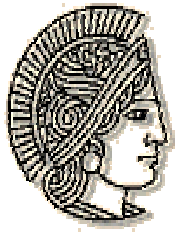
Organisationsmodelle zur komponentenorientierten Anwendungsentwicklung / terminologiebasierte Spezifikation von Fachkomponenten

Stephan Sahm



TU Darmstadt

APCON

Agenda

1. Hintergrund
2. Grundlagen
3. Organisationsmodelle zur komponentenorientierten Anwendungsentwicklung
4. Terminologiebasierte Spezifikation von Fachkomponenten
5. Ausblick

1. Hintergrund

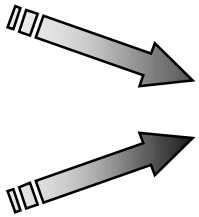
Grundlage: Studienarbeit „Entwicklung von Organisations- und Vorgehens-modellen zur komponentenorientierten Anwendungsentwicklung“

TU Darmstadt

Fachgebiet Wirtschaftsinformatik I -
Entwicklung von Anwendungssystemen

Prof. Dr. Erich Ortner

Eine große dt. Fluggesellschaft



2.1 Was ist eine Komponente (1)

- **Komponente:**
  - Menge **wiederverwendbarer, abgeschlossener, vermarktbarer** (Software-) **Artefakte**
  - stellt Dienste über eine **wohldefinierte Schnittstelle** zur Verfügung
  - ist in zur Zeit ihrer Entwicklung **unvorhersehbarer Kombination** mit anderen Komponenten einsetzbar
  - Artefakte hier u.a. :
    - **ausführbarer** Code,
    - syntaktische, semantische und nicht-funktionale **Spezifikation**,
    - automatisierte **Tests**
    - **Dokumentation**

2.1 Was ist eine Komponente (2)

- **Unabhängigkeit**: Komponenten und ihre Dienste sind nicht an bestimmte Anwendungen gekoppelt. ⇨ **Standardisierung**
- **Abgeschlossenheit**: Die Ablauffähigkeit einer Komponente ist nicht vom Vorhandensein einer anderen Komponente abhängig. ⇨ **Modellierung**
- **Offenheit**: Eine Komponente kann grundsätzlich mit anderen Komponenten in beliebigen Systemumgebungen zusammenwirken. ⇨ **Technologie**

- Fachkomponente:
  - **Komponente**, die eine bestimmte Menge von **Diensten einer betrieblichen Anwendungsdomäne** implementiert und an unternehmensspezifische Anforderungen angepasst werden kann.

2.2 Probleme bei der Wiederverwendung (1)

- OO hat Hoffnungen bzgl. Wiederverwendung kaum erfüllt
  - Trennung zwischen **Modellierungs-** und **Implementierungssichtweise**
  - Vererbung und das „**Problem der änderungsempfindlichen Basisklassen**“
  - häufig sehr feine Granularität ➔ instanziierte **Objekte** unabhängig, **referenzieren einander** aber **auf komplexe Weise**

„We don`t have spaghetti code in object systems, we have little balls of rigatoni that stick together producing mush.“ [Suth97]

  - **Zuordnung von Methoden zu Objekten** aus fachlicher Sicht teilweise problematisch z.B. *Buch.ausleihen( )*

2.2 Probleme bei der Wiederverwendung (2)

itelligence powered by APCON & SVC

- Organisatorische Probleme
  - Kostenfrage bei Entwicklung von „Komponenten“ in Anwendungs-entwicklungsprojekten ungeklärt
  - Zuständigkeiten bei Entwicklung und Wartung

- Unternehmenskulturelle Probleme
  - Projektleiter haben kein Interesse, „Komponenten“ zu entwickeln
  - „not-invented-here“-Syndrom

- Fachliche Probleme
  - „Komponenten“aus Anwendungsentwicklungsprojekten zu speziell für Wiederverwendung

⇨**These 1:**

Trennung von Komponentenproduktion und -konsumtion notwendig

2.2 Probleme bei der Wiederverwendung (3)

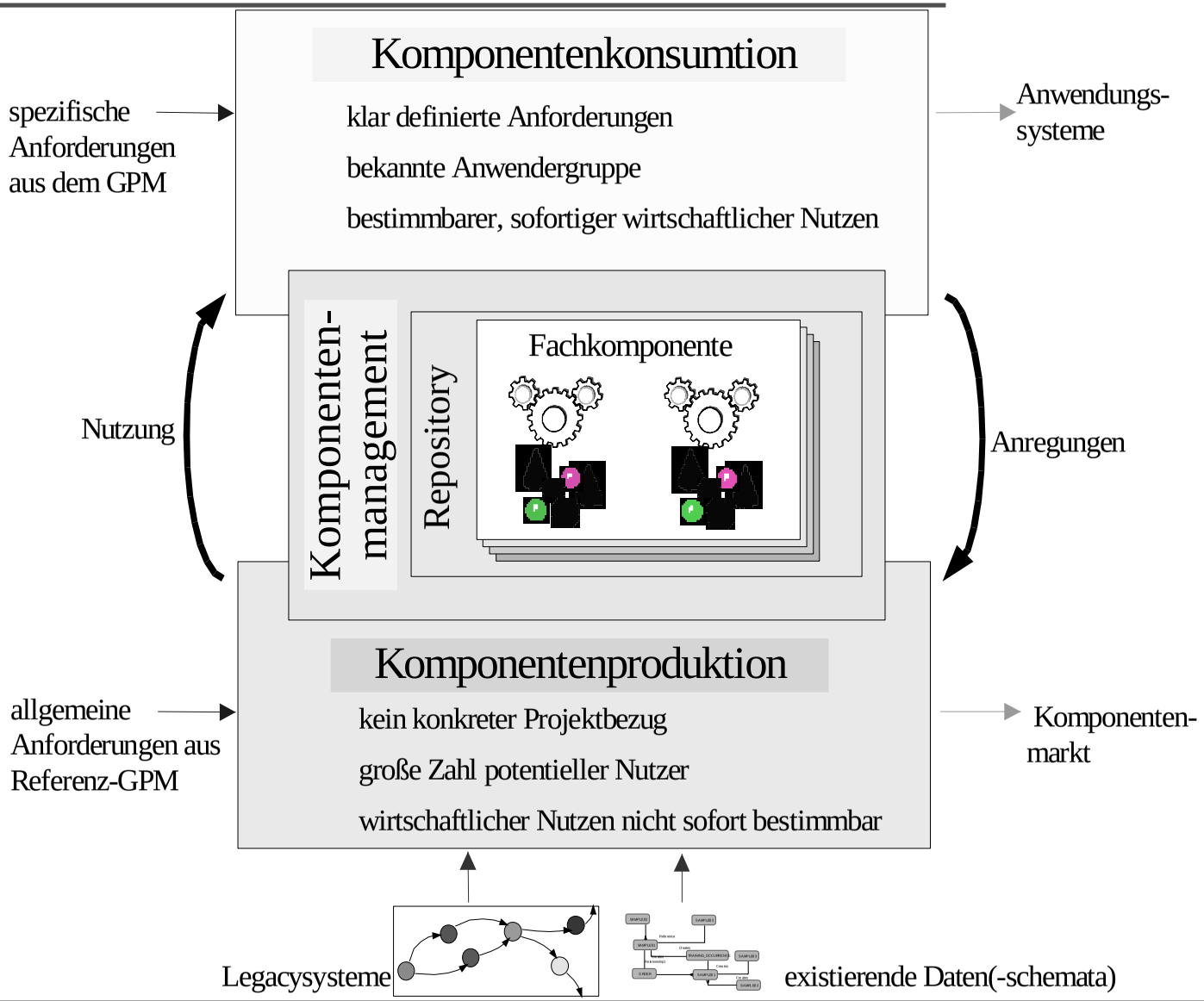
itelligence powered by APCON & SVC

- Beschreibungs-/Spezifikationsproblem
  - normalsprachlich: unpräzise und interpretationsfähig
  - formalsprachlich: aufwendig, unhandlich und praxisfern

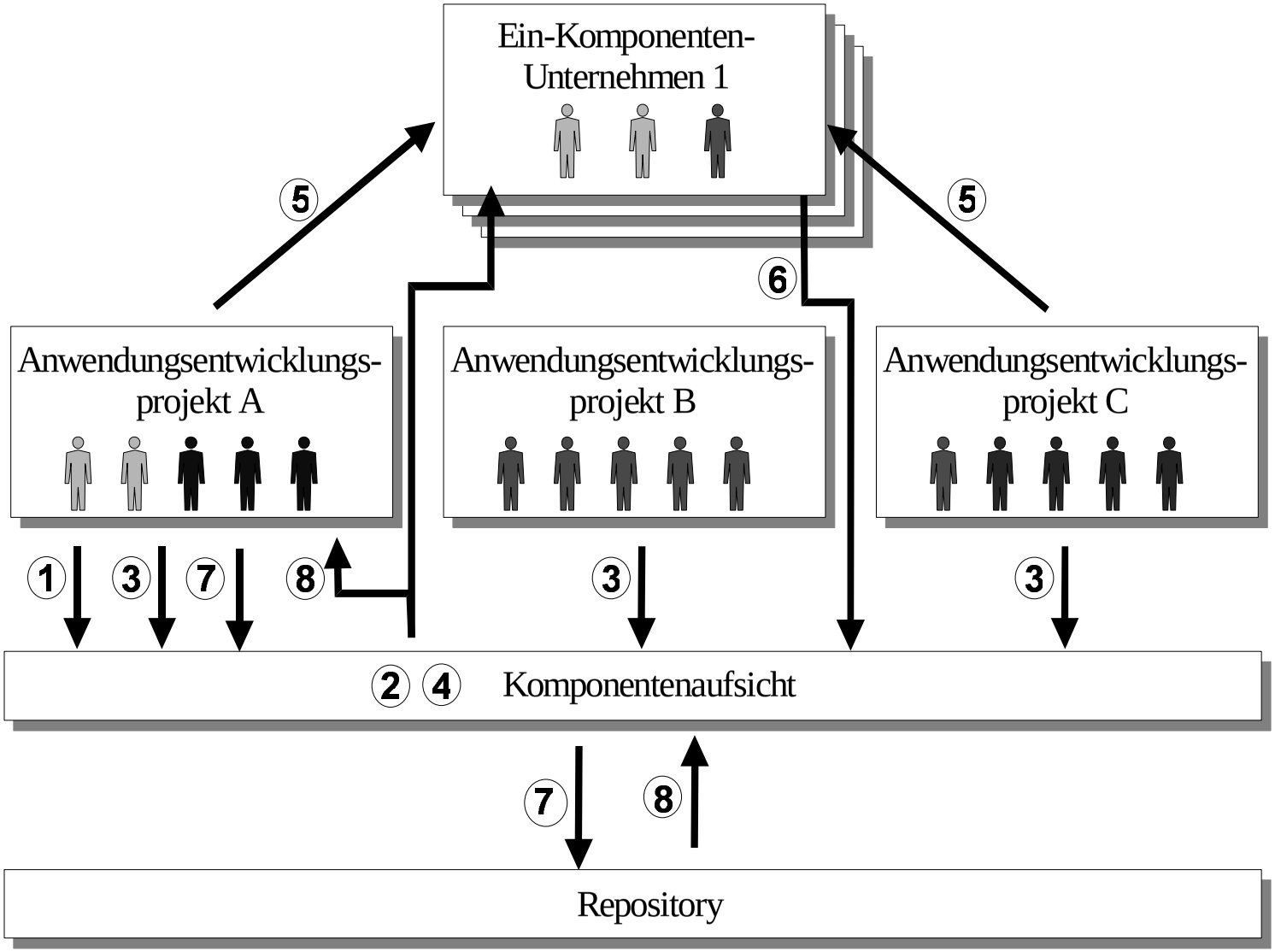
⇨**These 2:**

Materiale Sprachen zur Spezifikation notwendig

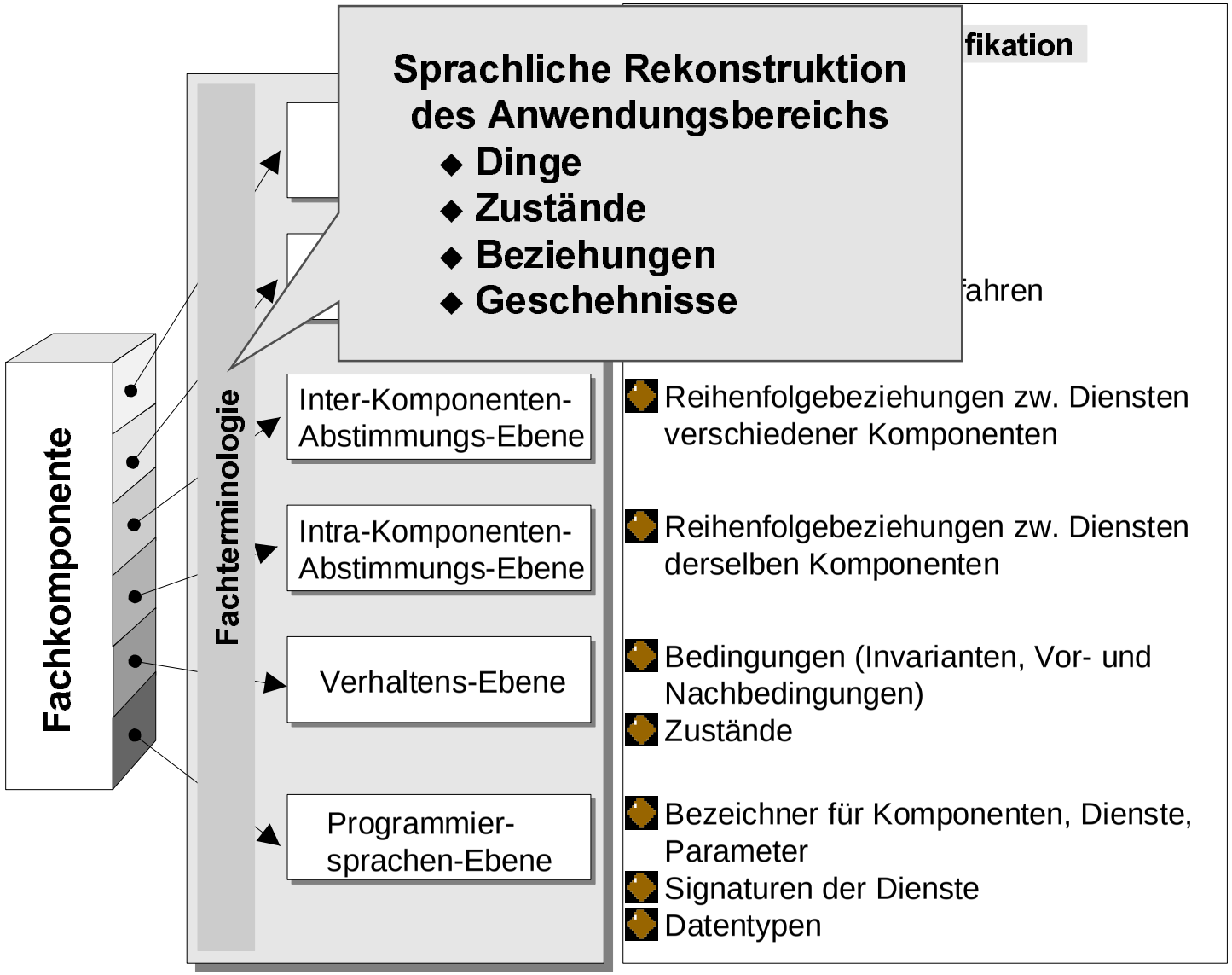
3. Aufbauorganisation (1)



3. Aufbauorganisation (2) - ein flexibles Organisationsmodell



4.1 Spezifikationsebenen



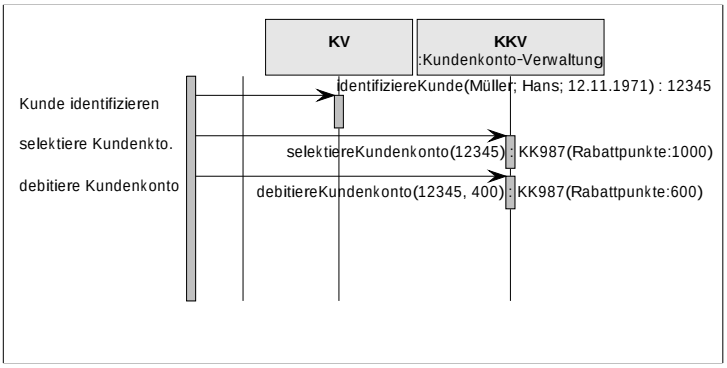
4.2 Beispiel einer Spezifikation

- Folien

Ebene	**Notation / Sprache**	**Verträge, Spezifikationen**
Komponenten-Begriffs-Ebene	universelle Normsprache	• *Meilen* sind *Rabattpunkte.* • *Meilenkonto* ist identisch mit *Kundenkonto.* • *Rabattpunkte* sind Teil eines *Kundenkontos.* • *Kundenkonto debitieren* wird ausgeführt, während *Prämie buchen* ausgeführt wird • ...
Leistungs-Ebene	Fachnormsprache für Service-Level	Normallast: 500 Aufrufe/Min. Maximallast: 10.000 Aufrufe/Min. Antwortzeit: unter Normallast 1 Sekunde unter Maximallast 2 Sekunden Verfügbarkeit: >= 99,999 % Durchsatz: >= 10.000 Aufrufe/Min. Ressourcenbedarf: unter Normallast 5 MB RAM unter Maximallast 100 MB RAM
Inter-Komponenten-Abstimmungs-Ebene	Fachnormsprache für Abstimmungs-Ebene	Es existiert ein Kunde, für den gilt, dass Kundennummer gleich Kundennummer in Kundenkonto KK.
	erweiterte OCL [Conr00]	`context KundenKonto_Verwaltung:: KundenKonto_debitieren(KK:string, zdR:long): void pre: Verwaltung_Kunde.exists(Kundennr = KK.Kundennr) sometime_past(Verwaltung_Kunde.Insert(Kunde. Kundennr = KK.Kundennr)) post: -- none`
Intra-Komponenten-Abstimmungs-Ebene	Fachnormsprache für Abstimmungs-Ebene	Irgendwann in der Vergangenheit fand statt, Einfügen Kundenkonto KK. Irgendwann in der Vergangenheit fand statt, Kreditieren Kundenkonto KK Im Anfangszustand gilt, Rabattpunkt von Kundenkonto KK ist gleich 0.
	erweiterte OCL [Conr00]	`context KundenKonto_Verwaltung:: KundenKonto_debitieren(KK:string, zdR:long): void pre: sometime_past(Verwaltung_Kundenkonto.Insert(KK)) sometime_past(KundenKonto_Verwaltung. kreditieren(KK, Integer)) post: -- none ---------- context KundenKonto initially: self.Rabattpunkte = 0`

Ebene	**Notation / Sprache**	**Verträge, Spezifikationen**
Verhaltens-Ebene	„lokale“ Fachnorm-sprache	*Meilenkonto* (*KK)* ist Parameter von *Kundenkonto debitieren*. *zu_debitierende_Rabattpunkte (zdR)*ist Parameter von *Kundenkonto debitieren*. Falls *Rabattpunkte* größer oder gleich *zu_debitierende_Rabattpunkte, Rabattpunkte* in *Kundenkonto* verringert um *zu_debitierende_Rabattpunkte* wird zurückgeben, sonst Exception *Operation nicht zulässig*.

	OCL	`context KundenKonto_Verwaltung::`
		`KundenKonto_debitieren(KK:string, zdR:long): void`
		`pre: KK.Rabattpunkte >= zdR`
		`post: KK.Rabattpunkte = KK.Rabattpunkte@pre - zdR`
		`----------`
		`context KundenKonto inv:`
		`self.Rabattpunkte >= 0`

	UML-Sequenzdiagramme (Auszug)	

Programmiersprachen-Ebene	OMG IDL	`interface KundenKonto_Verwaltung {`
		`void KundenKonto_debitieren (`
		`inout string KundenKonto,`
		`in long zu_debitierende_Rabattpunkte)`
		`raises (Operation_nicht_zulaessig);`
		`};`