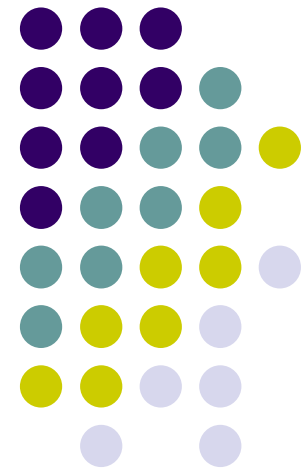


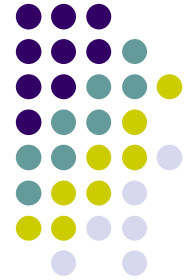
# Zwischenbericht Bakkalaureatsarbeit

---

Beispielkatalog für EJB-  
Entwurfsmuster

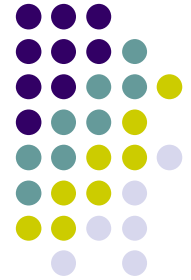
Ronny Klinger





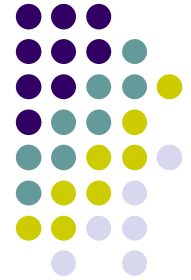
# Aufgabenstellung

- Einarbeitung in existierende EJB-Entwurfsmuster
- Aufarbeitung des Hotelbeispiels aus [Cheesman] unabhängig von konkreter Technologie
- Anpassung und prototypische Realisierung des Beispiels unter Verwendung der EJB 2.0-Technologie, dabei Anwendung der EJB-Entwurfsmuster
- Erstellung eines Kataloges übersichtlicher Java-Beispiele für ausgewählte Muster basierend auf der Implementierung des Hotelbeispiels



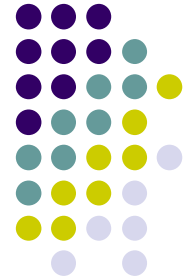
# Aktueller Stand (1)

- Einarbeitung in existierende EJB-Entwurfsmuster
  - Service Locator
  - **Business Delegate**
  - **Session Façade**
  - **Value Object**
  - Value Object Assembler
  - Value List Handler
  - Aggregate Entity
  - Data Access Object
  - **Service Activator**



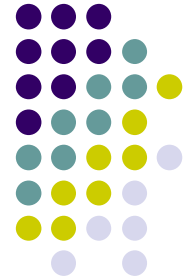
## Aktueller Stand (2)

- Aufarbeitung und prototypische Realisierung des Hotelbeispiels
  - Vorgehen nach dem in [Cheesman] vorgestellten Prozess
  - Spezifikation des Systems unabhängig von einer bestimmten Komponententechnologie und anschließend Anpassung an EJB 2.0
  - Dabei Anwendung verschiedener EJB-Entwurfsmuster



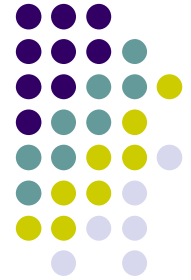
# Business Delegate (1)

- Problem
  - Clients benötigen i.A. Funktionalität von mehreren Enterprise Java Beans
  - Zugriff auf diese einzelnen Beans setzt detaillierte Kenntnisse des serverseitigen Geschäftsmodells voraus
  - Viel Code zum Verwalten und Erzeugen der Beans im Client
  - Es entsteht enge Kopplung zwischen fachlicher Schicht und Präsentationsschicht



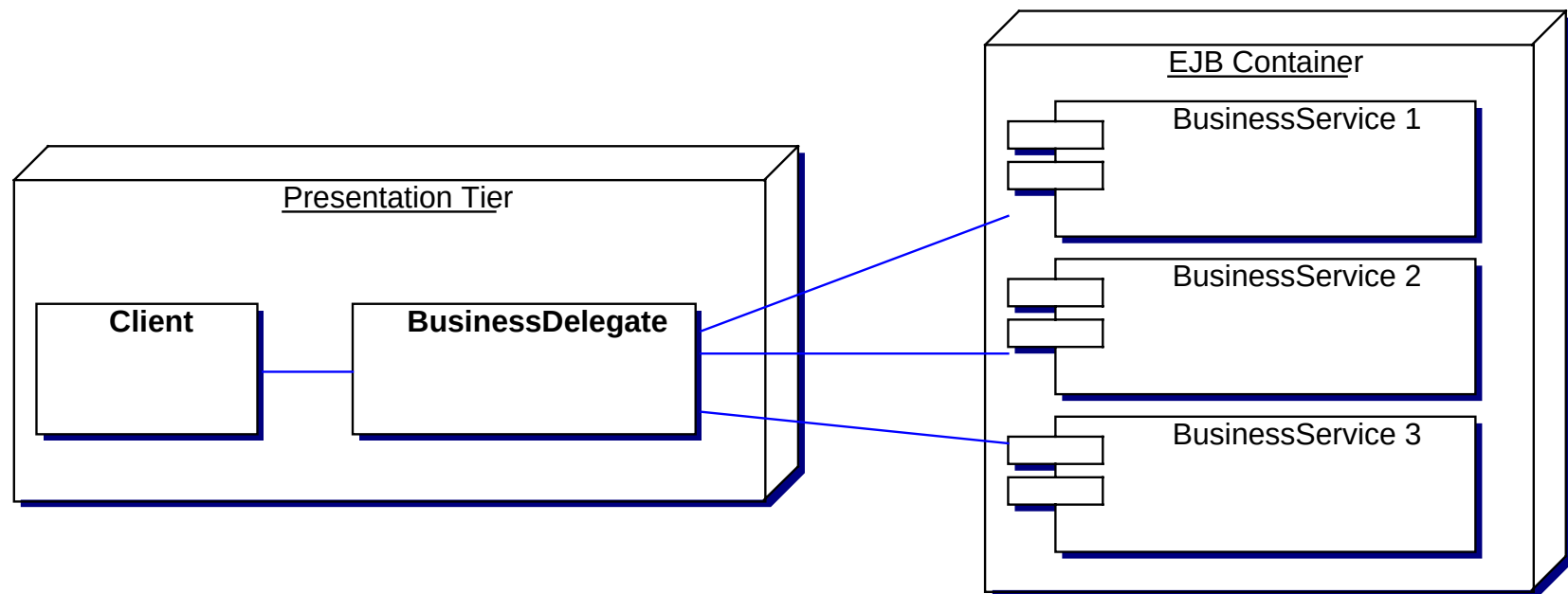
# Business Delegate (2)

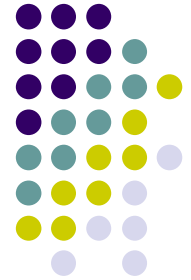
- Lösung
  - Neues Objekt auf Client-Seite als Repräsentant für die Geschäftslogik einführen
  - Clients greifen auf dieses Objekt zu, um Methoden aufzurufen
  - Umsetzung der EJB-Fehlermeldungen in einfachere anwendungsspezifische Fehlermeldungen
- Vorteil
  - Änderungen am serverseitigen fachlichen Modell führen im Client nur zur Änderung des Business Delegates



# Business Delegate (3)

- UML-Darstellung

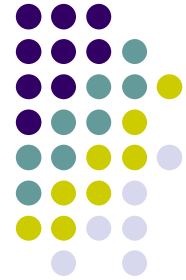




# Session Facade (1)

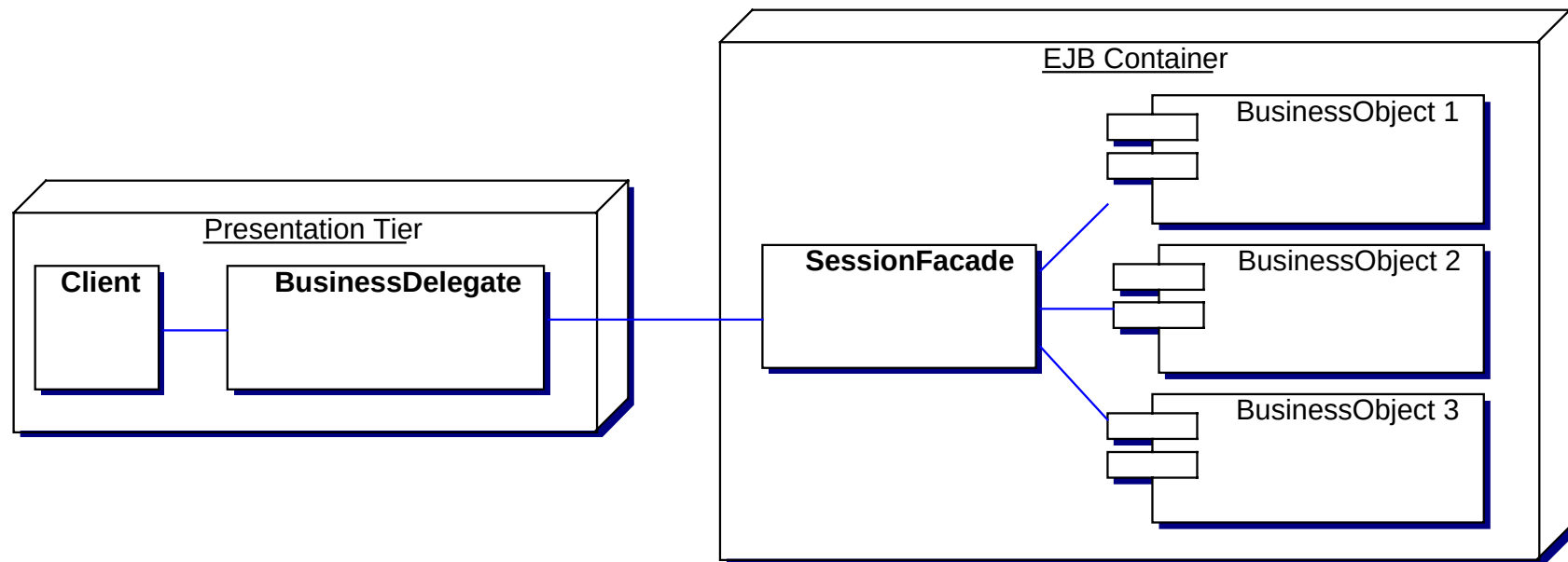
- Problem
  - Geschäftsprozesse sind i. A. auf mehrere EJBs verteilt
  - Client benötigt wie beim Business Delegate Kenntnis des Geschäftsmodells und greift dabei auf die EJBs zu
- Lösung
  - Einheitliche Schnittstelle schon auf Server-Seite
  - Als Session Bean implementiert

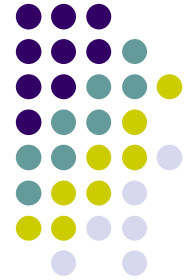




# Session Facade (2)

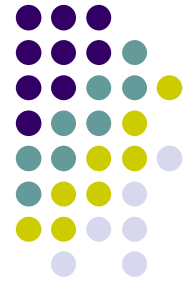
- UML-Darstellung
- Einsatz in Verbindung mit Business Delegate möglich





# Value Object (1)

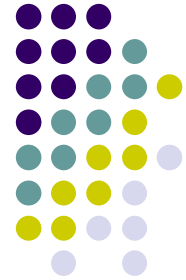
- Problem
  - Viele einzelne Methoden-Aufrufe auf Enterprise Beans erzeugen viel Netzwerklast
  - Bei Entity Beans: set- und get-Methoden für persistente Attribute
  - Bei Session Beans: Aufruf von mehreren Methoden, die eigentlich alle fachlich zusammengehören



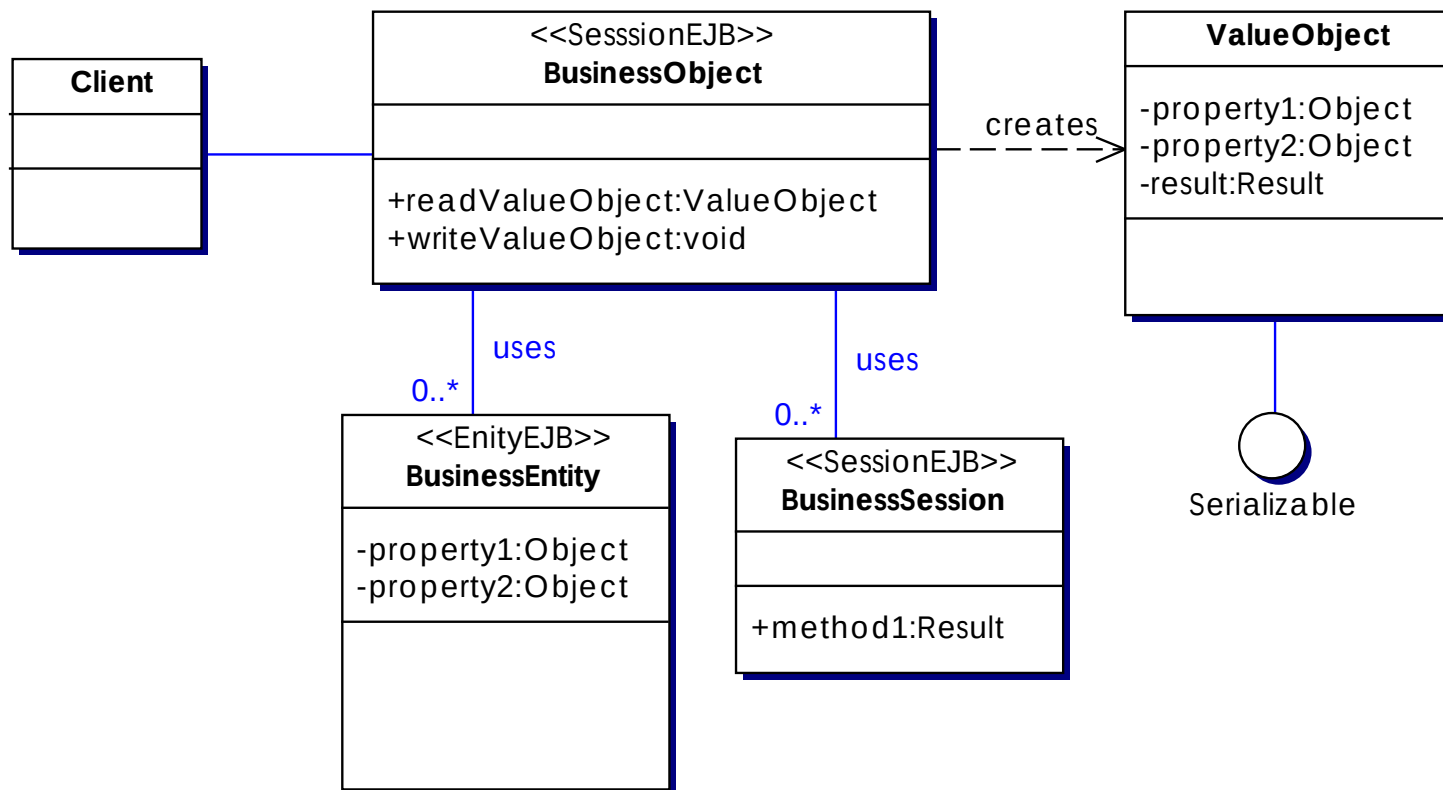
# Value Object (2)

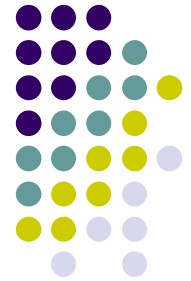
- Lösung
  - Einführung eines Transferobjektes, in dem mehr Daten mit einem Methodenaufruf transferiert werden können
- Sich ergebende Probleme
  - Konsistenzprobleme, falls Daten im Transferobjekt vom Client verändert werden können
  - anschließend muss Server-Objekt aktualisiert werden → spezielle Methode
  - Bei nur lesenden Zugriff auf Transferobjekt ergeben sich keine Probleme

# Value Object (3)



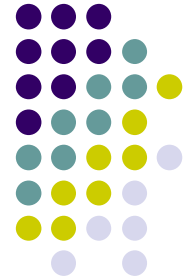
- UML-Darstellung





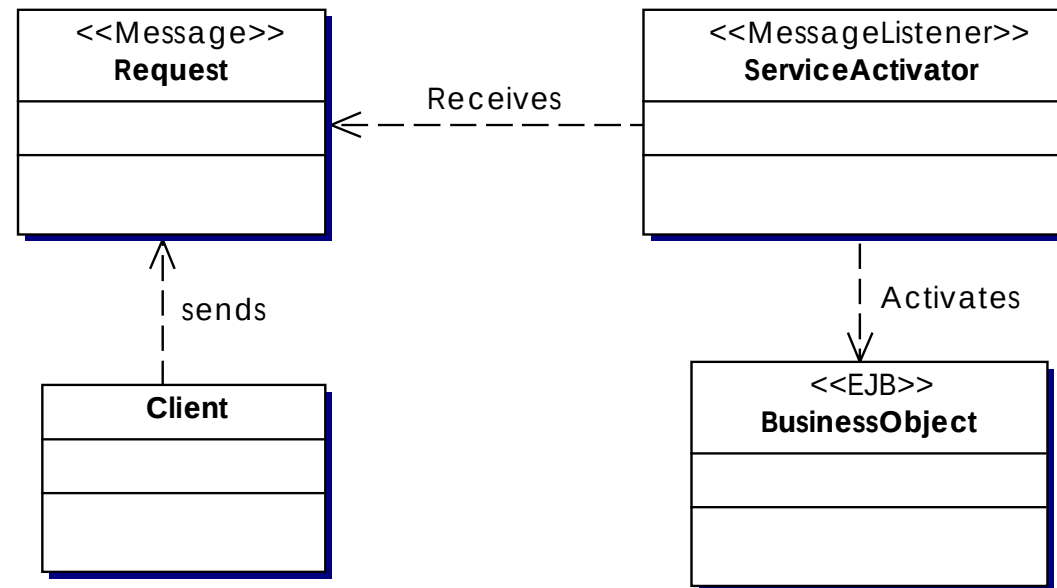
# Service Activator (1)

- Problem
  - Asynchrones Ausführen von Diensten soll möglich sein, z.B. beim Eintreten von bestimmten Ereignissen im System
- Lösung
  - Zuhilfenahme einer Message Driven Bean (ab EJB Spezifikation 2.0)
- Vorteile
  - Absender benötigt keine Kenntnisse des Empfängers (der EJB), sendet nur entsprechende Nachricht an eine Warteschlange



# Service Activator (2)

- UML-Darstellung



# Aufarbeitung des Hotelbeispiels (1)

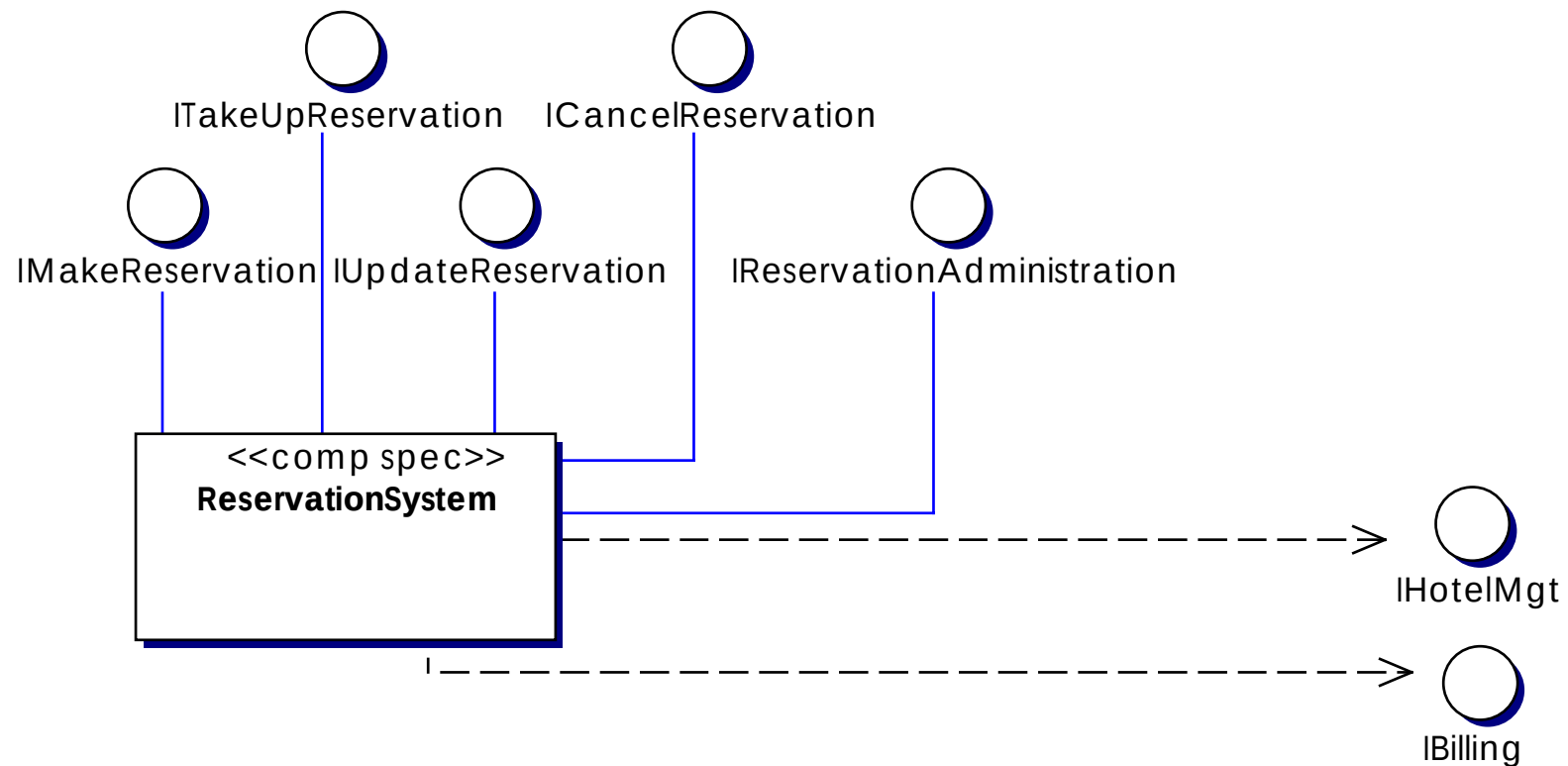


- Ziel 1:
  - Technologieunabhängige Spezifikation des Hotelbeispiels
  - Vorgehen nach dem in [Cheesman] vorgestellten Prozess
    - Anforderungsanalyse
      - Darstellung und Beschreibung der Geschäftsprozesse
      - Entwurf des Geschäftsmodells
      - Erstellung von Anwendungsfällen
      - Festlegung der Systemgrenze
    - Spezifikationsentwicklungsprozess
      - Identifizierung der Komponenten
      - Interaktion der Komponenten
      - Detailspezifikation des Systems

# Aufarbeitung des Hotelbeispiels (2)

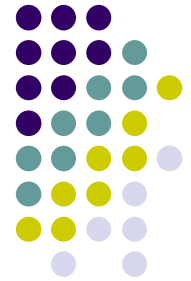


- System-Architektur



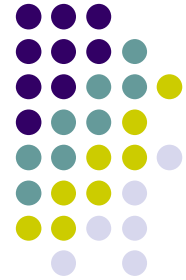


# Aufarbeitung des Hotelbeispiels (3)

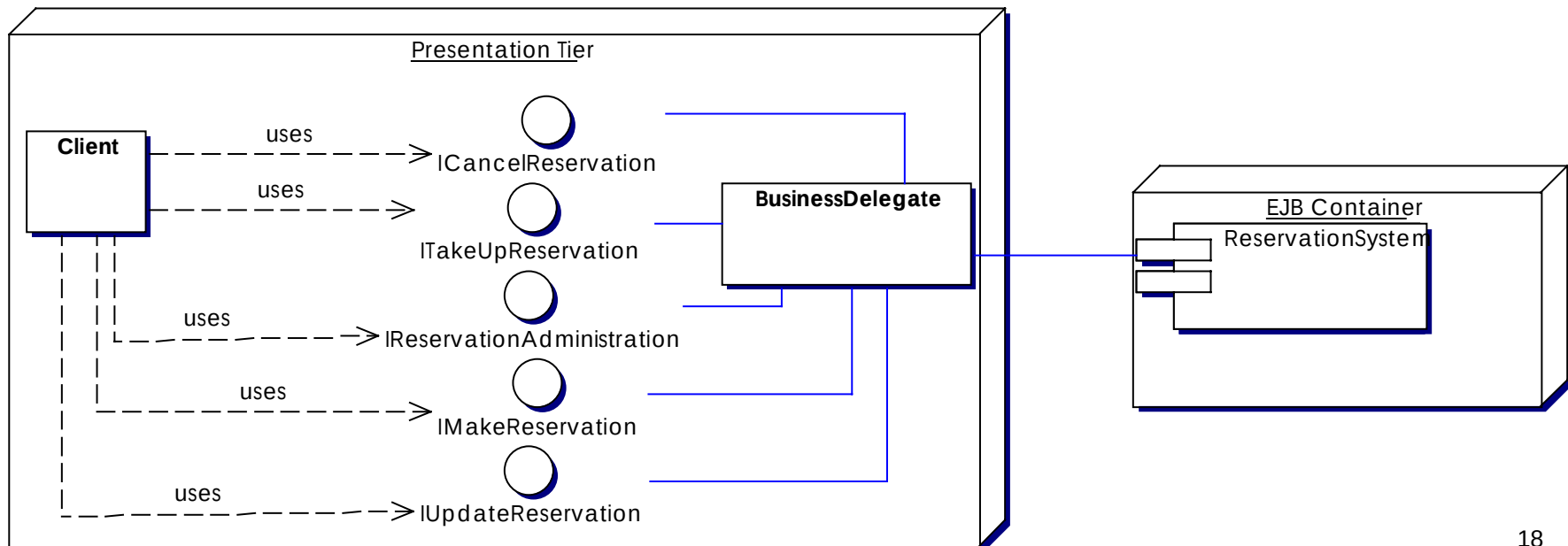


- Ziel 2:
  - Anpassung an EJB 2.0 und prototypische Realisierung
  - Anwendung von EJB-Entwurfsmustern
- Neuerungen in EJB-Spezifikation 2.0
  - Message Driven Beans
  - Lokale Interfaces
  - Vereinfachung der Containerverwalteten Persistenz
  - EJB-QL als Abfragesprache

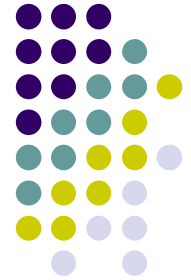
# Anpassung an EJB-Technologie (1)



- Probleme bezüglich unabhängiger Spezifikation
  - EJBs unterstützen nur ein fachliches Interface
    - Separation auf Client-Seite
    - Verwendung von mehreren Session Beans



# Anpassung an EJB-Technologie (2)

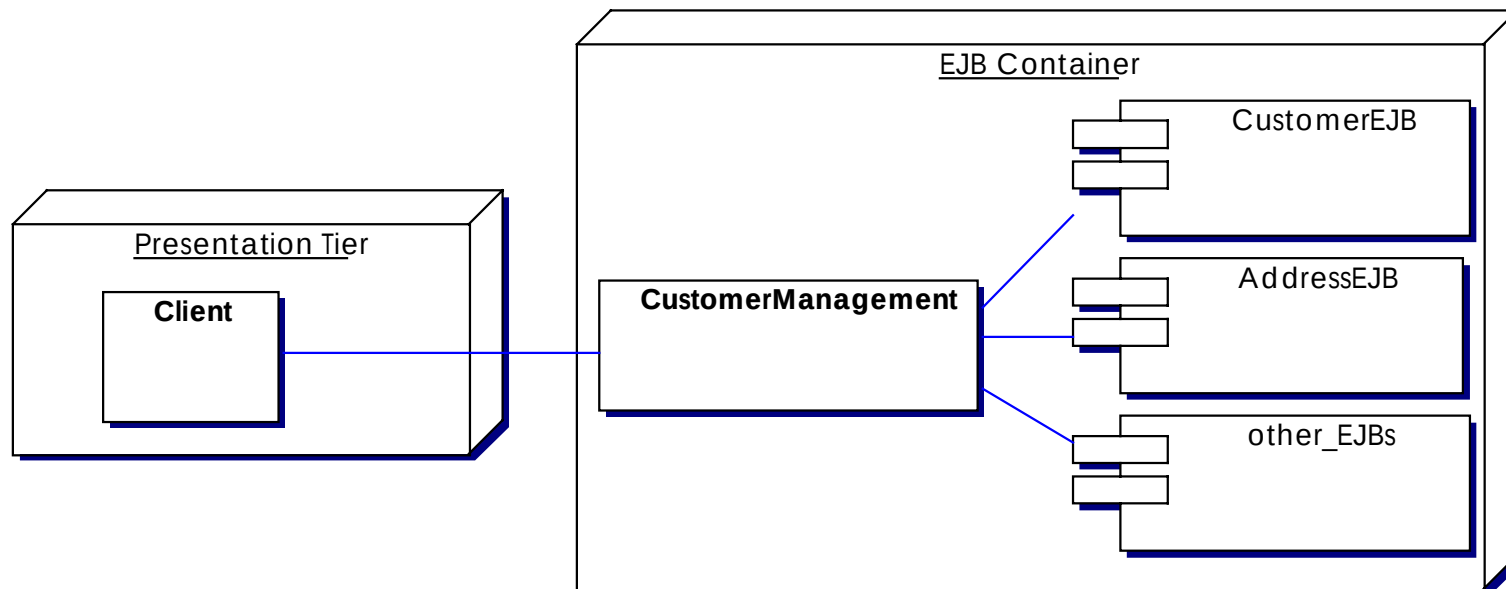


- Probleme bezüglich unabhängiger Spezifikation
  - Je Methode nur 1 Rückgabeparameter
    - Einführung eines neuen Objektes
  - keine parametrisierten Attribute
    - Stattdessen Verwendung einer neuen Methode

# Anwendung der EJB-Entwurfsmuster (1)



- Session Facade
  - Einheitliche Schnittstelle auf Server-Seite bei der Kundenverwaltung

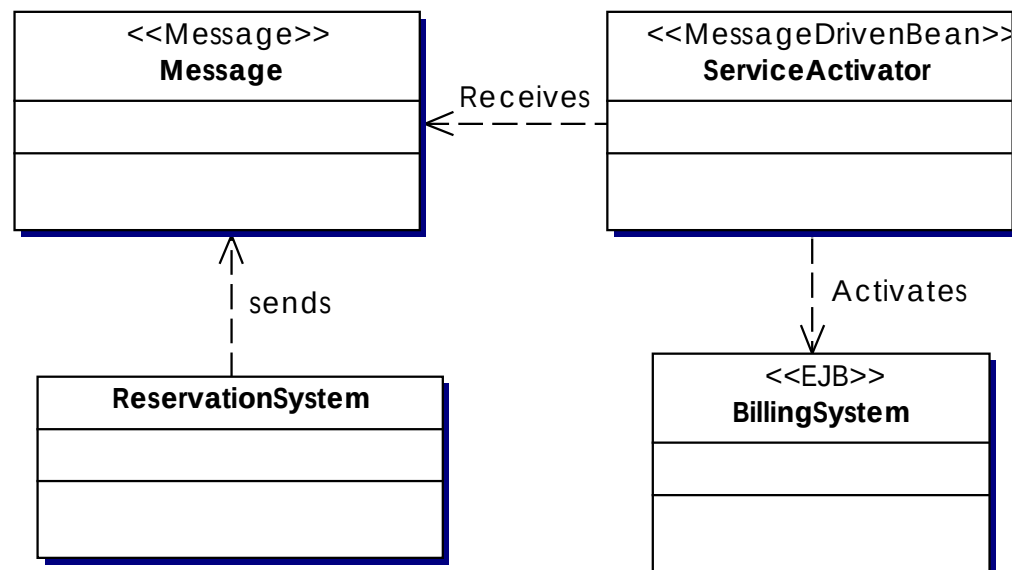


# Anwendung der EJB-Entwurfsmuster (2)

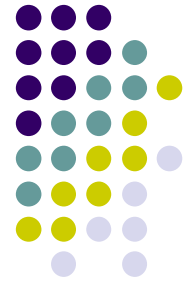


- Value Object
  - Transfer von Kundendaten zum Client

- Service Activator
  - Information des Rechnungssystems

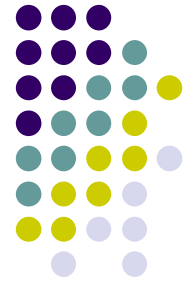


# Primärschlüsselgenerierung für Entity Beans



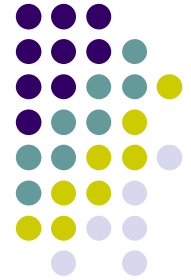
- Möglichkeiten der Primärschlüsselgenerierung für Entity Beans
  - Verwendung eines Schlüssel aus mehreren Attributen
  - Zufallszahl und Prüfung, ob Primärschlüssel vorhanden
  - Noch kein Entwurfsmuster
  - Besser wäre eine Möglichkeit einer automatischen Generierung durch Container

# Zusammenfassung und Ausblick



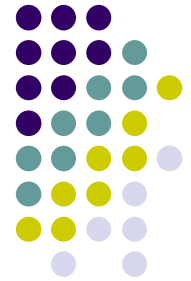
- Einarbeitung in bestehende EJB-Entwurfsmuster
- Aufarbeitung des Hotelbeispiels und Anpassung an EJB 2.0
- Vervollständigung der prototypischen Implementierung des Beispiels
- Erstellung eines Kataloges wichtiger EJB-Entwurfsmuster basierend auf der Implementierung

# Ende





# Spezifikationsprozess nach Cheesman



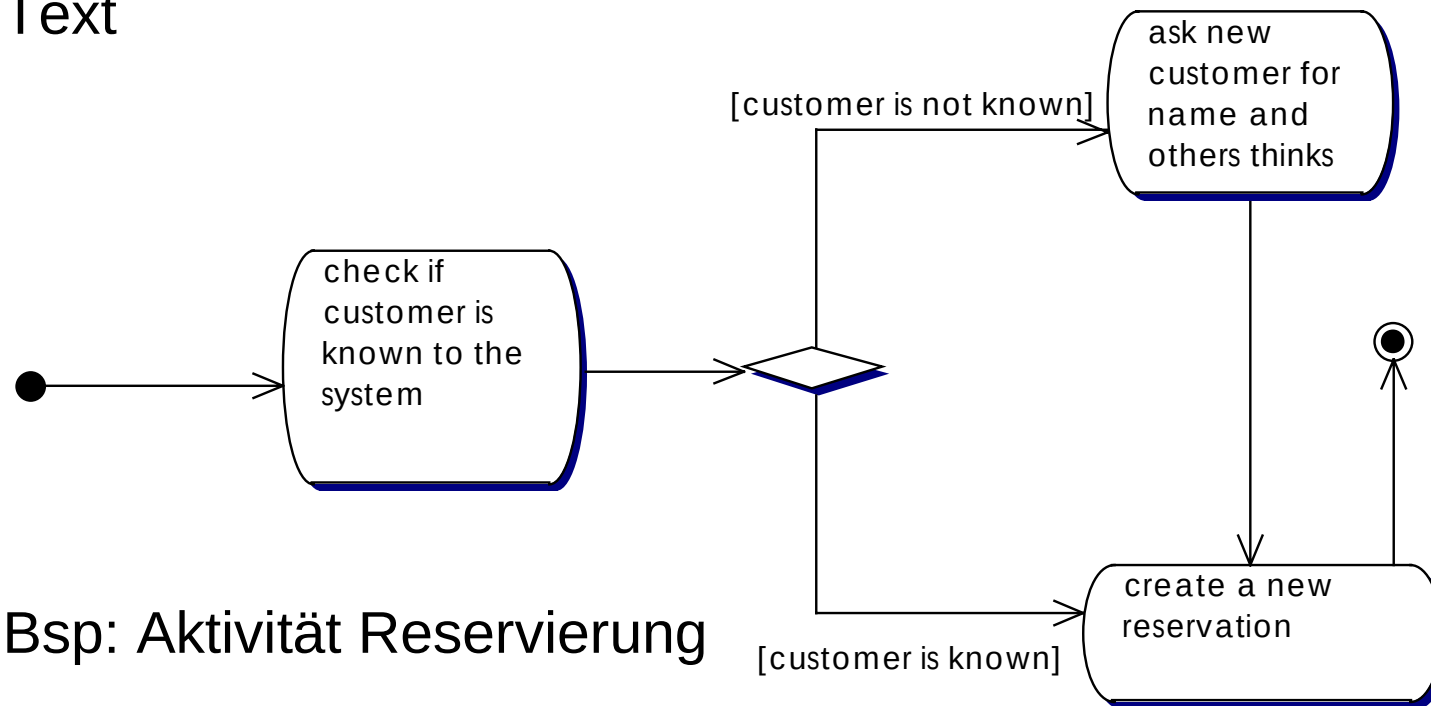
- Aufarbeitung des Hotelbeispiel nach dem in Cheesman vorgestellten Prozess

# Anforderungsanalyse (1)

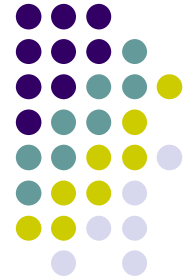


- Darstellung und Beschreibung der Geschäftsprozesse

- Aktivitätsdiagramme
- Text

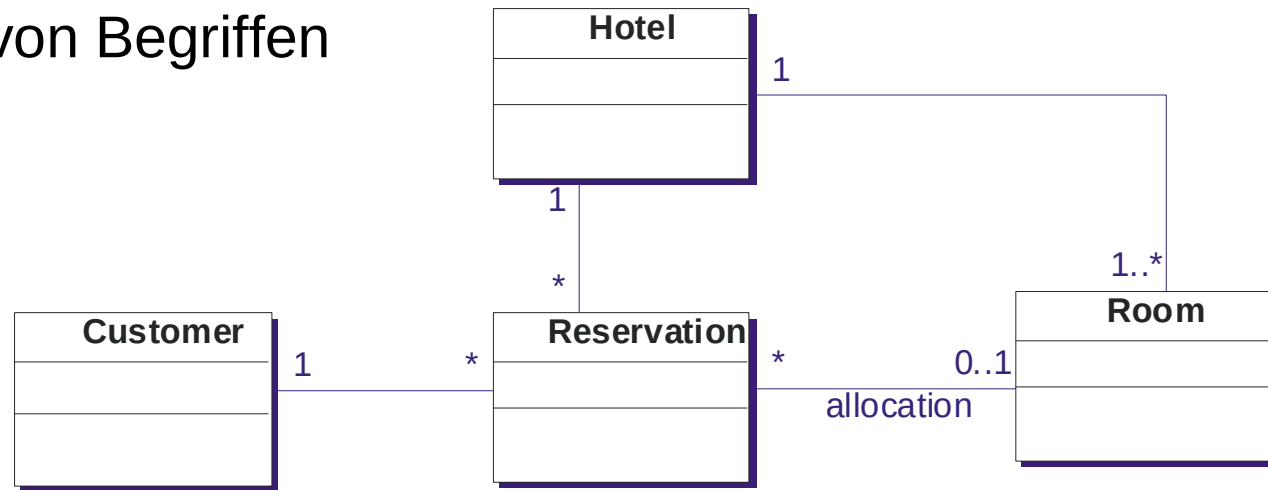


- Bsp: Aktivität Reservierung

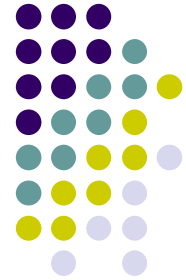


# Anforderungsanalyse (2)

- Entwurf des Geschäftsmodells
  - Darstellung der Beziehungen der einzelnen Objekte
  - Ähnlich UML Klassendiagramm
  - Klärung von Begriffen

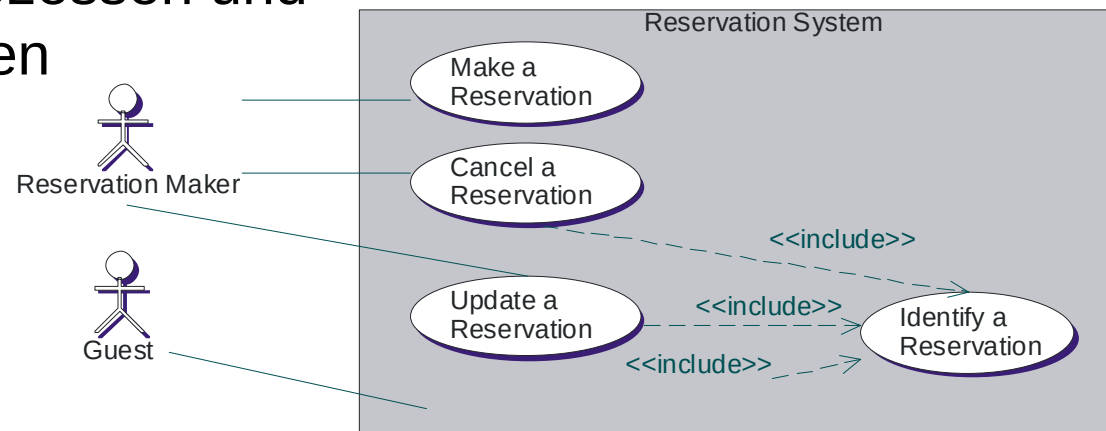


- Ausschnitt Entwurf Geschäftsmodell



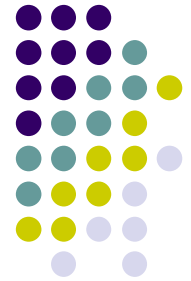
# Anforderungsanalyse (3)

- Erstellung von Anwendungsfällen
  - Identifikation von Akteuren
  - Aus Geschäftsprozessen und deren Beziehungen



- Festlegung der Systemgrenze
  - Welcher Teil soll im Endprodukt enthalten sein?

# Spezifikations- entwicklungsprozess (1)

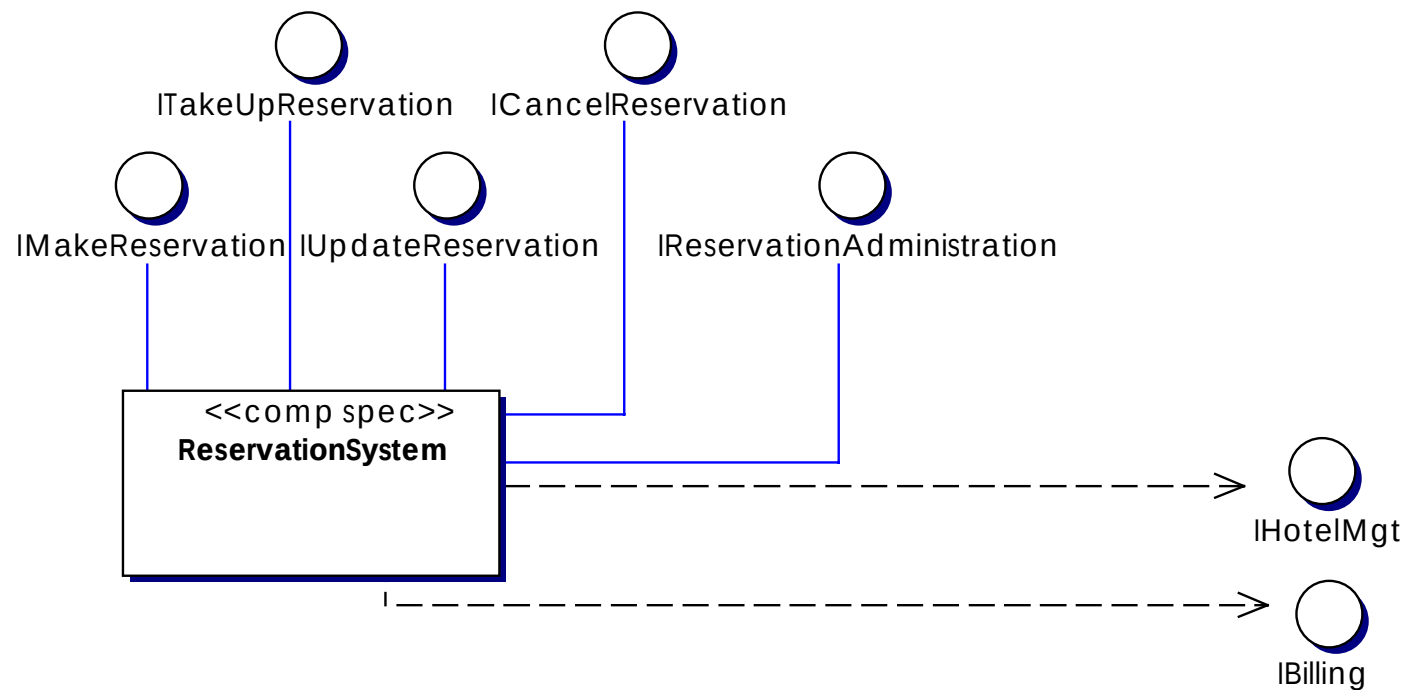


- Identifizierung der Komponenten
  - Ausgehend vom Entwurf des Geschäftsmodell und den Anwendungsfällen Identifizierung einer ersten Menge von Interfaces für die Komponenten
  - Benötigte Operationen finden (noch ohne Details)
  - Interfaces:
    - System-Interfaces: Interfaces, auf die Anwender zugreift
    - Geschäftslogik-Interfaces: Interfaces innerhalb des Systems
  - Erster Entwurf der System-Architektur

# Spezifikations- entwicklungsprozess (2)



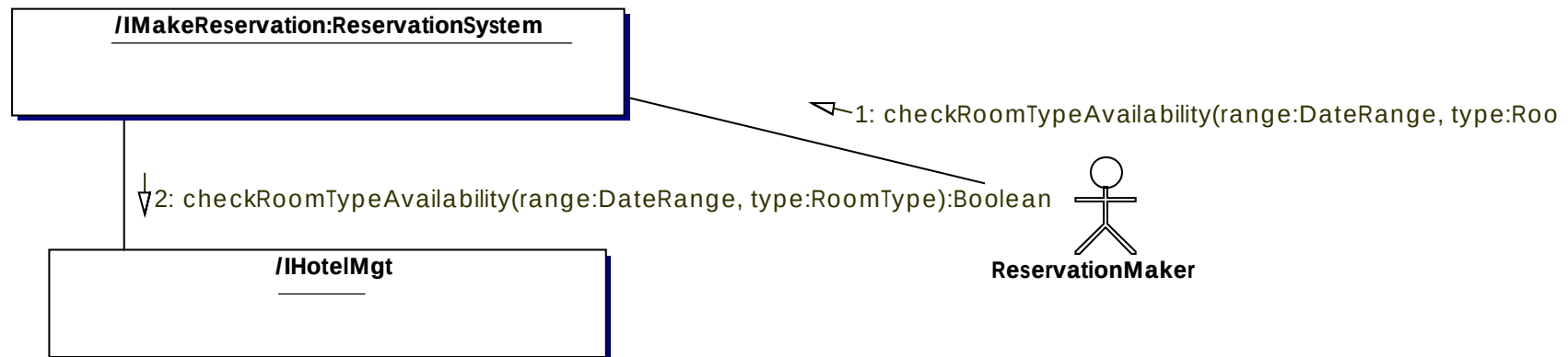
- Identifizierung der Komponenten
  - Erster Entwurf der System-Architektur



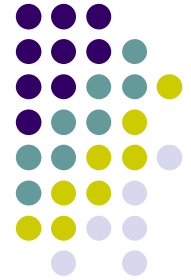
# Spezifikations- entwicklungsprozess (3)



- Interaktion der Komponenten
  - Darstellung der Interaktion
  - Operationsparameter der Geschäftslogik-Interfaces finden

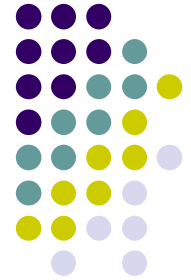


# Spezifikations- entwicklungsprozess (4)



- Detailspezifikation des Systems
  - Iterative Verfeinerung des Modell
  - Einführung von Hilfsklassen (Datumsbereich, etc.)





# Literatur

- Cheesman
  - [John Cheesman, John Daniels. UML Components A Simple Process for Spezifying Component-Based Systems. Addison-Wesley, Amsterdam, 2001.