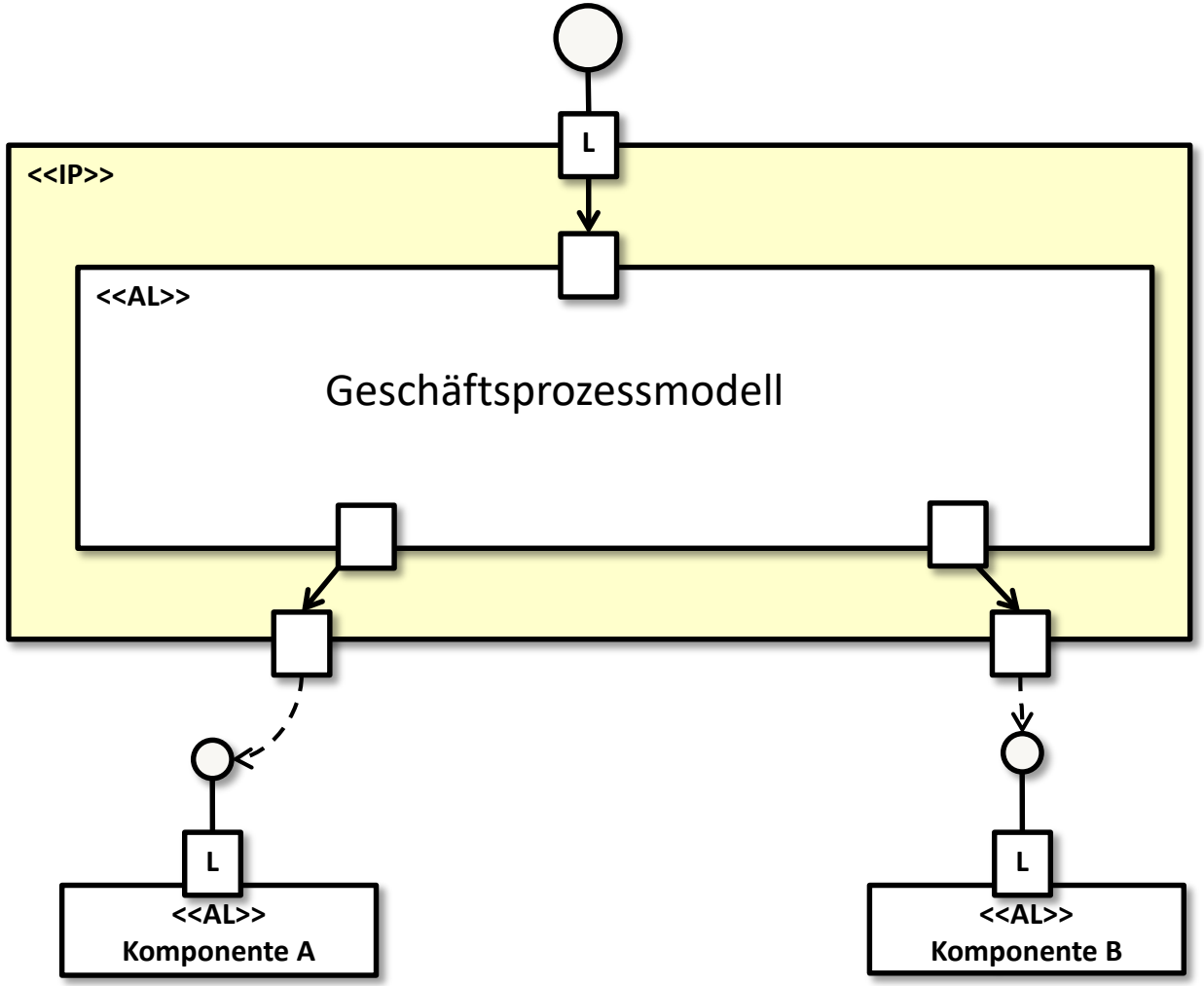
Lösung:

- Der Geschäftsprozess wird als ausführbares Geschäftsprozessmodell formuliert
- Dieses Modell wird durch die Process Engine der Integrationsplattform ausgeführt
- Das Process Modell wird an die Schnittstellen der importierten Services gebunden

Ergebnis

- Eine leichtgewichtige Implementierung des Geschäftsprozesses

Integrationsmuster - Orchestration



Integrationsmuster - Orchestration

Ziele:

- Effektivität der IT-Prozesse
- Kosteneffizienz im Entwicklungsprozess des automatisierten Prozessmodells
- Höhere Wartbarkeit und Agilität durch Wiederverwendbare Funktionen der Integrationsplattform

Negativanzeigen

- Sprachen für Prozessmodellierung sind nur für Prozesse mit beschränkter Komplexität geeignet
- Fehlerbehandlung kann komplexer als der eigentliche Prozess werden

Integrationsmuster – MDM-Hub (Koexistenz)

Aufgabe:

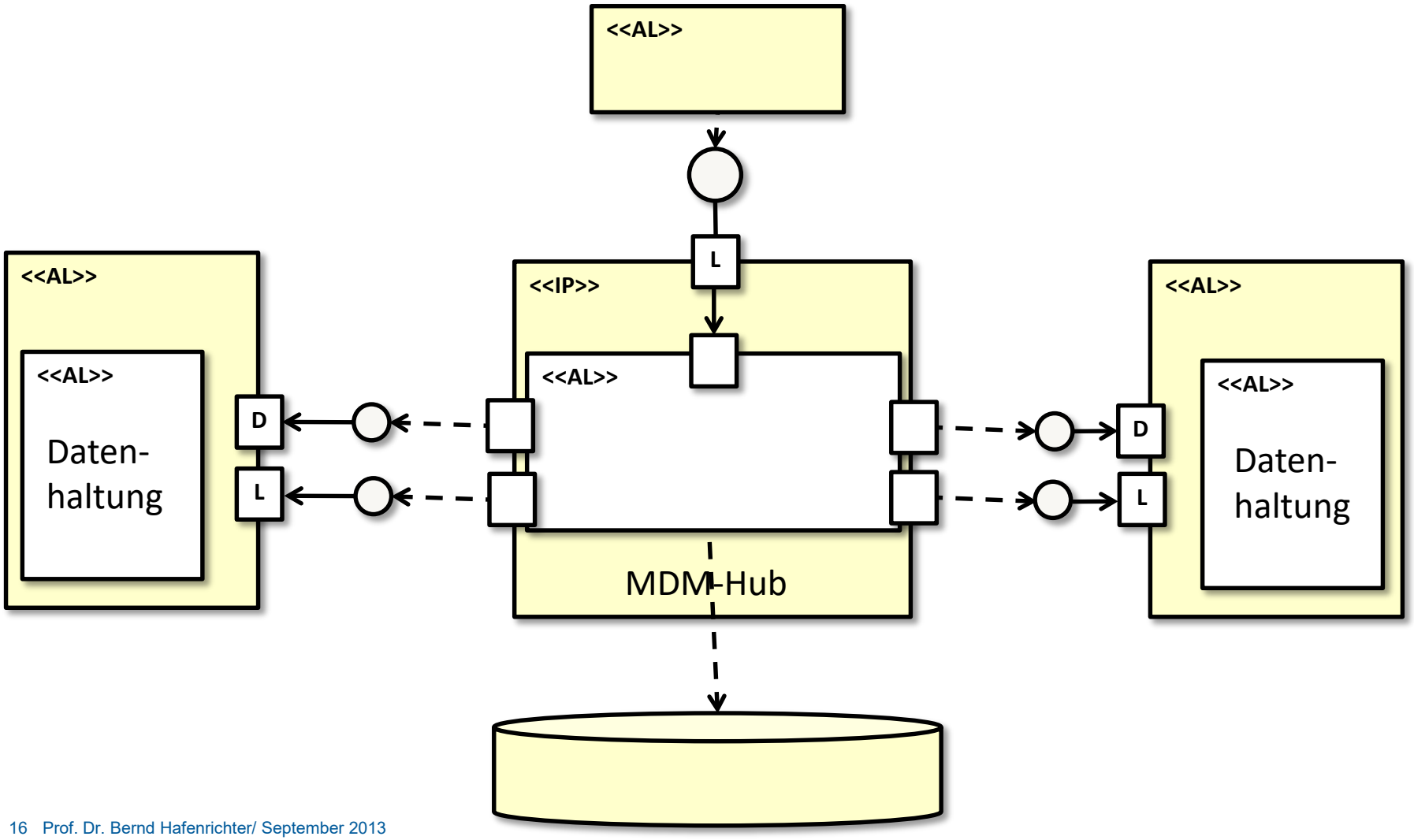
- Weit entfernte AL-Komponenten sollen auf einen einheitlichen und konsistenten Bestand von Stammdaten zugreifen
- Der Zugriff muss performant erfolgen
- Die Stammdatenbestände sind redundant und haben unterschiedliche Datenstrukturen und –qualität
- Eine zentrale Bestandskomponente ist nicht vorhanden

Integrationsmuster – MDM-Hub (Koexistenz)

Lösung:

- Aufbau eines Master Data Management (MDM)-Hubs nach dem Prinzip „Koexistenz“
- Grundidee: Einzelne AL-Komponenten werden berechtigt die Daten außerhalb des MDM-Hubs vorzuhalten und auch zu ändern
- Die Änderungen werden an den MDM-Hub weitergegebenen, konsolidiert und in einer zentralen Master Data Base persistent abgelegt (= „Konsistente Sicht im nachhinein“)
- Die Änderungen werden an die angeschlossenen AL-Komponenten verteilt

Integrationsmuster – MDM-Hub (Koexistenz)



Integrationsmuster – MDM-Hub (Koexistenz)

Ziele:

- Korrektheit/Verfügbarkeit der AL-Komponenten. Durch den asynchronen Abgleich der Bestandsdaten sind die AL-Komponenten unabhängig vom MDM-Hub
- Kosteneffizienz: Die lokalen Anwendungen können einfach auf die Stammdaten zugreifen. Keine Aufwendige Integration notwendig
- Korrektheit der Daten: Domänenspezifische Services zur Qualitätssicherung kombiniert mit manuellen Bereinigungsprozessen

Negativanzeigen

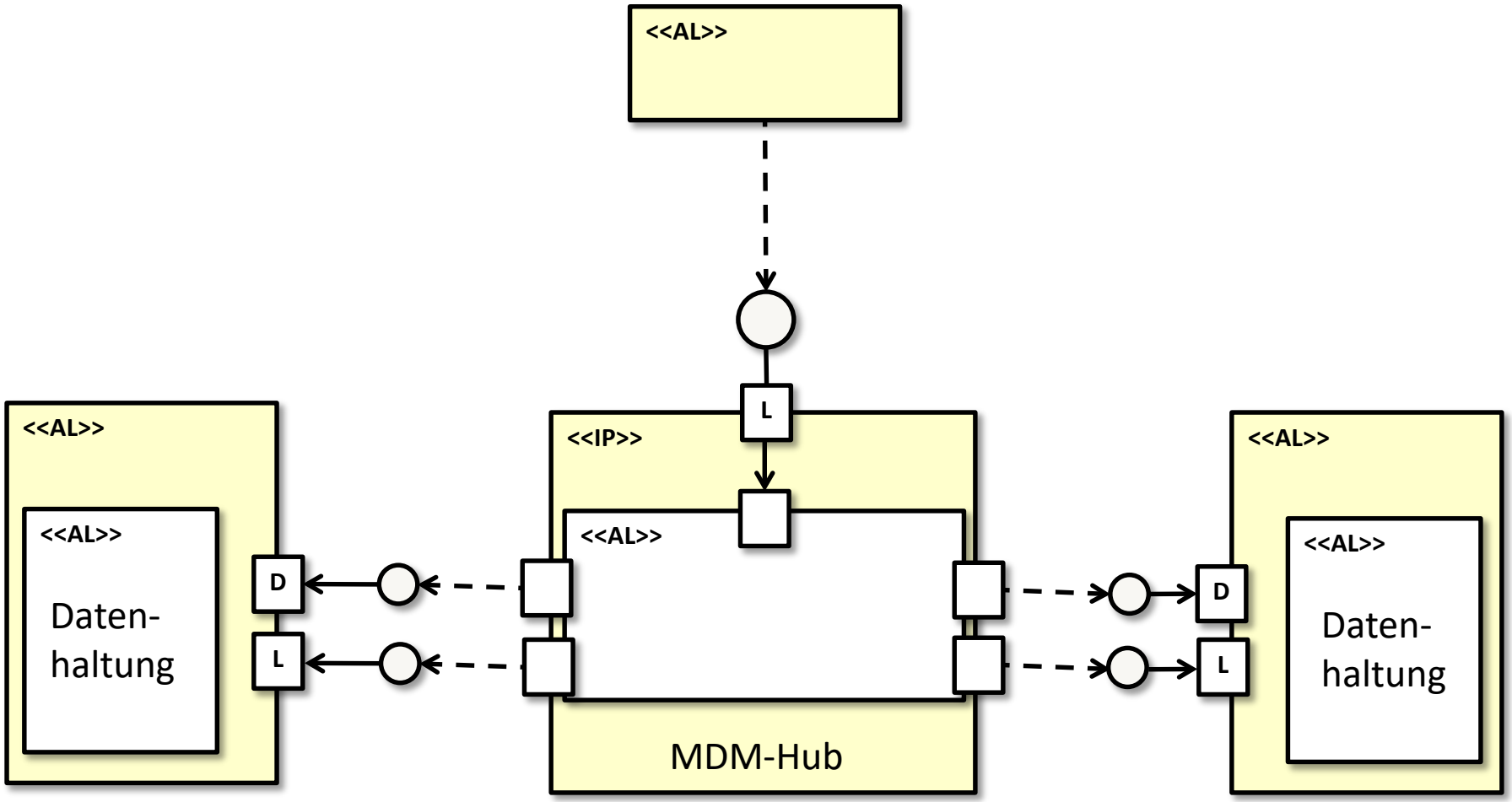
- Verzögerungen bei der Verteilung von Stammdatenänderungen. Problematisch wenn hohe Anforderungen an die Aktualität bestehen
- Fehlende gegenseitige Sperren führen zu zeitweise Inkonsistenten Daten

Integrationsmuster – MDM-Hub (Registrierung/Virtuell)

Eine Weiterer Ansatz:

- Aufbau eines Master Data Management (MDM)-Hubs nach dem Prinzip „Registrierung/Virtuell“
- MDM-Hub enthält keinen eigene Kopie des Datenbestandes
- MDM-Hub verwaltet nur notwendige Metainfromationen (z.B. Mappingtabellen zur Umsetzung der Fachschlüssel zwischen verschiedenen Applikationen und Domänen)
- MDM-Hub greift bei Anfragen online auf die AL-Komponenten zu und transformiert die Ergebnisse (falls notwendig)
- Stellt einheitliche Sicht auf Daten für andere AL-Komponenten zur Verfügung
- Wichtig: Die angebunden AL-Komponenten dürfen nur einen disjunkten Teil der Daten verwalten (z.B. Privat- und Firmenkunden)

Integrationsmuster – MDM-Hub (Registrierung/Virtuell)



Integrationsmuster – MDM-Hub

Ziele:

- Korrektheit/Verfügbarkeit der AL-Komponenten. Durch den asynchronen Abgleich der Bestandsdaten sind die AL-Komponenten unabhängig vom MDM-Hub
- Kosteneffizienz: Die lokalen Anwendungen können einfach auf die Stammdaten zugreifen. Keine Aufwendige Integration notwendig

Negativanzeigen

- Performance für andere Komponenten beim Zugriff auf MDM-Hub

Integrationsmuster – Batch

Aufgabe:

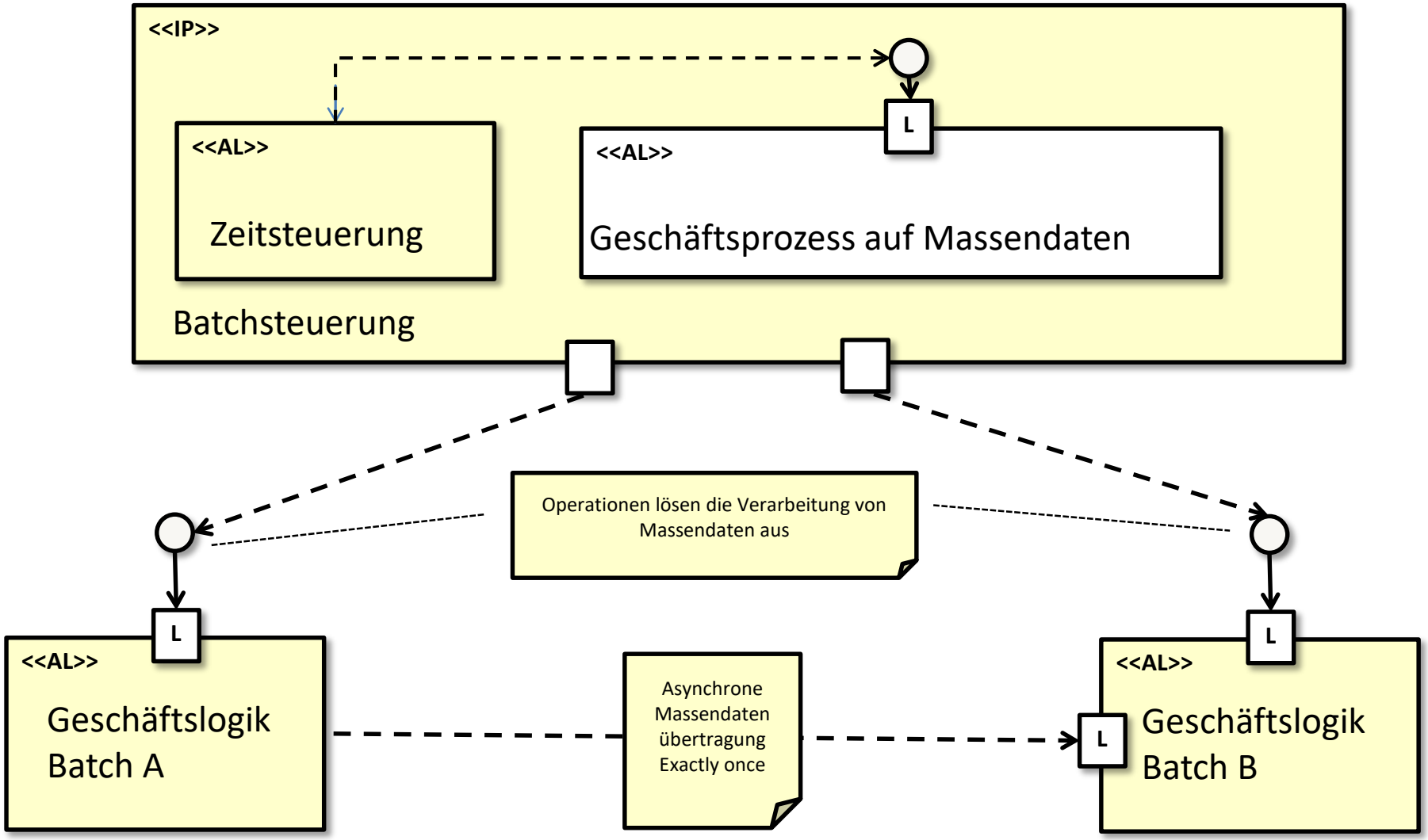
- Verarbeitung von Massendaten in den Anwendungskomponenten
- Die Bearbeitung ist unabhängig von aktuellen Benutzereingaben
- Mehrere AL-Komponenten sind involviert
- Hohe Anforderungen an den Durchsatz

Integrationsmuster – Batch

Lösung:

- Die Geschäftslogik wird durch Batch-Läufe implementiert
- Das Starten der Batchläufe erfolgt Ereignis gesteuert durch eine Batchverarbeitung
- Die Übertragung der Daten erfolgt asynchron z.B. über Dateien

Integrationsmuster – Batch



Integrationsmuster – Batch

Lösung:

- Abhängigkeit von der Verfügbarkeit: Batches sollen laufen können, auch wenn die AL-Komponenten, von denen er Daten bezieht temporär nicht verfügbar ist
- Vertrauen: Batches sollen nie von der Korrektheit der gelieferten Daten ausgehen
- Wissen: Batches sollen minimales Wissen über die Implementierung der Datenliefernden Komponenten machen. (Schlüsseln und DB-Interne gehören nicht in eine Exportdatei)

Integrationsmuster – Batch

Ziele:

- Kosteneffizient des Betriebs: Massendatenverarbeitung kann performanter implementiert werden als Einzelverarbeitung.

Negativanzeigen

- Prozesse die Benutzerinteraktion benötigen oder sofort ausgeführt werden müssen sind ungeeignet für Batchverarbeitung
- Batchprozesse können schwer auf online-Verarbeitung umgestellt werden. (d.h. auch zukünftige Anforderungen berücksichtigen)