Desenvolvimento de Software Baseado em Componentes

Rosana T. Vaccare Braga

(material resumido a partir de slides do Prof. Paulo Masiero)

Introdução

- Frustração com as promessas da Orientação a objetos em relação ao reúso de classes.
- Frameworks são uma solução para um domínio específico que consideram uma arquitetura OO composta de várias classes.
- DSBC surgiu nos anos 90 como uma solução mais ampla e independente

Introdução

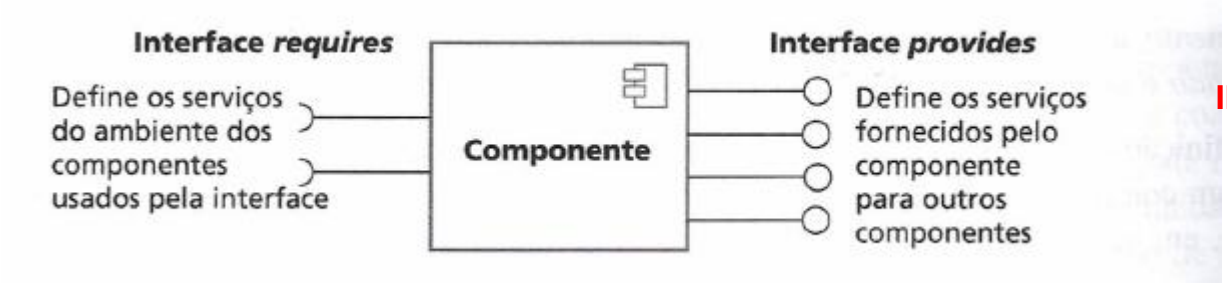
- CSBC: processo de definição, implantação e integração ou composição de componentes independentes, não firmemente acoplados ao sistema.
- CSBC tem quatro pontos principais:
  - Componentes independentes.
  - Padrões de componentes.
  - Middleware para apoiar a integração.
  - Processo de desenvolvimento.

Problemas

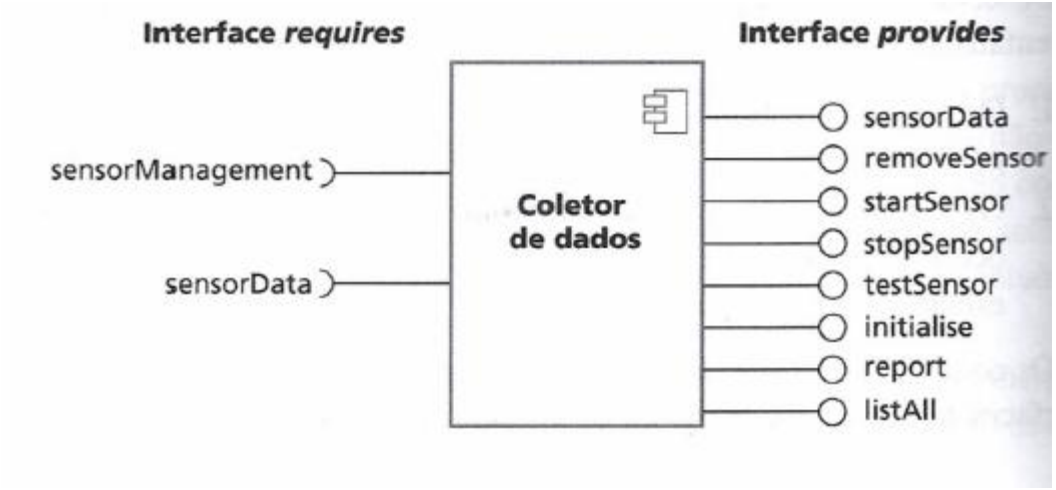
- Confiabilidade dos componentes
- Certificação de componentes
- Previsão de propriedades emergentes
- Compromisso de requisitos (como selecionar e configurar componentes?)

Componentes e modelos de componentes

- Um componente de software é uma unidade de composição com **interfaces contratualmente especificadas** e **dependências de contexto explícitas**.Um componente de software pode ser implantado independentemente e está sujeito a composição por terceiros
- É uma unidade executável independentemente
- Os serviços oferecidos são disponibilizados somente por meio de uma interface

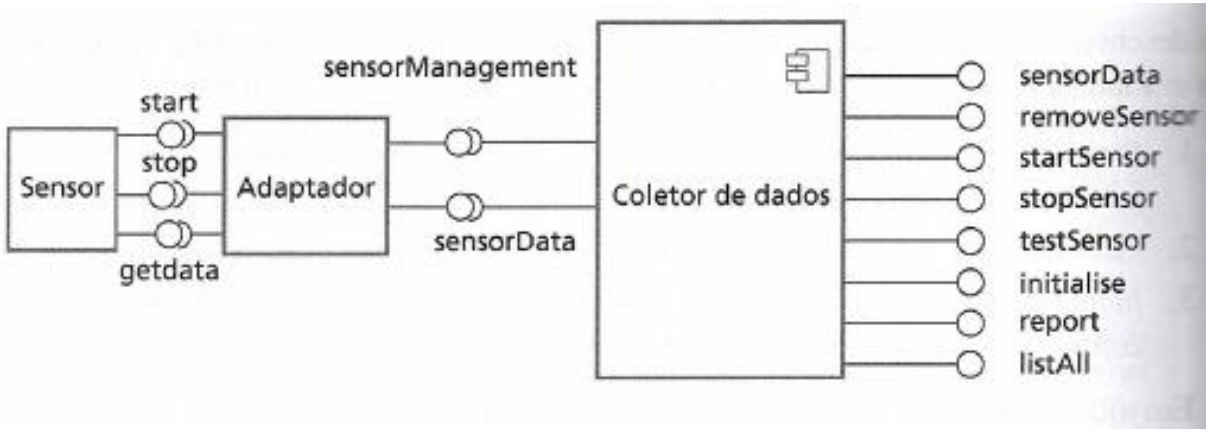


INTERFACES

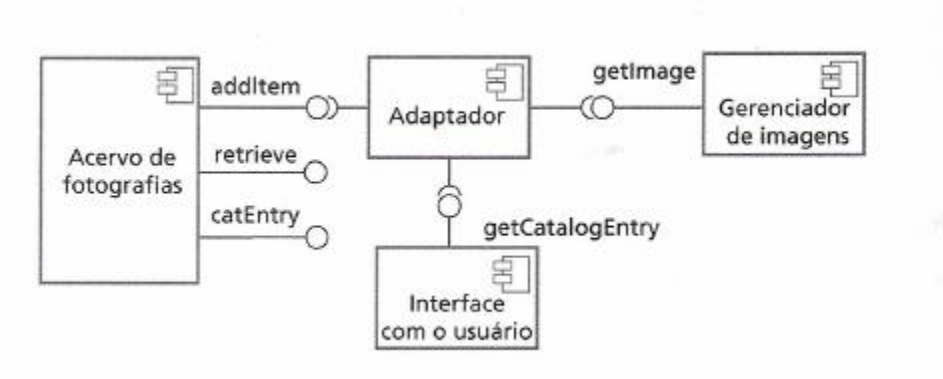


EXEMPLO

Exemplo - 1

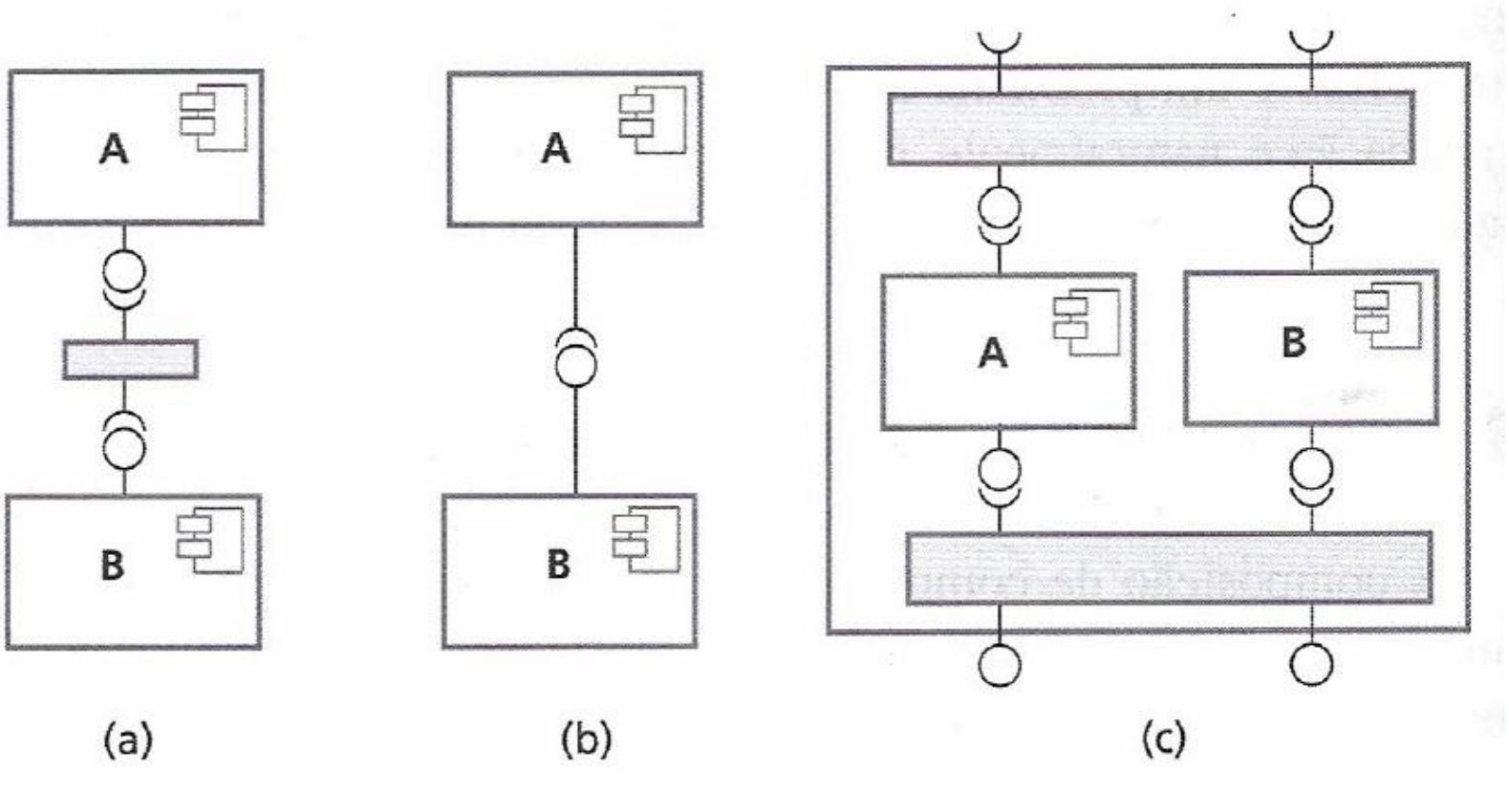

sensorManagement
start
stop
getdata
Sensor
Adaptador
sensorData
Coletor de dados
sensorData
removeSensor
startSensor
stopSensor
testSensor
initialise
report
listAll

Exemplo - 2




```
public void addItem (Identifier pid ; Photograph p; CatalogEntry photodesc) ;
public Photograph retrieve (Identifier pid) ;
public CatalogEntry catEntry (Identifier pid) ;
```

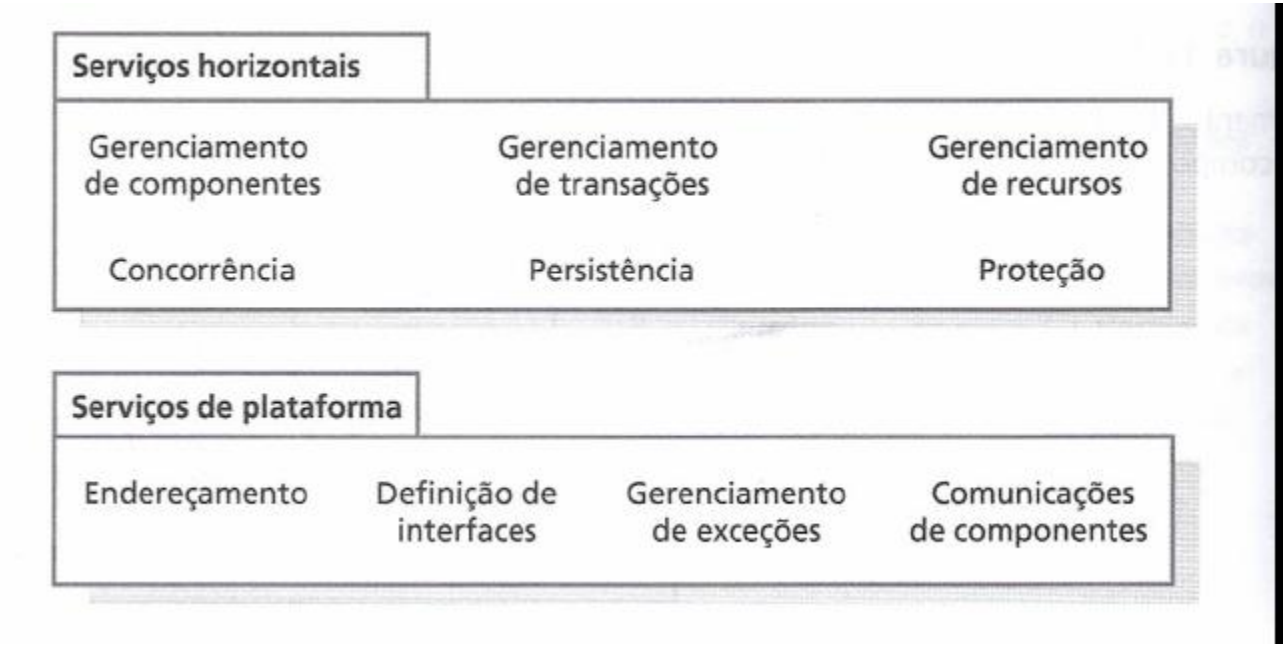
Composição de componentes



Sequencial Hierárquica Aditiva

Modelos de Componentes

- CORBA (OMG)
- COM+ (MICROSOFT)
- EJB: Enterprise Java Beans (SUN)
- Serviços oferecidos:



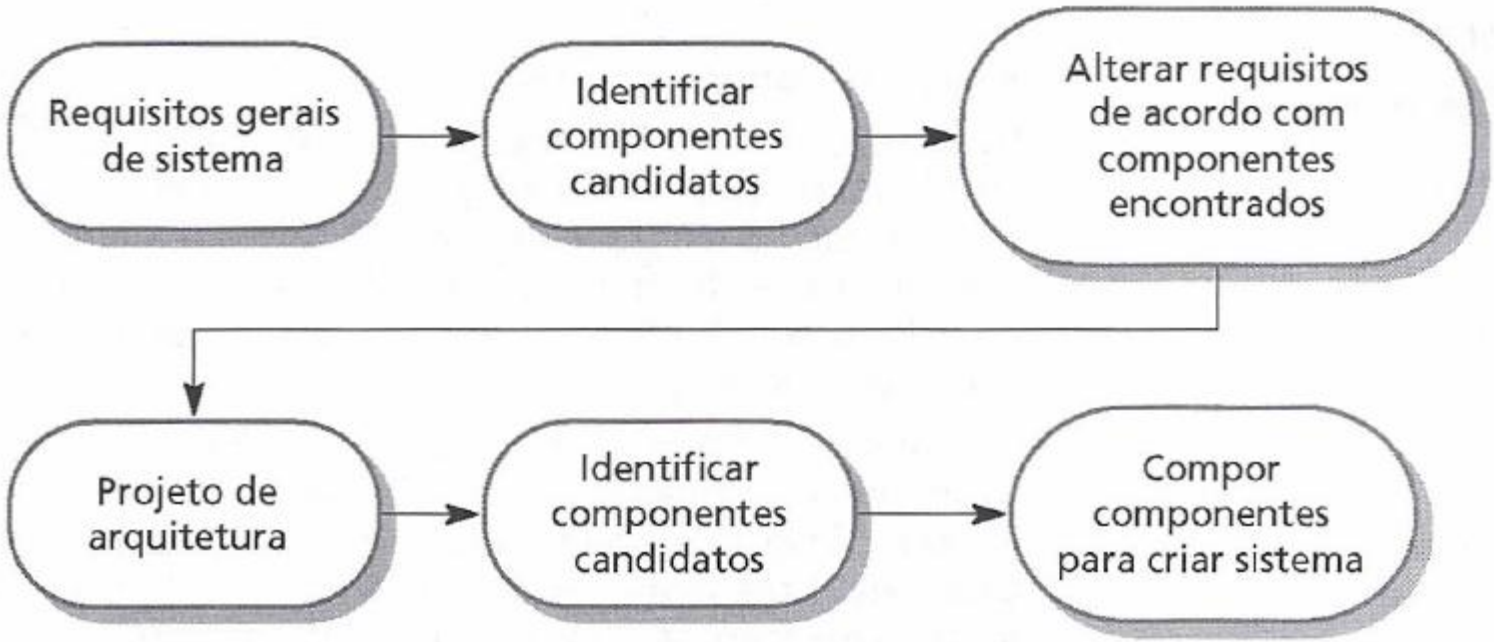
Desenvolvimento de componentes para reúso

- Visão de longo prazo: haverá um mercado de fornecedores de componentes.
- Componentes desenvolvidos internamente não são em geral usáveis imediatamente
- Para ser reusável, o componente deve ser tratado e isso tem um custo: documentação, validação e generalização.
- Mudanças:remover métodos específicos, tornar nome mais geral, tratar exceções, definir uma interface de “configuração” e integrar com outros componentes

O processo de DSBC

- Há alguns métodos propostos para o DSBC
  - UML Components
  - Catalysis
- Em linhas gerais:
  - Especificar os requisitos do sistema (é preciso um conjunto completo)
  - Refinar e modificar os requisitos, dependendo dos componentes disponíveis
  - Projetar a arquitetura do sistema e buscar componentes
  - Desenvolver compondo os componentes

CSB "com" Reúso



Identificar Componentes

Obs. Des. com reúso vs Des. para reúso

UML Components
Cheesman and Daniels

- Objetivos dos componentes
- Formas de componentes
- Arquiteturas de Sistema e de Componentes
- Processos (*workflows*)
- Artefatos do processo
- Processo de Especificação

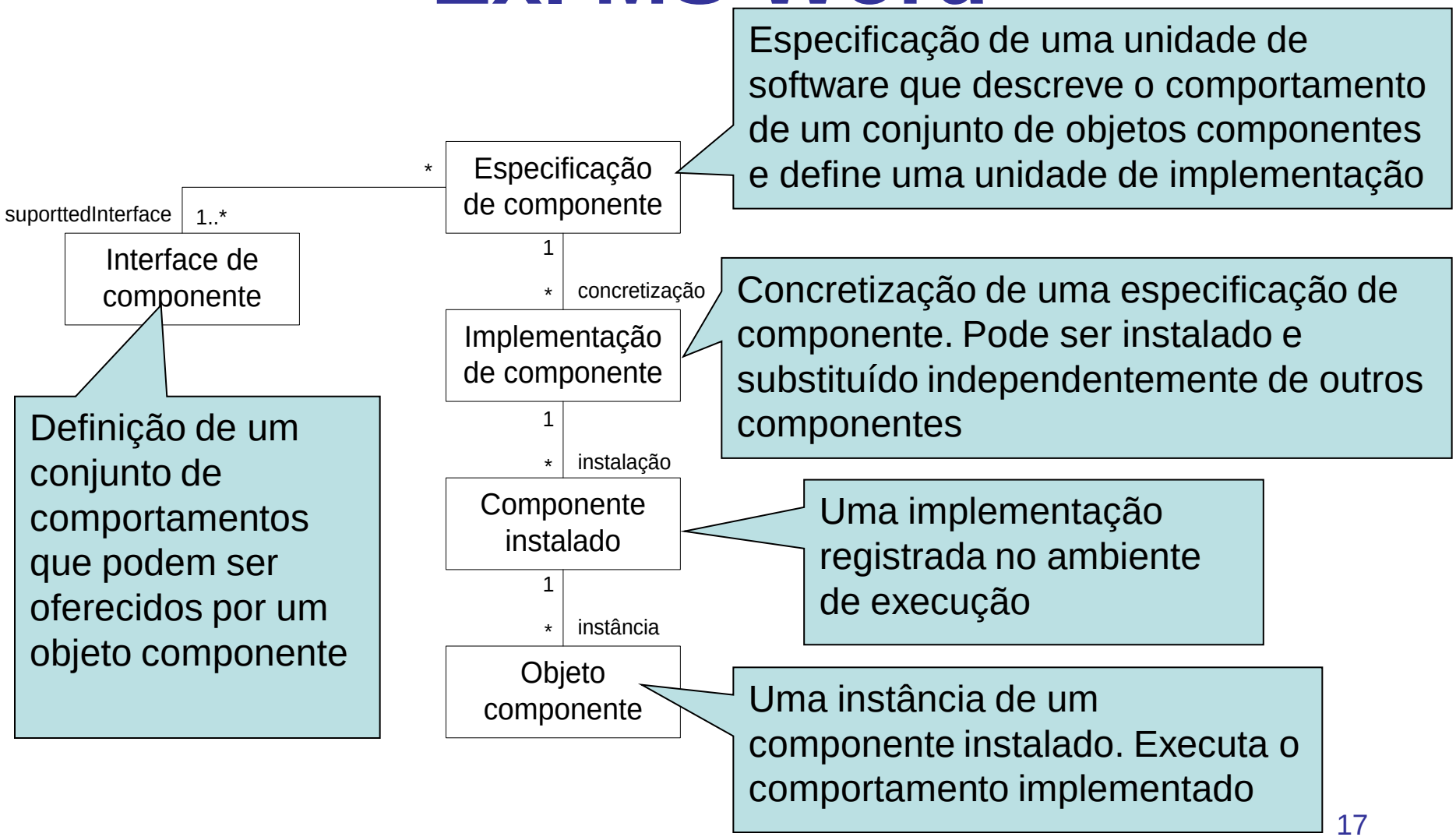
Objetivos dos componentes

- Complexidade
  - Princípio “dividir e conquistar”
- Funções e dados combinados
- Motivação: gestão da mudança
  - “construir para mudar”
  - Ênfase durante arquitetura e projeto nas dependências entre componentes e gestão das dependências
- Devem ser facilmente substituíveis
- Gestão da mudança x reúso

Formas de componentes

- A visão do componente muda durante o ciclo de vida do projeto
- Há diversas formas de componentes e cada uma reflete algum aspecto do componente durante o ciclo de desenvolvimento
- Definição das diversas formas do componente, ao invés de definição de componente

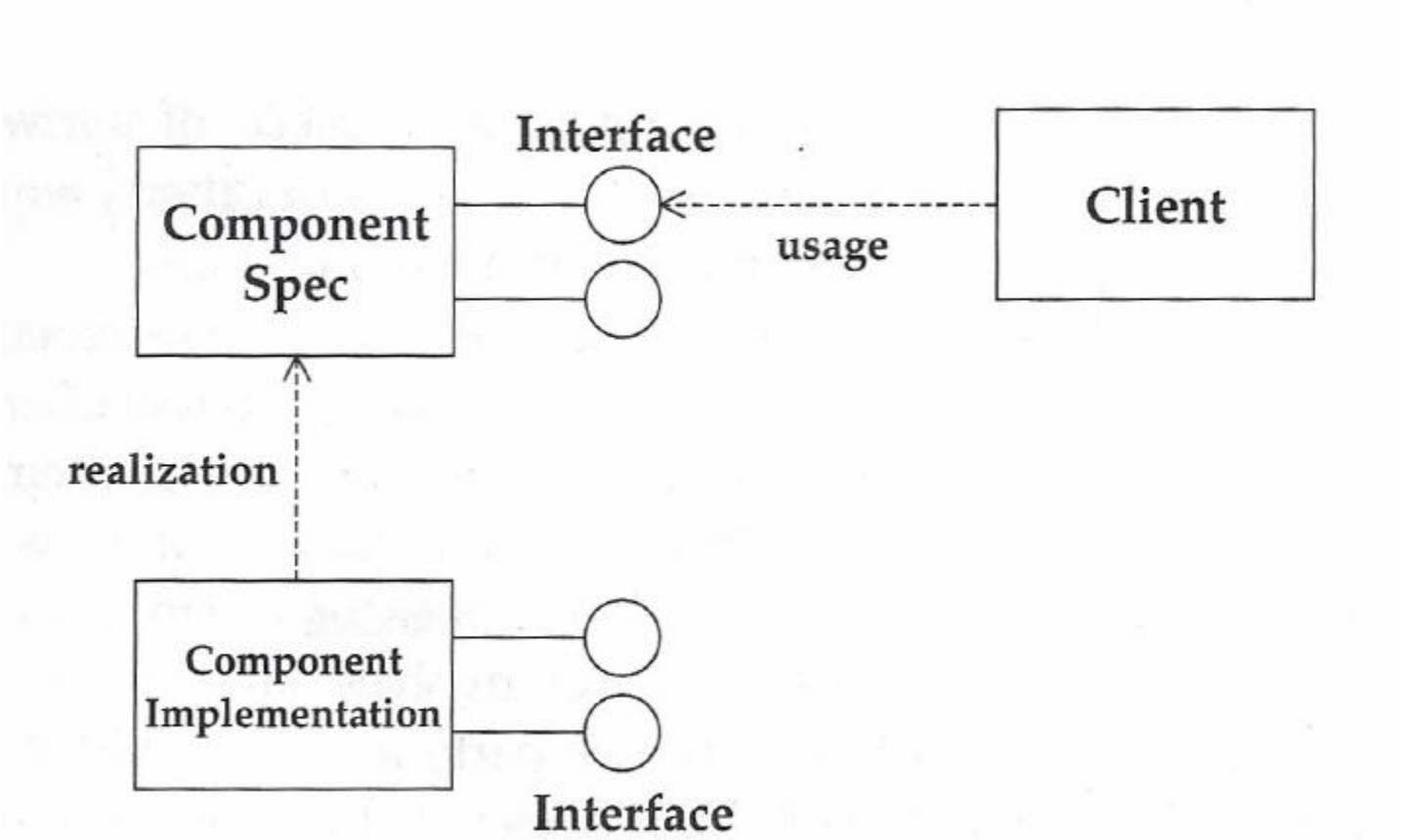
Formas de componentes
Ex. MS Word



Formas de componentes

- Em tempo de execução o componente possui conteúdo (estado)
  - Além dos serviços providos pelo componente, a informação gerenciada também é importante
  - Na substituição do componente deve-se garantir os serviços e as informações

Contratos de uso e de implementação

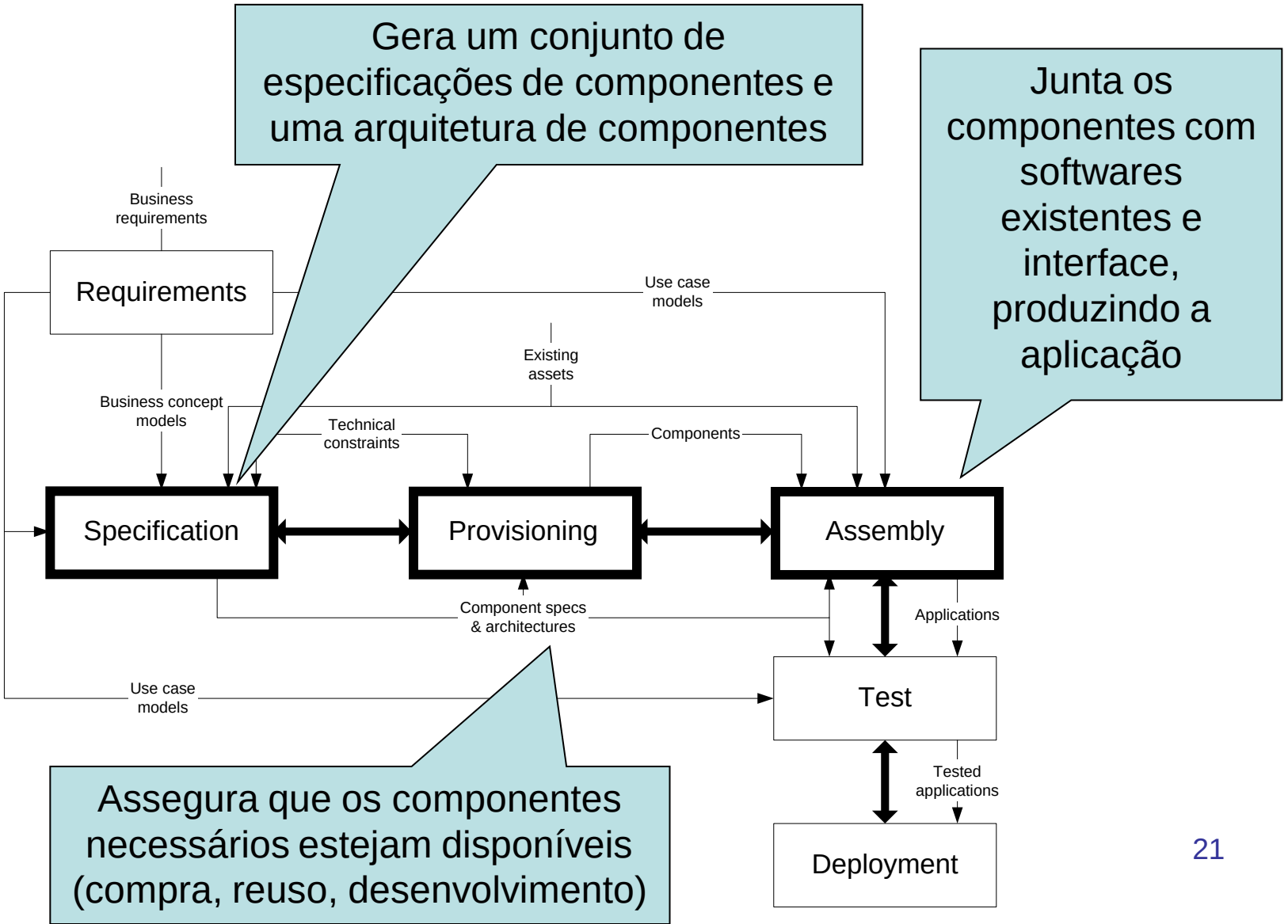


0 Different types of contracts

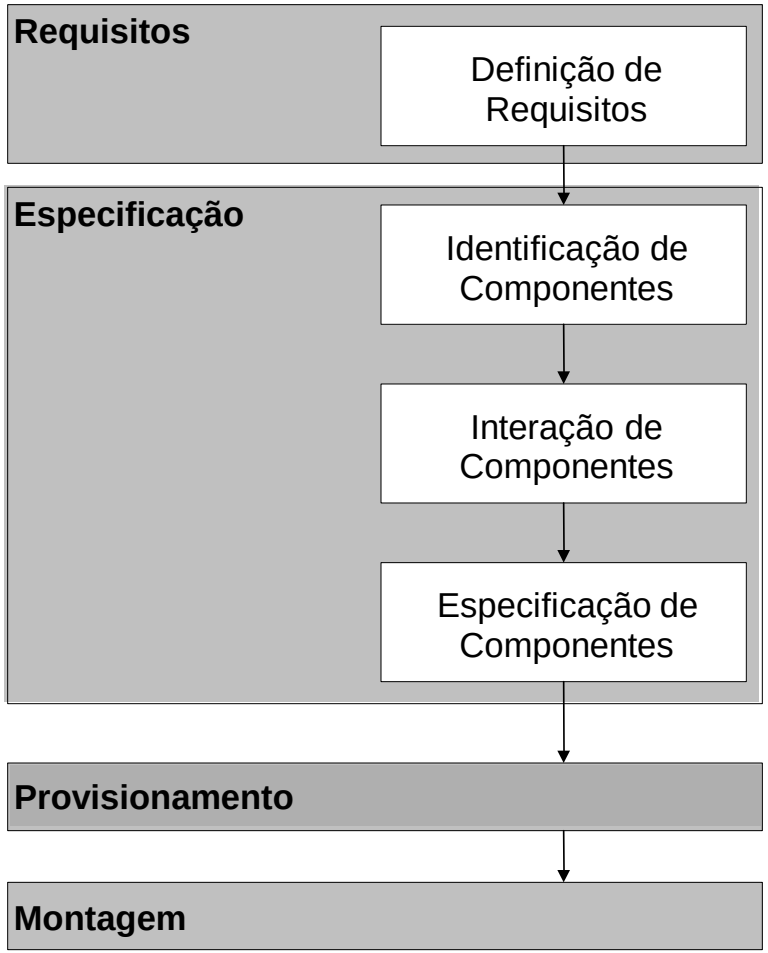
Contratos de uso

- Lista das operações
- Cada operação, além da sua assinatura é definida por uma pré-condição e uma pós-condição (Pode ser usado OCL para isso).
- Modelo de Informação : informação ou estado que é mantido (persistido) entre requisições de clientes.

Processo de Desenvolvimento



Sub-processos (ou workflows)



- Modelo de conceito de negócio
- Modelo de caso de uso

- Modelo de tipo de negócio
- Especificações de interface
- Especificações de componente
- Arquitetura de componente

Artefatos do processo Requisitos

- Modelo de conceitos de negócio
  - Cria um vocabulário comum entre os envolvidos no projeto
- Modelo de casos de uso
  - Descreve as interações entre um usuário e o sistema e auxilia na identificação dos limites do sistema

Artefatos do processo Especificação

- Modelo de tipos de negócio
  - Formaliza o modelo de conceito de negócio e define o conhecimento que o sistema possui do mundo externo
- Especificações de interface
  - Conjunto de especificações de interface
  - Cada especificação de interface é um contrato com um cliente de um objeto componente

Artefatos do processo Especificação

- Especificações de componente
  - Conjunto de especificações de componente
  - Cada especificação de componente é definida em termos de especificações e restrições
- Arquitetura de componente
  - Descreve como especificações de componentes são combinadas em uma determinada configuração

Visão do Sistema

Deseja-se desenvolver um sistema de reserva de hotel a ser feito para qualquer hotel de uma cadeia. Presentemente cada hotel tem seu próprio sistema de reservas e eles são incompatíveis entre si. As reservas podem ser feitas por telefone a uma central de reservas ou diretamente em cada hotel ou pela Internet. A maior vantagem do sistema será oferecer acomodações em hoteis alternativos quando o desejado está cheio. Cada hotel terá suas próprias instalações para fazer reservas na recepção, no escritório e na mesa do porteiro. Cada hotel tem um administrador de reservas que é responsável por controlar as reservas do hotel, mas qualquer usuário autorizado poderá fazer reservas. O tempo esperado para completar uma reserva por telefone é 3m. Para agilizar o processo, os detalhes de clientes anteriores serão armazenados e tornados disponíveis.

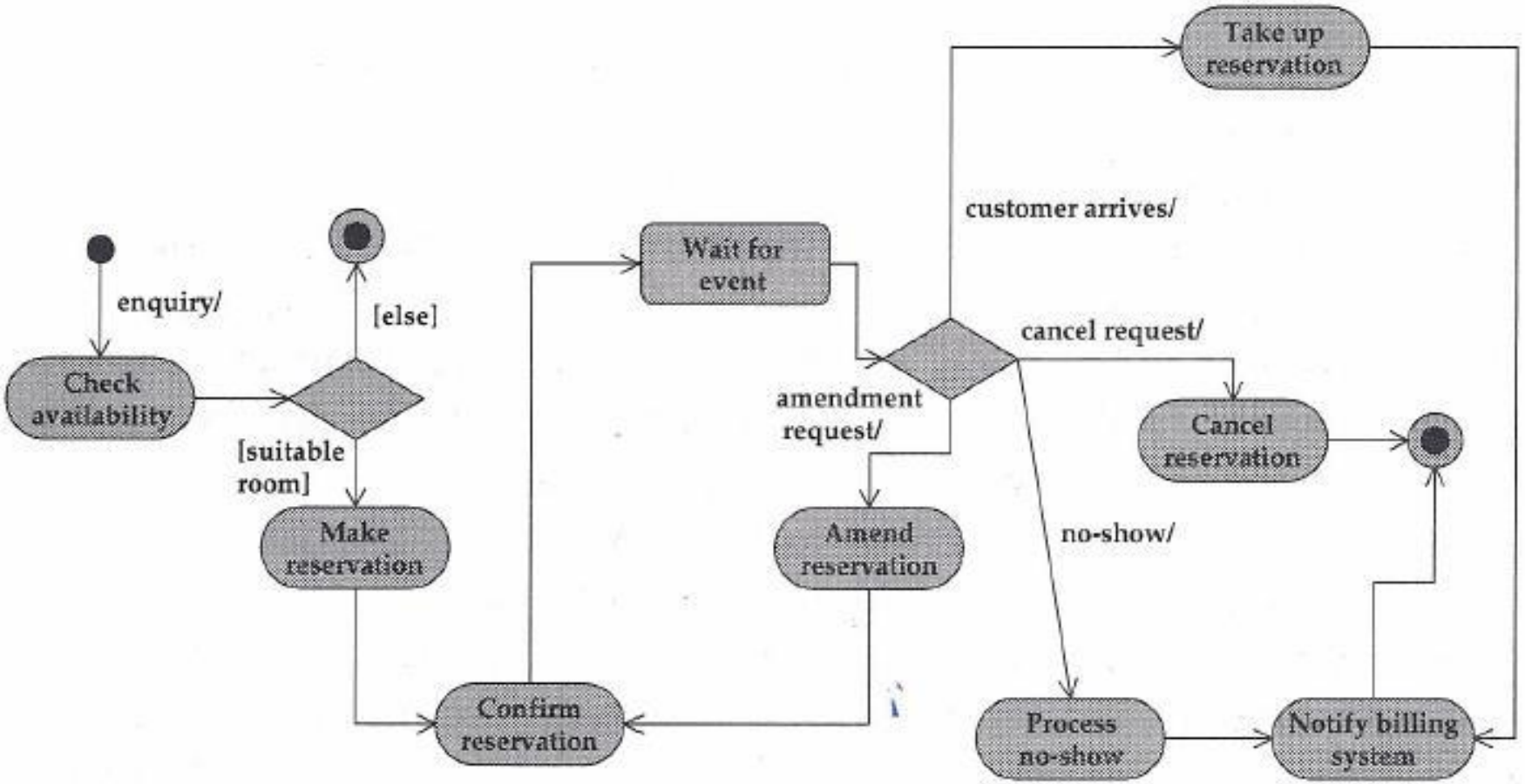


Figure 4.1 Business process for hotel reservation

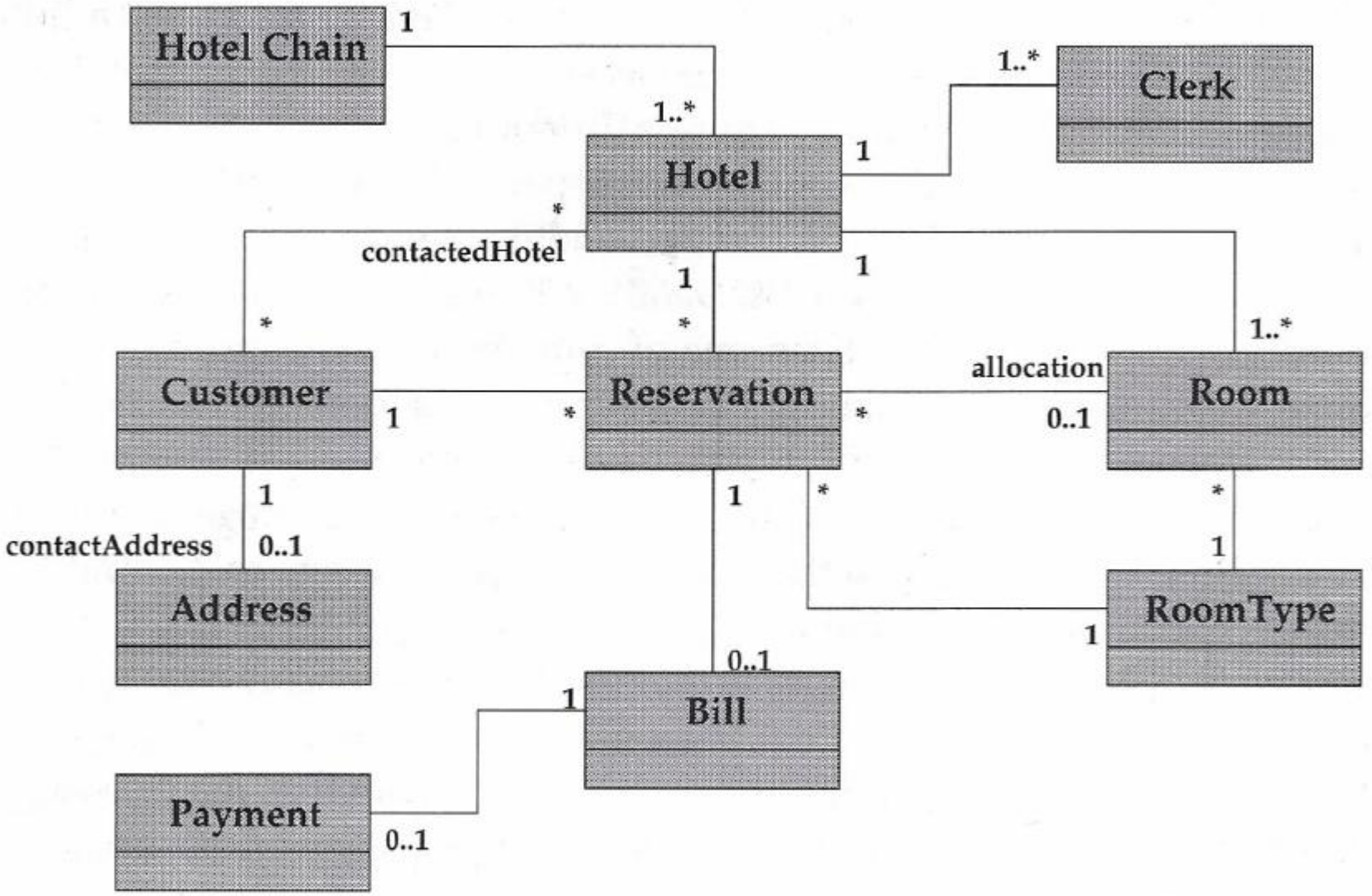
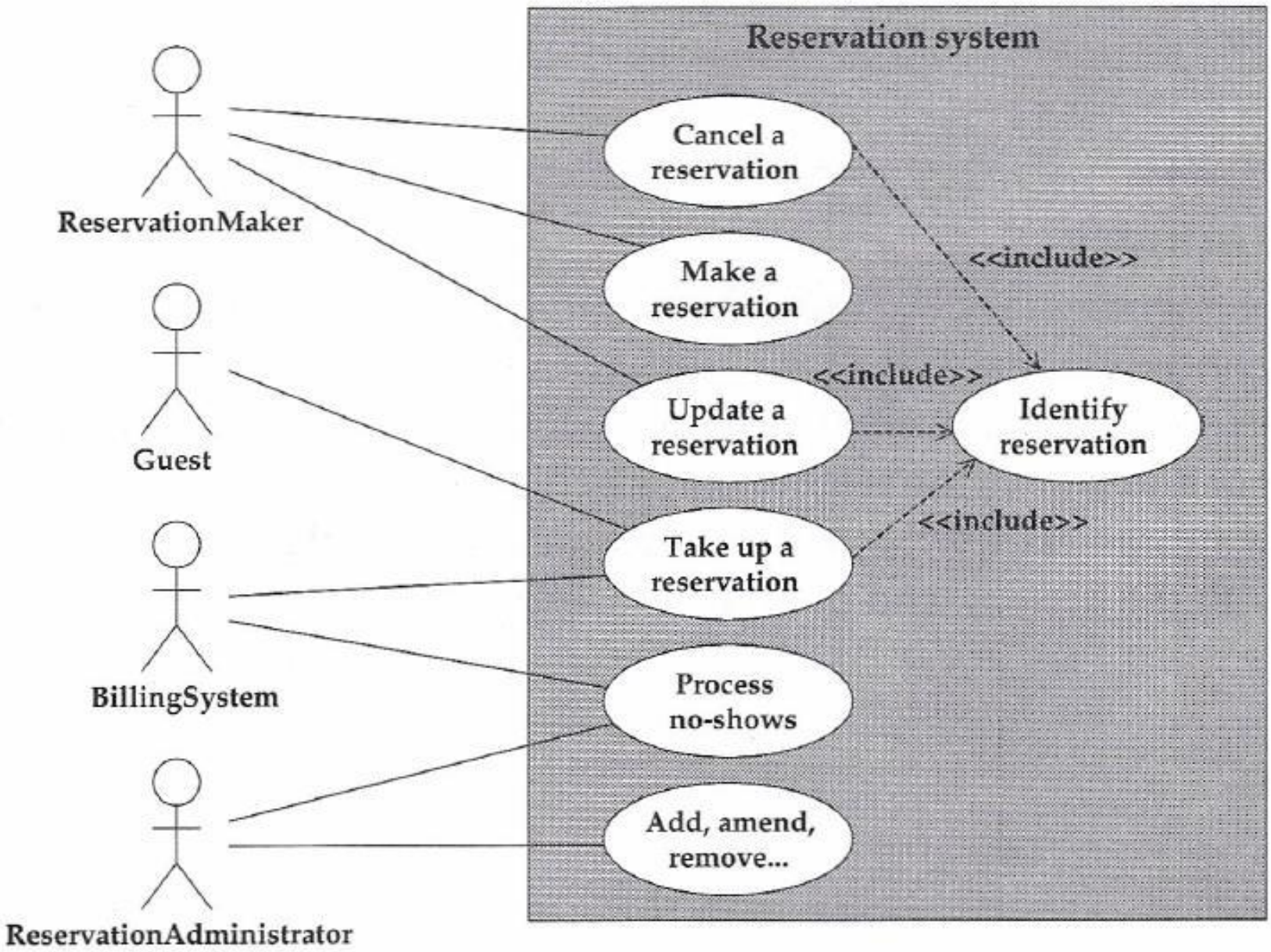


Figure 4.2 Business concept model for hotel reservation


Reservation system
ReservationMaker
Guest
BillingSystem
ReservationAdministrator
Cancel a reservation
Make a reservation
Update a reservation
Take up a reservation
Process no-shows
Add, amend, remove...
Identify reservation
<<include>>
<<include>>
<<include>>

Name	**Make a reservation**
Initiator	**Reservation Maker**
Goal	**Reserve room(s) at a hotel**

Main success scenario

1. Reservation Maker asks to make a reservation.
2. Reservation Maker selects, in any order, hotel, dates and room type.
3. System provides price to Reservation Maker.
4. Reservation Maker asks for reservation.
5. Reservation Maker provides name and post code (zip code).
6. Reservation Maker provides contact e-mail address.
7. System makes reservation and allocates tag to reservation.
8. System reveals tag to Reservation Maker.
9. System creates and sends information by e-mail.

Extensions

3. Room not available.
    a. System offers alternatives.
    b. Reservation Maker selects from alternative.

3b. Reservation Maker rejects alternative.
    a. Fail

4. Reservation Maker declines offer.
    a. Fail

5. Customer already on file (based on name and post code).
    a. Resume 7.

Name	**Identify reservation**
Initiator	**Included only**
Goal	**Identify an existing reservation**

Main success scenario

1. Actor provides reservation tag.
2. System locates reservation.

Extensions

2. System cannot find a reservation with the given tag.
    a. Actor provides name and post code.
    b. System dysplays active reservations for that customer.
    c. Actor selects the reservation.
    d. Stop.
2. The reservation tag refers to a reservation at a different hotel.
    a2. Fail

2b. No active reservations at this hotel for this customer.
    a. Fail

Name	**Take up reservation**
Initiator	**Guest**
Goal	**Claim a reservation and check in to the hotel**

Main success scenario

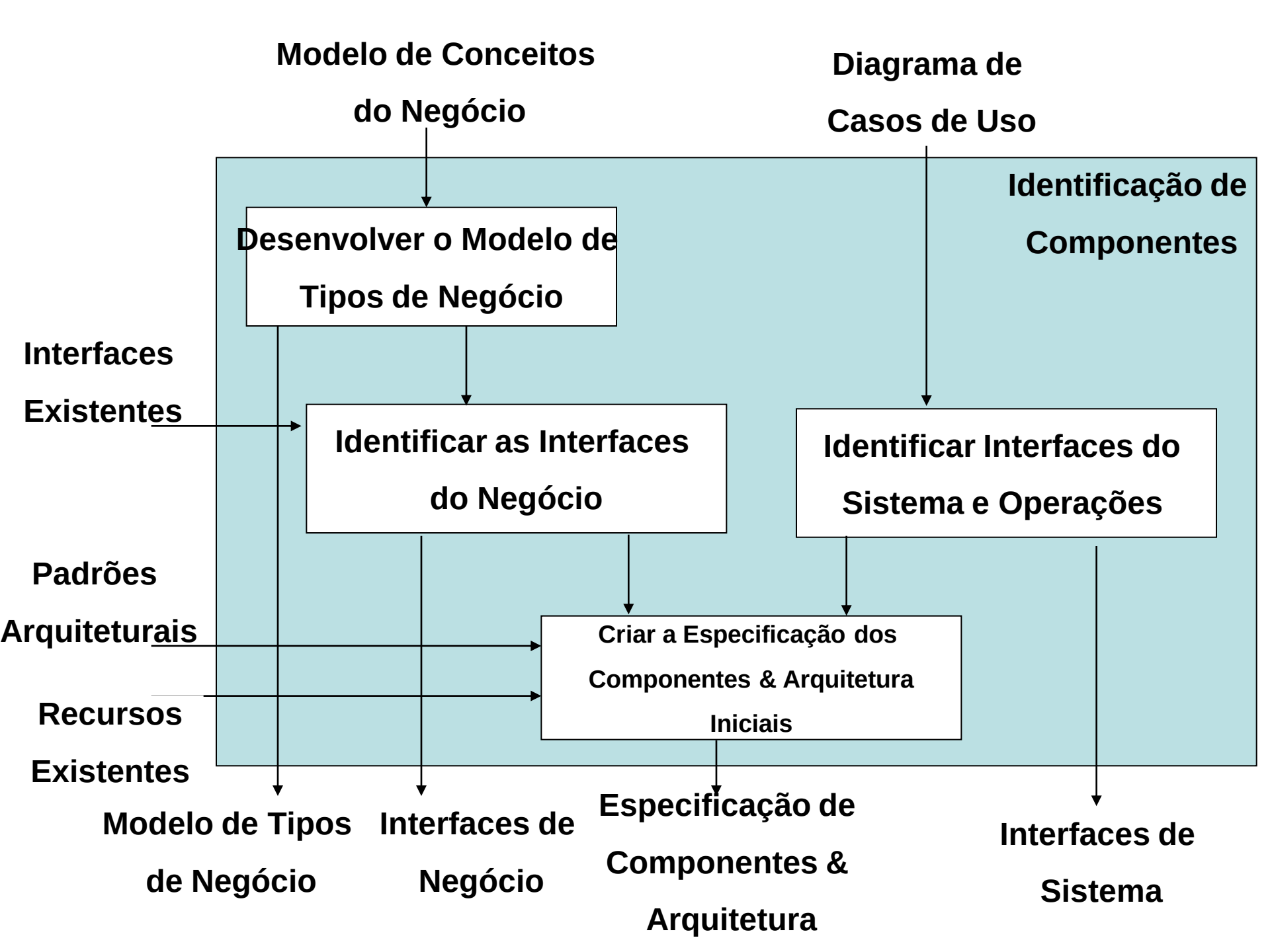
1. Guest arrives at hotel and claims a reservation.
2. Include Identify Reservation.
3. Guest confirms details of stay duration, room type.
4. System allocates room.
5. System notifies billing system that a stay is starting.

Extensions

3. Reservation not identified.
    a. Fail

Identificação de Componentes

- É o primeiro passo do processo de especificação de componentes
- Objetivo: criar um conjunto inicial de interfaces e de especificação de componentes.
- Produz: modelo de tipos do negócio (interno)
- Ênfase em descoberta: que informação precisa ser gerida? que interfaces são necessárias para geri-las? que componentes são necessários para para oferecer essas funcionalidades e como elas vão se integrar?.

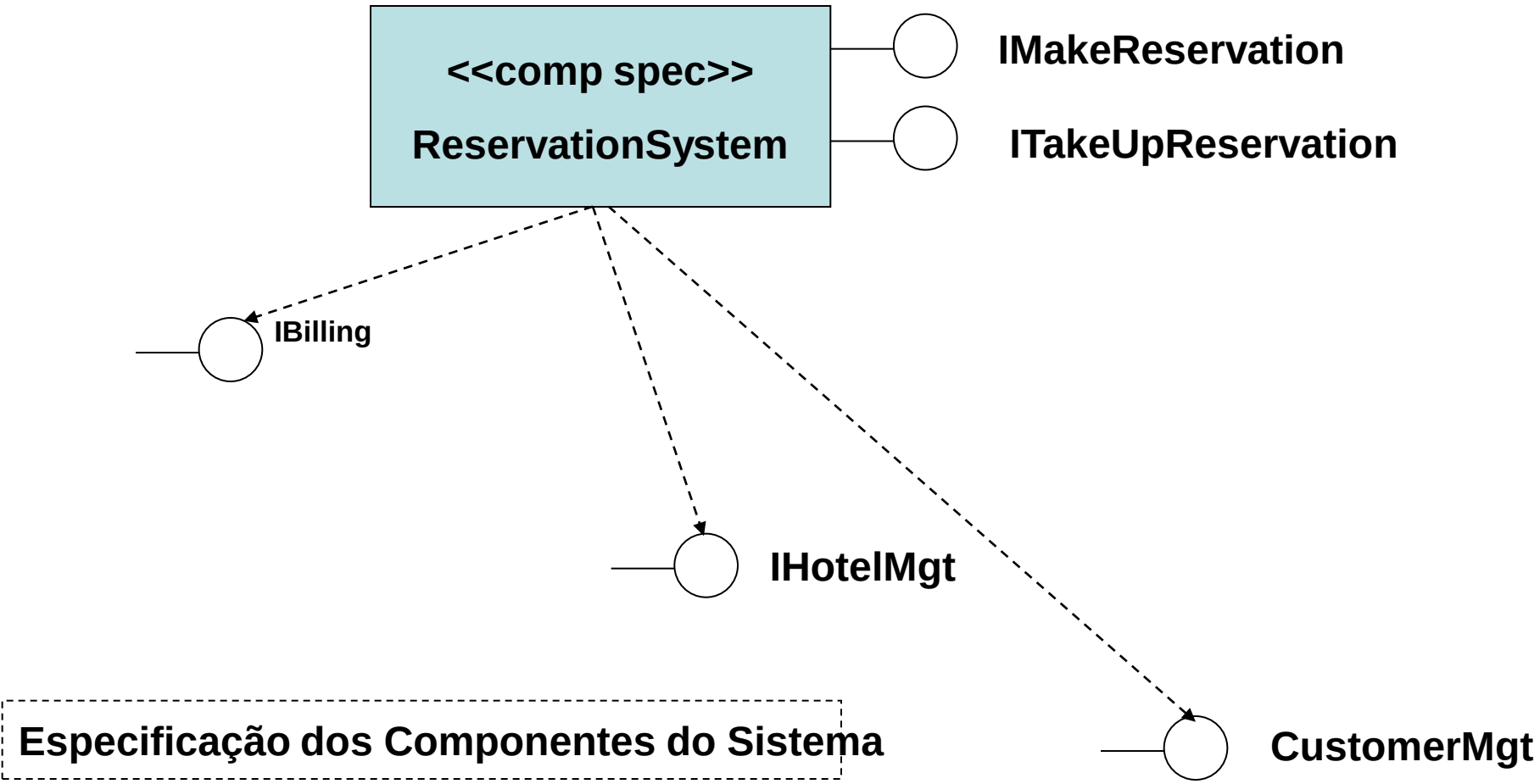

Modelo de Conceitos do Negócio
Diagrama de Casos de Uso
Identificação de Componentes
Desenvolver o Modelo de Tipos de Negócio
Interfaces Existentes
Identificar as Interfaces do Negócio
Identificar Interfaces do Sistema e Operações
Padrões Arquiteturais
Recursos Existentes
Criar a Especificação dos Componentes & Arquitetura Iniciais
Modelo de Tipos de Negócio
Interfaces de Negócio
Especificação de Componentes & Arquitetura
Interfaces de Sistema

Identificação das Interfaces

- Preocupação principal com o sistema de negócio (*business system*), que são os aspectos de uma aplicação independentes da interface com o usuário (seria o lado do servidor).
- A arquitetura da aplicação é vista como tendo três camadas: Tipos de Diálogos, Interface do Sistema e Interfaces do Negócio.
- Os diálogos implementam a lógica dos casos de uso, que são divididos em passos. Estes são usados para identificar as operações do sistema necessárias para atender às suas responsabilidades.

Especificações dos Componentes do Sistema

- No caso de estudo, as especificações derivadas dos casos de uso são fortemente sobrepostas e gerem conceitos que têm o mesmo tempo de vida.
- Elas poderiam ser apoiadas por uma única especificação de componente. Mas a conexão do sistema de faturamento não faz sentido para as outras especificações

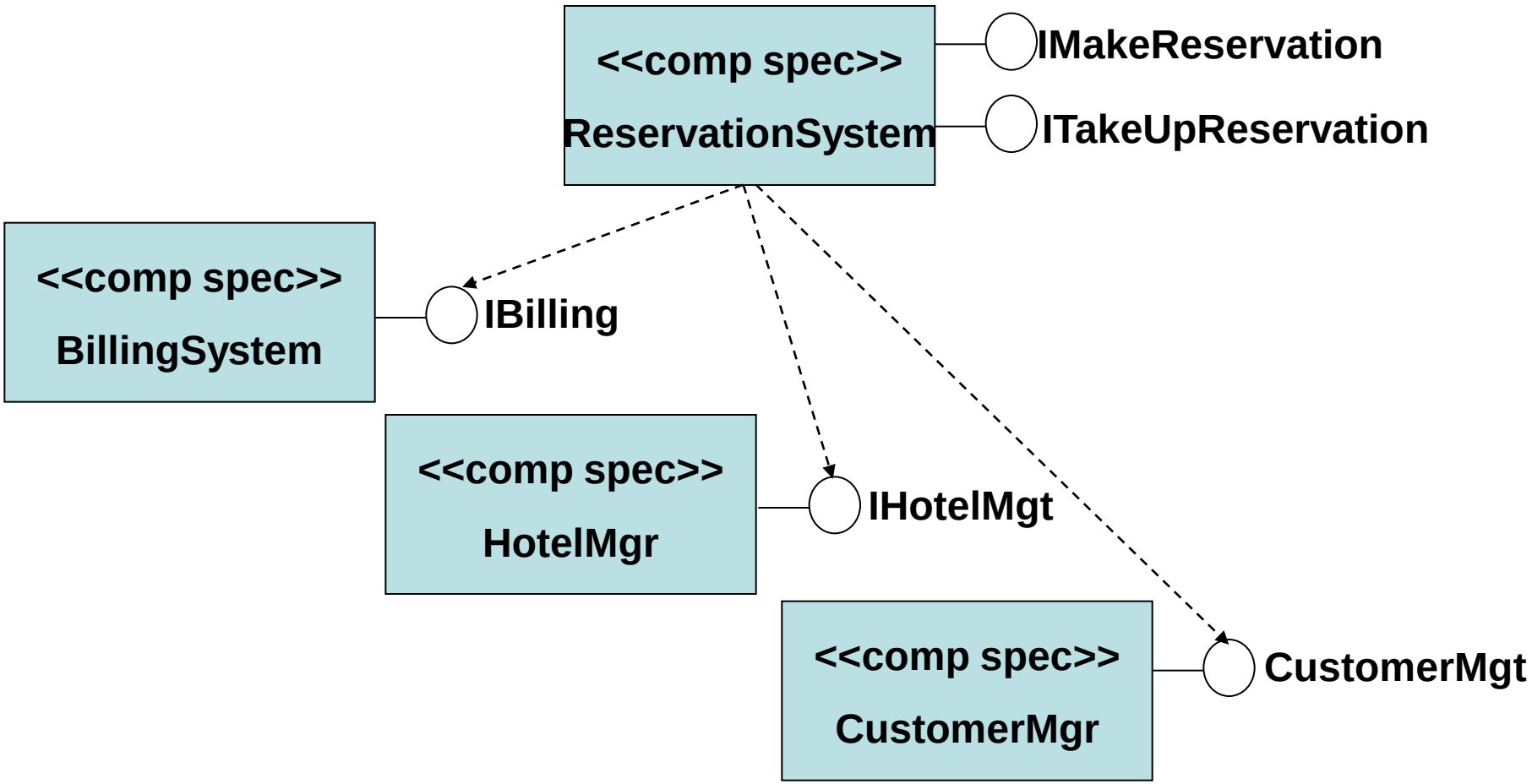


Especificação dos Componentes do Sistema

Especificações dos Componentes de Negócio

- O ponto de partida para as interfaces de negócio é um especificação de componente por interface.
- Como as interfaces de gerenciamento foram criadas para gerir instâncias de tipos de negócio básicos, que naturalmente são independentes, então eles levam a especificação de componentes separadas.

Uma Arquitetura Inicial



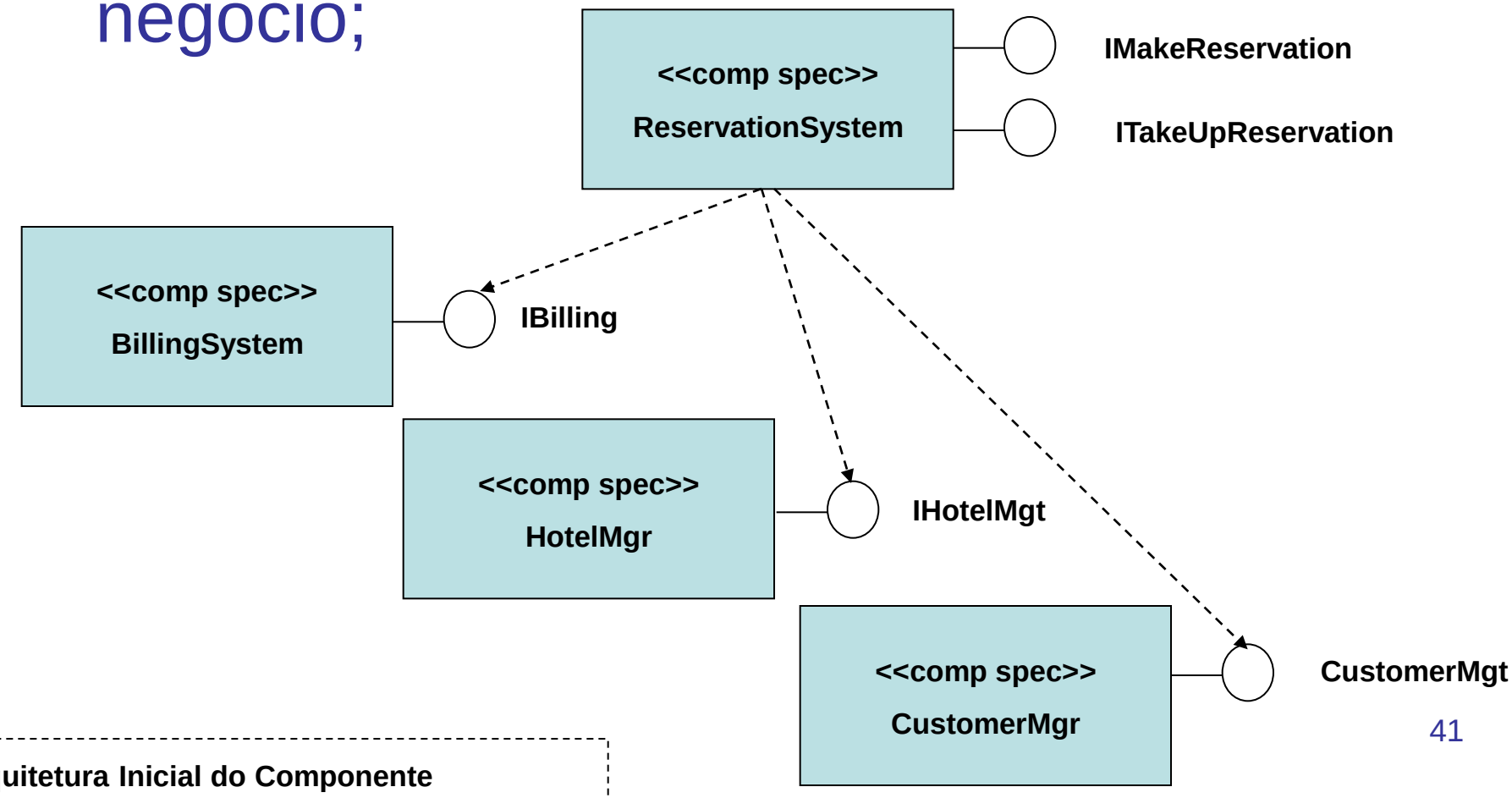
Especificação da Arquitetura de Componentes

Interações dos Componentes

- A fase anterior forneceu o conjunto inicial dos componentes e interfaces.
- Mostra como os componentes trabalham juntos para oferecer as funcionalidades.
- A modelagem de interações é uma técnica genérica para modelagem de comportamento

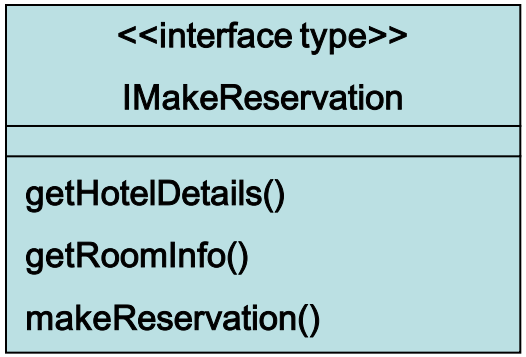
Descobrir Operações de Negócio

- O foco é descobrir as operações de negócio;



Descobrir Operações de Negócio
(cont)

- O foco é descobrir as operações de negócio;

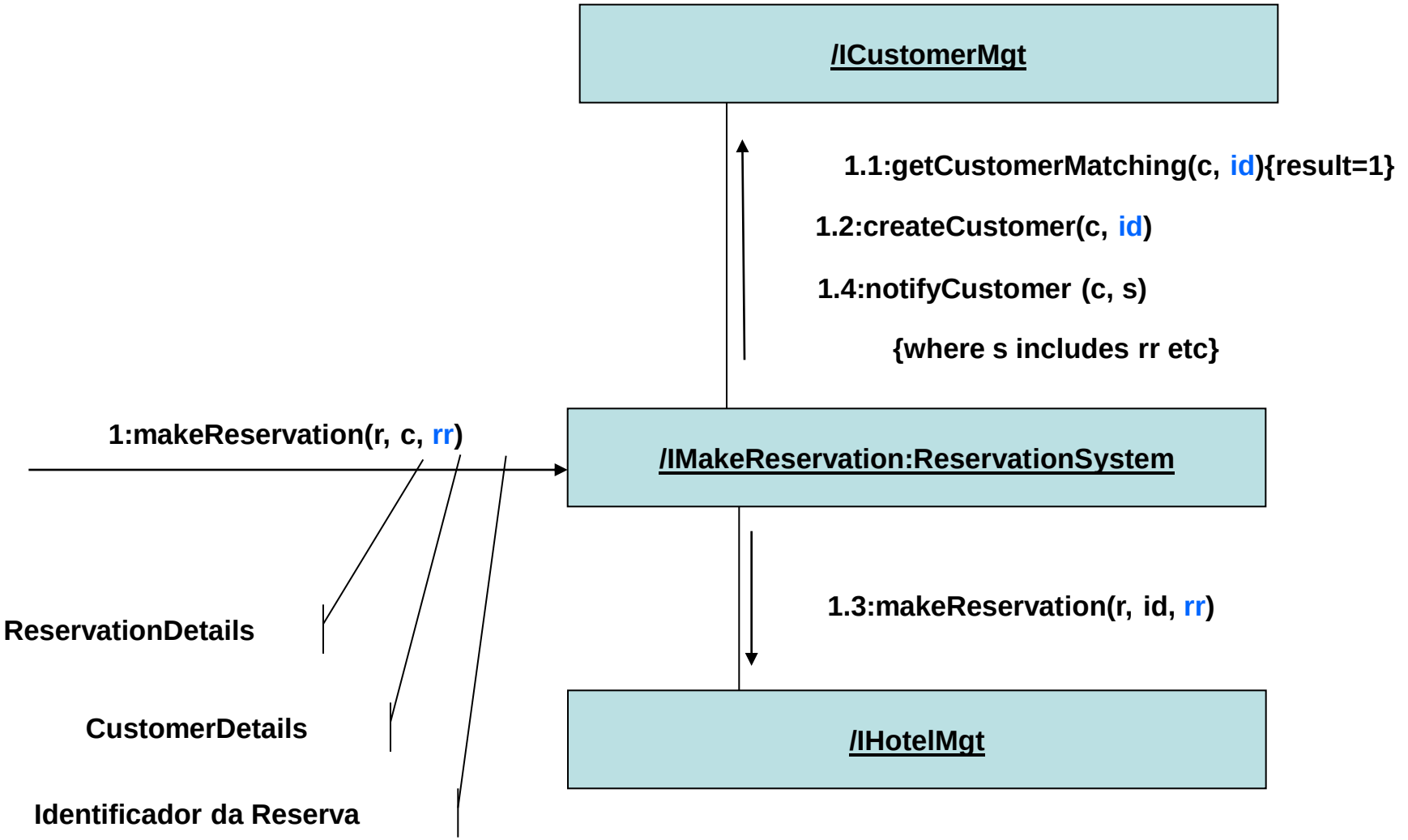


Conjunto inicial de operações das interfaces

Descobrir Operações de Negócio (cont.)

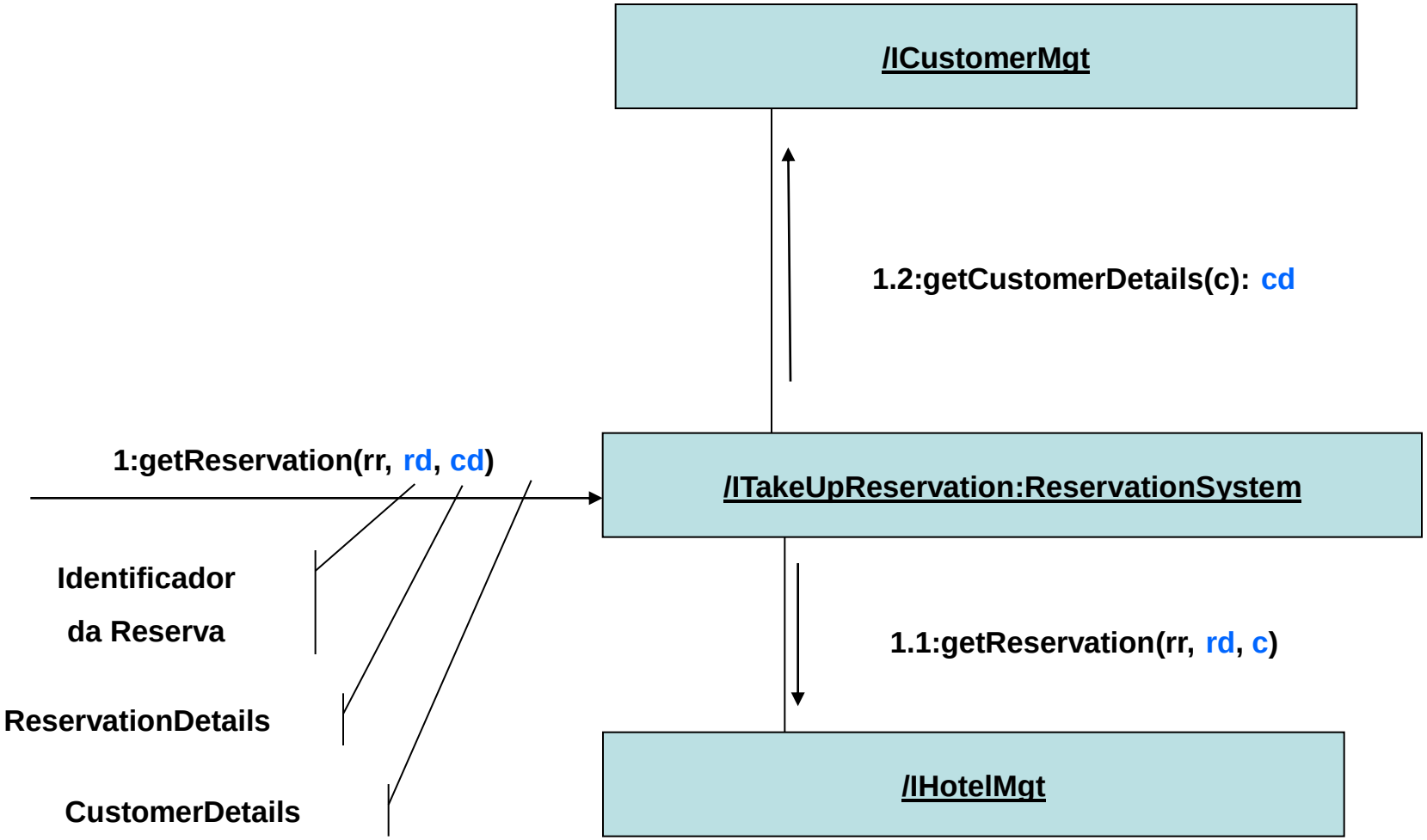
- Diagramas de colaboração são usados para descobrir e modelar cada possível fluxo de execução;
- Todos os fluxos alternativos importantes devem ser modelados;
- Deve-se encontrar e estabelecer as restrições ao fluxo de execução decorrente das chamadas as operações;

Completando a operação makeReservation



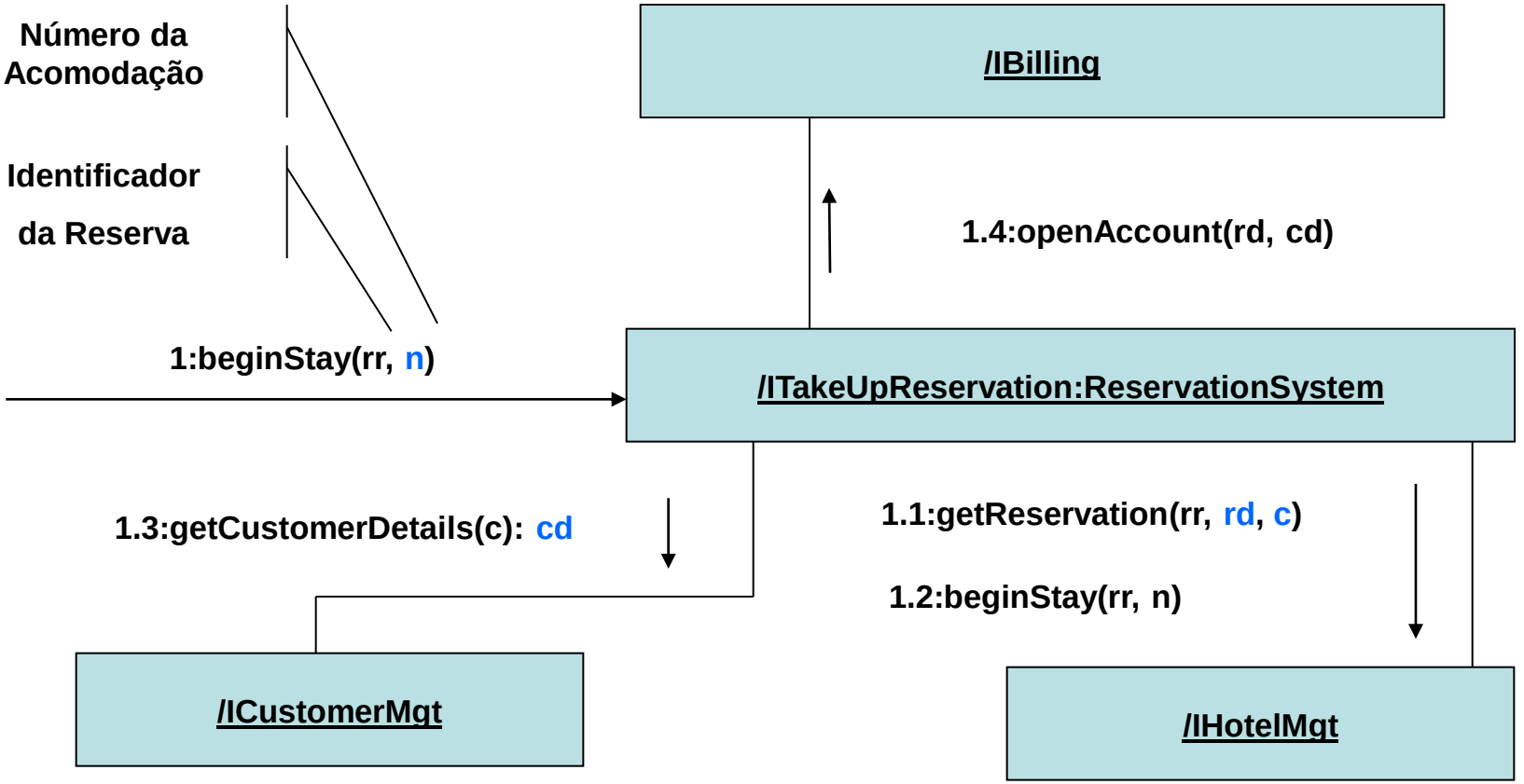
Interações de *makeReservation()* (novo cliente)

Demais Operações



Interações de *getReservation()*

Demais Operações



Interações de *beginStay()*

Interfaces do Sistema

<<interface type>> IMakeReservation
getHotelDetails(in match: string): Hoteldetails[] getRoomInfo(in res:ReservationDetails, out availability:Boolean, out price:Currency) makeReservation(in res:ReservationDetails, in cus:customerDetails, out resRef:String):Integer

<<interface type>> ITakeUpReservation
getReservation(in resRef:String, out rd:ReservationDetails, out cus:CustomerDetails):Boolean beginStay(in resRef:String, out roomNumber:String):Boolean

<<interface type>> IBilling
openAccount(in res:ReservationDetails, in cus:CustomerDetails)

Interfaces do Sistema e as assinaturas de suas operações

Interfaces de Negócio

<<interface type>> IHotelMgt
getHotelDetails(in match: string): Hoteldetails[]
getRoomInfo(in res:ReservationDetails, out availability:Boolean, out price:Currency)
makeReservation(in res:ReservationDetails, in cus:customerDetails, out resRef:String):Integer
getReservation(in resRef:String, out rd:ReservationDetails, out cusId:CustId):Boolean
beginStay(resRef:String, out roomNumber:string:Boolean

<<interface type>> ICustomerMgt
getCustomerMatching(in custD:CustomerDetails, out cusId:CustId):Integer
createCustomer(in custD: CustomerDetails, out cusId:CustId):Boolean
getCustomerDetails(in cus:CustId):CustomerDetails
notifyCustomer(in cus:custId, in msg:String)

Interfaces de Negócio e as assinaturas de suas operações

Refinar Interfaces

- Após a descoberta e definições de operações é necessário refinar as operações e interfaces;
- Pode-se fatorar as interfaces com generalizações ou divisão em mais interfaces;
- As operações também podem ser fatoradas para se tornar mais genéricas;

Refinar Interfaces (cont)

- Não se deve antecipar funcionalidades e requisitos;
- Muita da flexibilidade dos sistemas baseados em componentes advém do fato de que componentes podem oferecer múltiplas interfaces;
- Novos requisitos podem ser acomodados usando-se novas interfaces e gerenciando a dependência;

Especificação de Componentes

- O objetivo é especificar os contratos de uso e os contratos de realização;
- O contrato de uso é definido pela especificação de interface;
- O contrato de realização é definido pela especificação de componente;

Especificação de Interfaces

- Uma interface é um conjunto de operações que definem, cada uma, serviços e funções do objeto componente;
- As interfaces definem como gerenciar as dependências e, portanto, são unidades de contrato dos componentes;
- A primeira tarefa é descrever o estado dos objetos componente;

Modelos de Informação da Interface

- Cada interface deve possuir um modelo de informação que especifica as operações das interfaces;
- É um modelo dos possíveis estados de um objeto componente;
- As mudanças de estado dos objetos componente, causados pelas operações, podem ser descritas por esse modelo;

Modelos de Informação da Interface: Especificação das Operações

- Deve mostrar:
  - os parâmetros de entrada;
  - os parâmetros de saída;
  - os resultados da mudança de estado dos objetos componentes;
  - As restrições aplicáveis.

Exemplo de Especificação de Interface

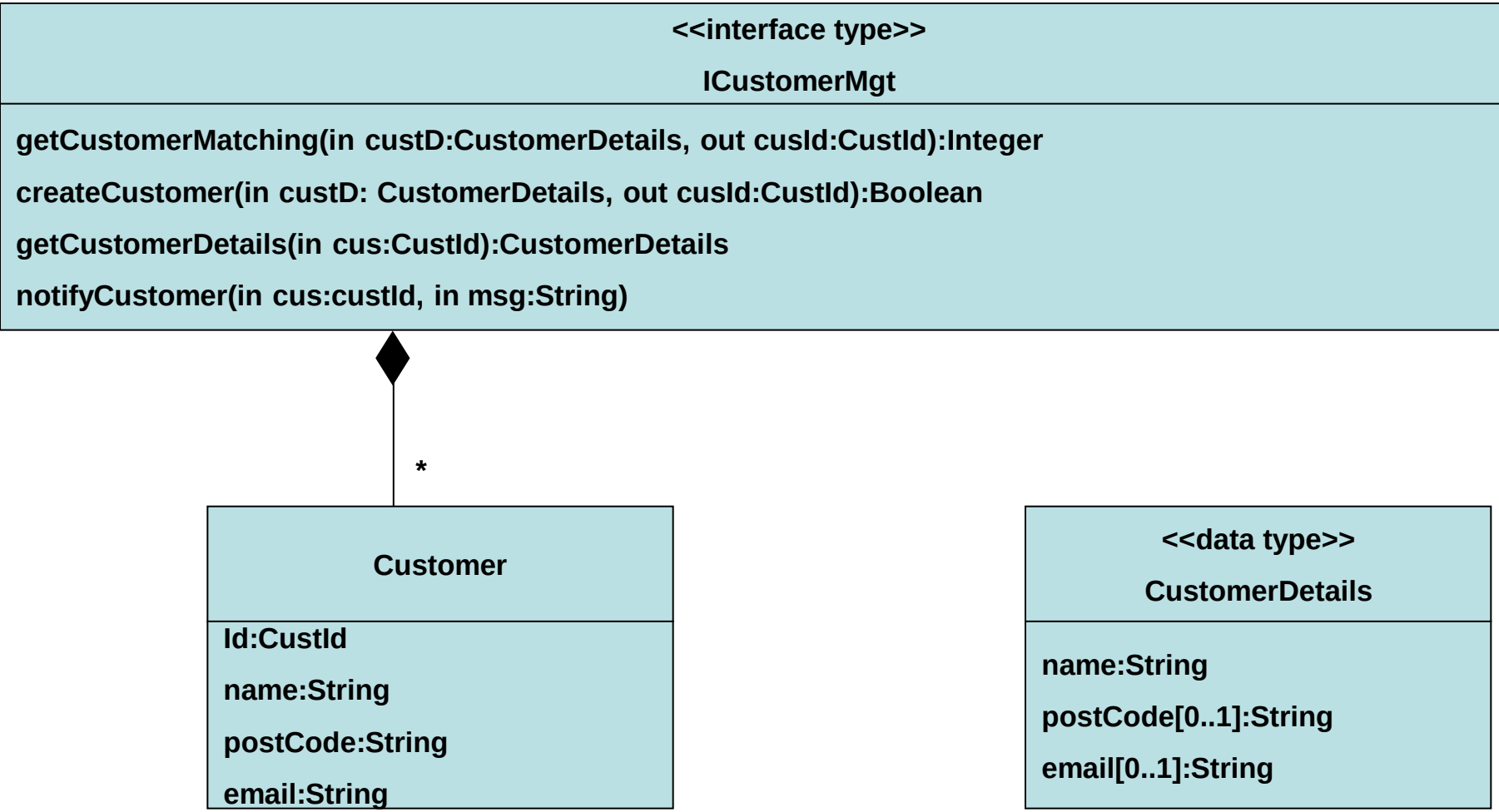


Diagrama de especificação de interface para ICustomerMgt

Pré-condições e Pós-condições

- Cada operação tem uma pré-condição e uma pós-condição;
- A pós-condição especifica o efeito da operação se a pré-condição for verdadeira;
- A pré-condição não é uma condição para que uma operação seja invocada;
- A pré-condição é uma condição de garantia para que a operação garanta que a pós-condição seja verdadeira;

Pré-condições e Pós-condições

(cont.)

- OCL é usada para especificar as pré e pós-condições;
- Uma operação para alterar o nome do cliente:

```
context ICustomerMgt::
        changeCustomerName(in cus:CustId,newName:String)
pre:
     -- Cus é um identificador de cliente valido
     customer->exists(c | c.id = cus)

post:
     -- o nome do cliente de identificar cus é newName
       customer->exists(c | c.id = cus and c.name = newName)
```

```
pre:
  -- hotel e tipo de acomodações válidos
  hotel->exists(h | h.id = res.hotel and
              h.room.roomType.name->includes(res.roomType))


post:
  -- retornar verdadeiro para sucesso da operação
  result implies
    -- a reserva foi criada
    -- identificar o hotel
    Let h = hotel->select(x |
                          x.id=res.hotel)->asSequence->first in
        -- deve haver uma reserva a mais que anteriormente
        (h.reservation - h.reservation@pre)->size=1 and
        -- identificar a reserva
        Let r=(h.reservation-h.reservation@pre)->asSequence->first in
        -- a ref retornada é o da nova reserva
        r.resRef=resRef and
```

Especificação dos Componentes

- Para cada especificação de componentes é necessário estabelecer a quais interfaces sua realização dá suporte;
- Um diagrama de especificação de componentes mostra a dependência de interfaces de objeto componente as interfaces oferecidas;

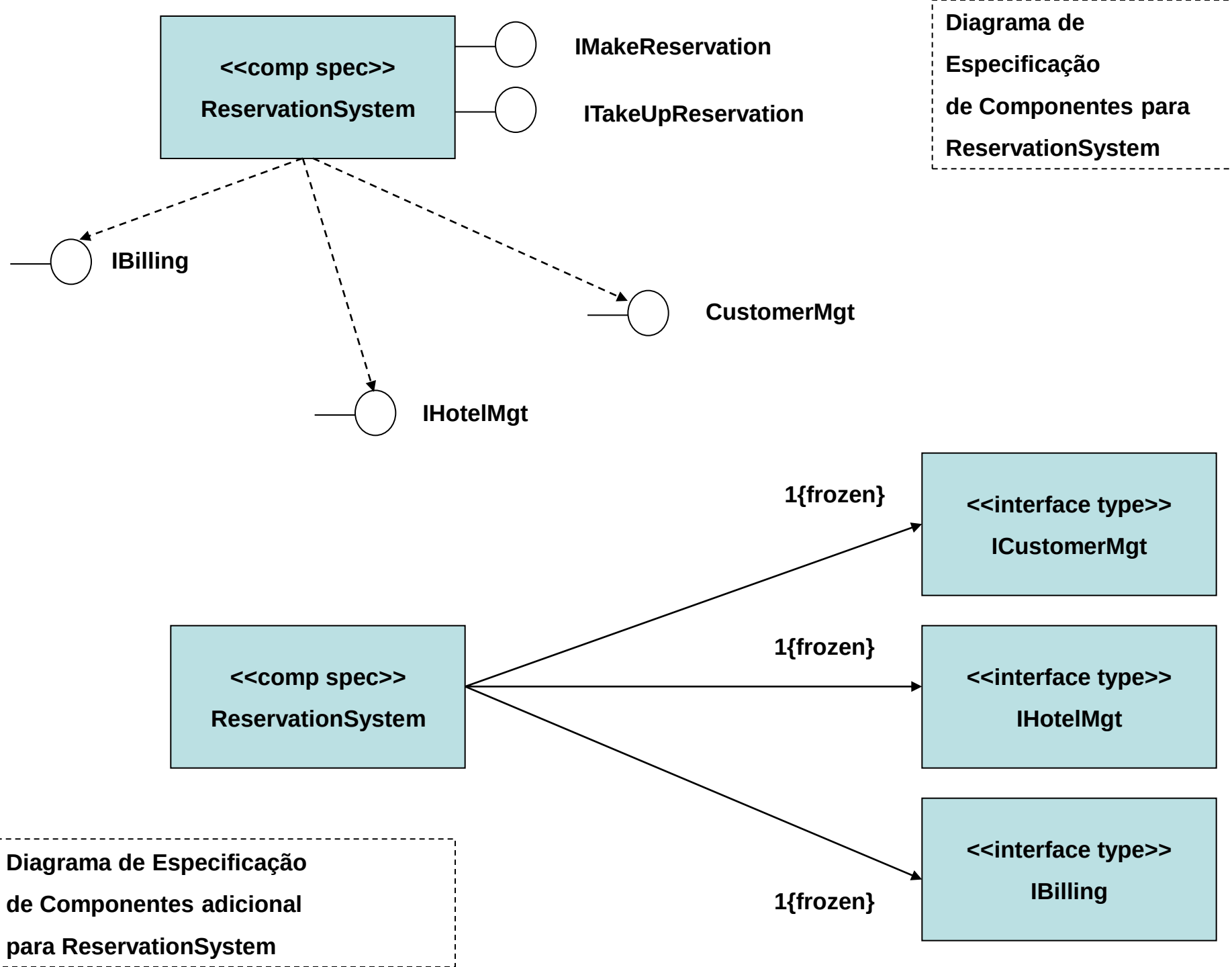


Diagrama de
Especificação
de Componentes para
ReservationSystem
<<comp spec>>
ReservationSystem
IMakeReservation
ITakeUpReservation
IBilling
CustomerMgt
IHotelMgt
1{frozen}
<<interface type>>
ICustomerMgt
<<comp spec>>
ReservationSystem
1{frozen}
<<interface type>>
IHotelMgt
1{frozen}
<<interface type>>
IBilling
Diagrama de Especificação
de Componentes adicional
para ReservationSystem