wirtschaftsinformatik
managementinformationssysteme

# system architectures

Übung 3

Wintersemester 2009/2010

Arbeitsgruppe Wirtschaftsinformatik

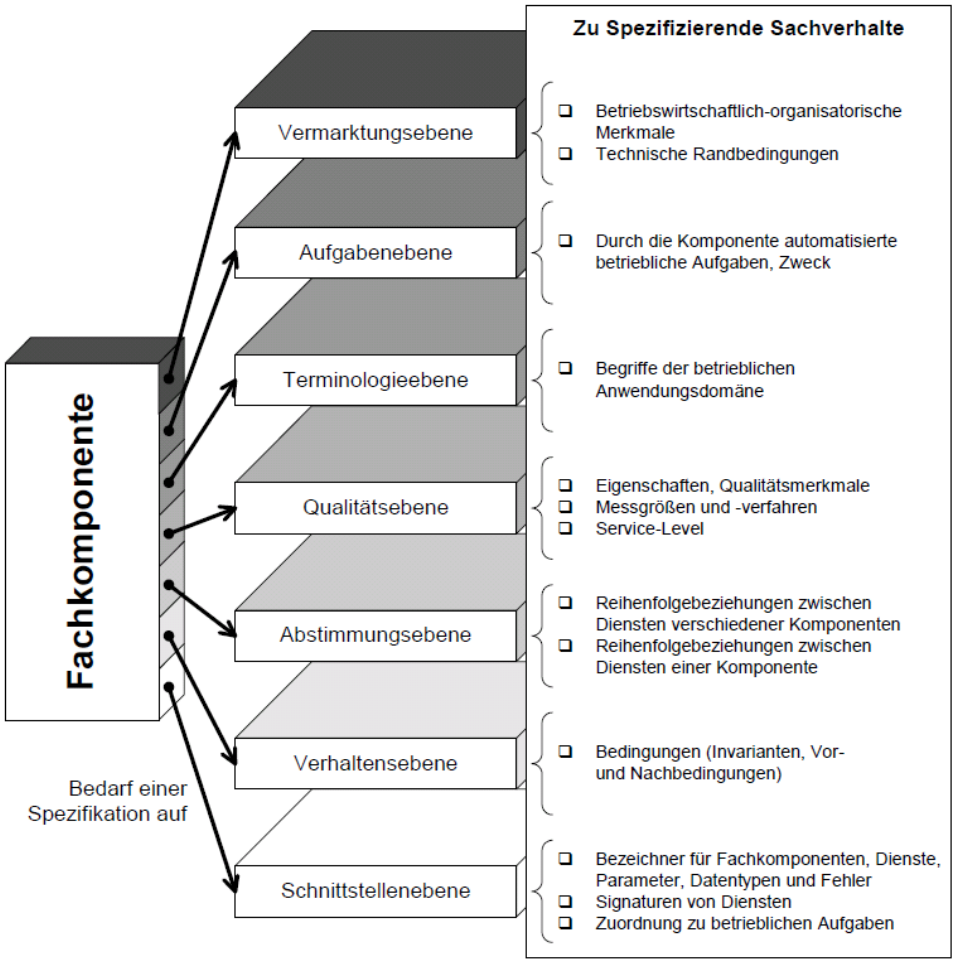
– Managementinformationssysteme –

# rekapitulation: definition fachkomponente

- A component consists of different software artefacts. It is reusable, enclosed and marketable, offers services by well defined interfaces and can be employed in unforeseeable combinations with other components during development time

- A business component (BC) is a component, which offers a certain set of services in a business domain

# spezifikation von fachkomponenten




Quelle:
Memorandum der Fachkomponenten

# schnittstellenebene

- **Zweck**
  - Nennung der angebotenen **Dienste**
  - Benennung öffentlich verfügbarer Attribute, Variablen, Konstanten
  - Vereinbarung spezieller Datentypen
  - Festlegen der **Schnittstellensignaturen**
  - Deklaration von Fehlermeldungen, Ausnahmezuständen
  - Herstellung **technischer Kommunikationsfähigkeit**
  - **Semantische Aspekte im Hintergrund**
  - Verknüpfung von **Attributen mit Begriffen, die auf Terminologieebene** definiert wurden
  - Verknüpfung von **Diensten mit betrieblichen Aufgaben, die auf der** Aufgabenebene definiert wurden

# schnittstellenspezifikation

- **Notation**
  - **Interface Definition Language (IDL)**
    - Von der Object Management Group (OMG) vorgeschlagen
    - Offener Standard
    - In Forschung und Industrie akzeptiert
  - **Tabellenform**
    - Für korrespondierende Begriffe und Datentypen
    - Für korrespondierende Aufgaben und angebotene Dienste

# beispiel (teil I)

```
interface Bestandskontenverwaltung {
  typedef string KontoNr;
  typedef double Bestand;

  struct Zeitpunkt { ... };

  struct Konto {
    KontoNr      n;
    Bestand      Sicherheitsbestand;
    Bestand      Meldebestand;
    ...
  };

  struct Buchungssatz {
    KontoNr      n;
    Zeitpunkt    Termin;
    string       Auftragsnummer;
    double       Menge;
  };
```

# beispiel (teil 2)

```
  exception ZuGeringerBestand {};

  void     Buche(in Buchungssatz b);
  void     Reserviere(in Buchungssatz b) raises (ZuGeringerBestand);
  Bestand  BerechneBestandZum(in KontoNr n, in Zeitpunkt z);
  Bestand  BerechneBedarf(in KontoNr n, in Zeitpunkt z);
  void     SetzeSicherheitsbestand(in KontoNr n, in double Menge);
  void     SetzeMeldebestand(in KontoNr n, in double Menge);
};

interface extern {
  typedef long PeriodeInTagen;

  struct Zeitpunkt {
    unsigned short Tag;
    unsigned short Monat;
    unsigned short Jahr;
  };

  PeriodeInTagen BerechnePeriodeInTagen(in Zeitpunkt b, in Zeitpunkt e);
};
```

Quelle:
Memorandum der Fachkomponenten

# aufbau

- Schlüsselwort ***interface***
  - Festlegung des Namens der Fachkomponente
  - Untergeordnete Teile der Fachkomponente werden dadurch eindeutig identifizierbar

- Typdefinitionen (Schlüsselwort ***typedef)***
- Strukturdefinitionen (Schlüsselwort ***struct)***
- Auf Basis vordefinierter Typen
- Benötigt für die Spezifikation der Signaturen der Dienste

# aufbau (2)

- Definition von Strukturen
  - Name der Struktur
  - Zusammensetzung der Struktur aus vordefinierten Typen

- Definition von Ausnahmezuständen
  - Schlüsselwort ***exception***
  - z.B. aufgrund eines zu geringen Bestandes kann der notwendige Bedarf nicht reserviert werden

# aufbau (3)

- Deklaration der angebotenen Dienste mit
  - Schnittstellensignatur
  - Typ des Rückgabewertes
  - Name des Dienstes zur eindeutigen Identifikation innerhalb der Fachkomponente
  - Angabe der vom Dienst erwarteten Eingangsparameter

- Deklaration von externen Diensten
  - Spezifikation der Dienste, die von der Fachkomponente genutzt werden
  - Notation identisch zu der angebotener Dienste

# verknüpfung attribute mit fachbegriffen

| Fachbegriff | Datentyp |
|---|---|
| Konto | Konto |
| Bestand | Bestand |
| Buchungssatz | Buchungssatz |

| Aufgabe | Dienst |
|---|---|
| Buche Buchungssatz b | void Buche (in Buchungssatz b) |
| Reserviere Buchungssatz b | void Reserviere (in Buchungssatz b) raises (ZuGeringerBestand) |
| Berechne Bestand des Kontos mit der Kontonummer n zum Zeitpunkt z | Bestand BerechneBestandzum (in KontoNr n, in Zeitpunkt z) |

# Übung

- Fahrkartenkauf am Schalter bei der DB AG
  - Zielknoten
  - Umsteigehäufigkeit
  - Abfahrt/Ankunftszeit
  - Klasse (1./2.)
  - Zugtyp
  - Sitzplatzreservierung
  - Sparangebote
  - BahnCard
  - Zahlungsart

# Übung 2

- Interface Fahrkartenkauf_am_Schalter {
  - typedef string Zielbahnhof;
  - typedef double Preis;

  - struct Zugart{

    string Zugtyp;

    int Klasse;

    }

# verhaltensebene

- **Zweck**
  - Nähere Beschreibung des Verhaltens der Fachkomponente
  - Spezifikation von **Invarianten - Bedingungen, die immer erfüllt sein müssen**
    - Meldebestand eines Bestandskontos muss immer größer sein als der Sicherheitsbestand
  - Spezifikation von **Vor- und Nachbedingungen - Bedingungen, die vor bzw.** nach Ausführung von Diensten der Fachkomponente erfüllt sein müssen
  - Auch Spezifikation der Bedingungen, die die Fachkomponente an Dienste stellt, die von Außen nachgefragt werden (externe Dienste)

# notation

- **Object Constraint Language (OCL)**
  - Von der OMG vorgeschlagen
  - Als Ergänzung der IDL zur Spezifikation auf der Verhaltensebene

- **Ergänzung**
  - Alle Bedingungen auch in natürlichsprachlicher Form
  - Verbesserte Verständlichkeit für bestimmte Adressaten
  - Geringere Präzision