(Engenharia de) Requisitos

Cheesman

Cp 4

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.

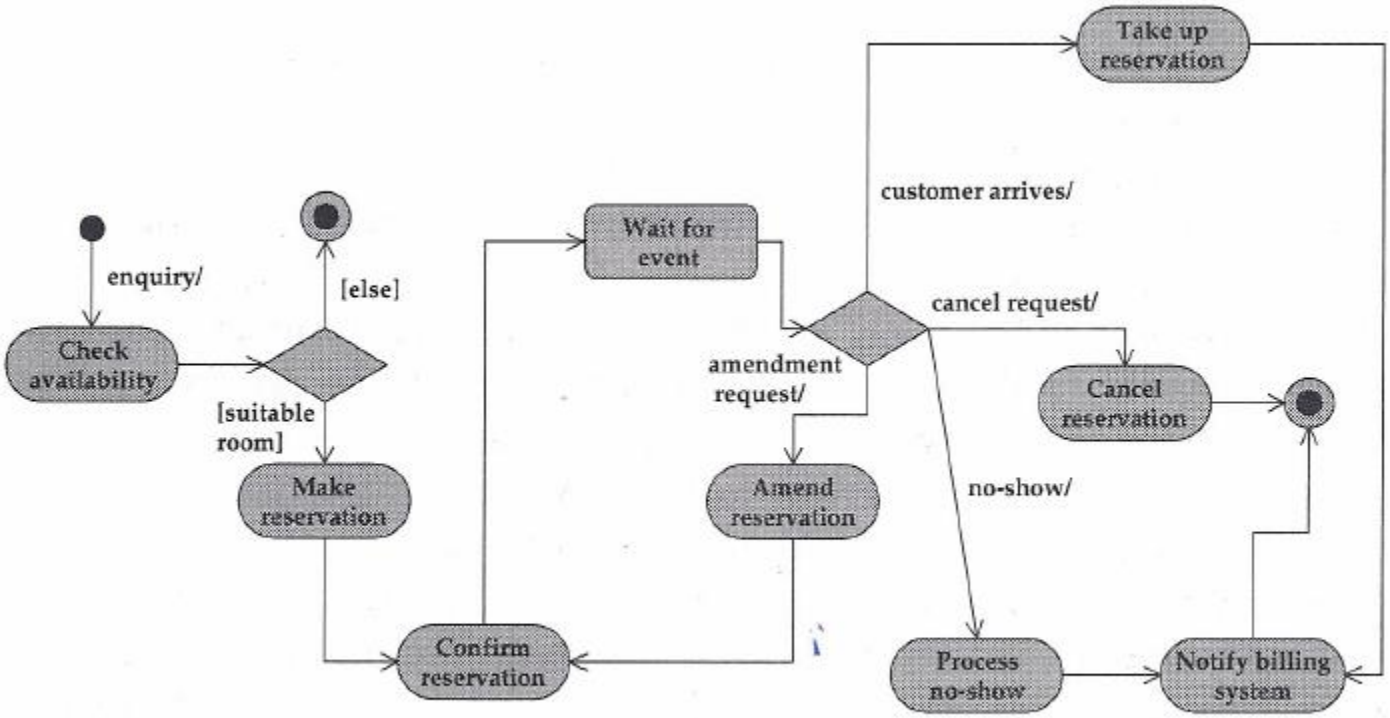


Figure 4.1 Business process for hotel reservation

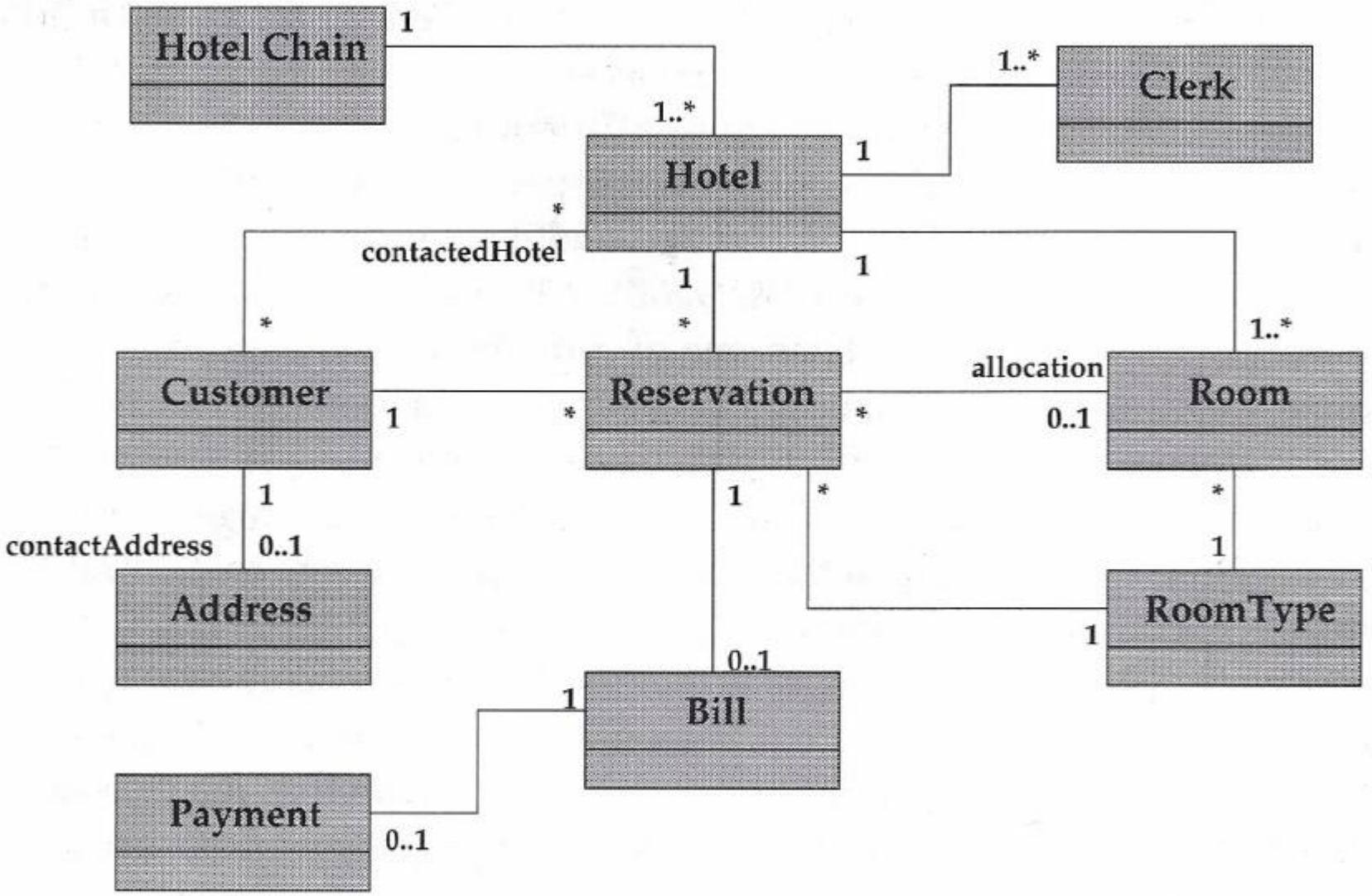
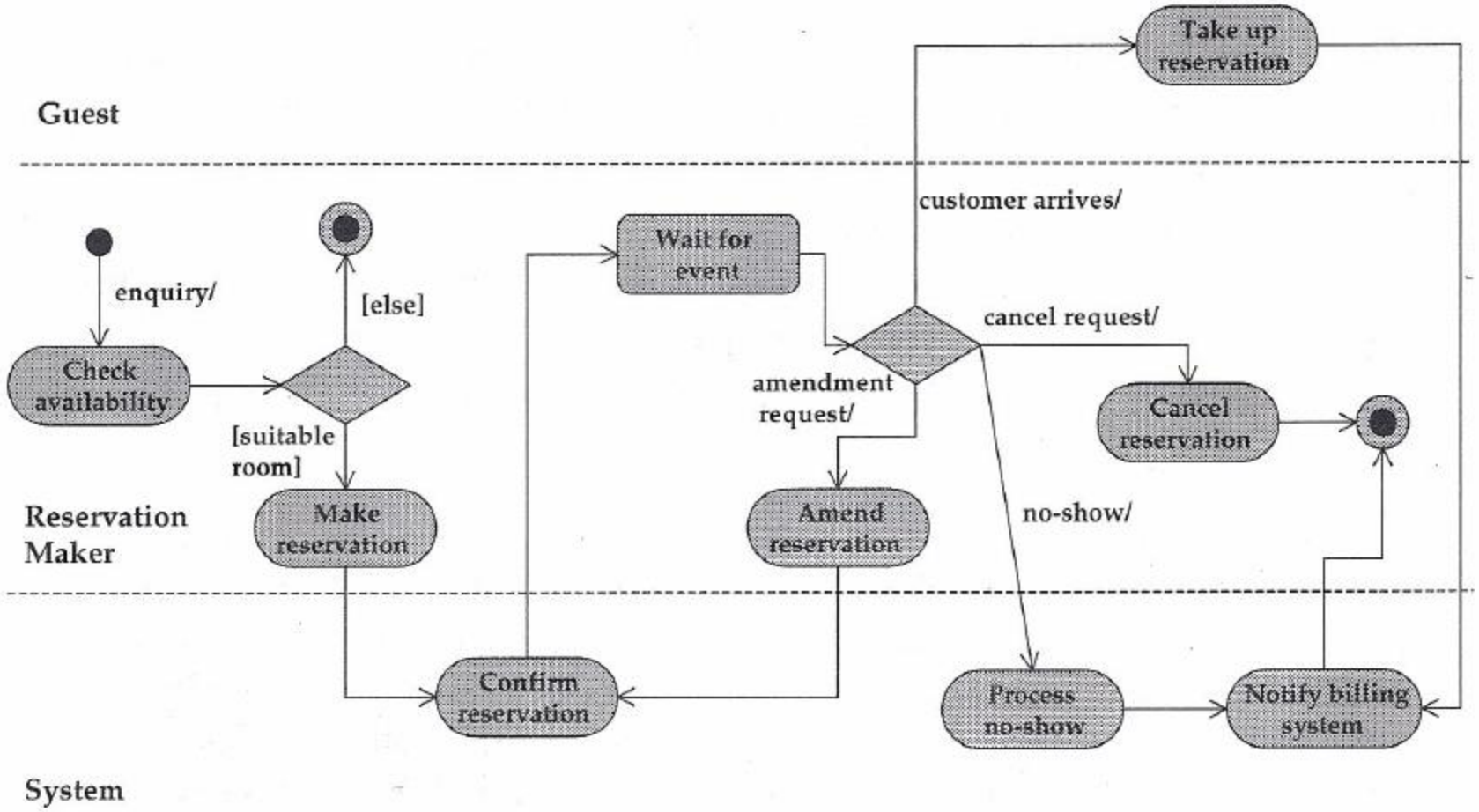
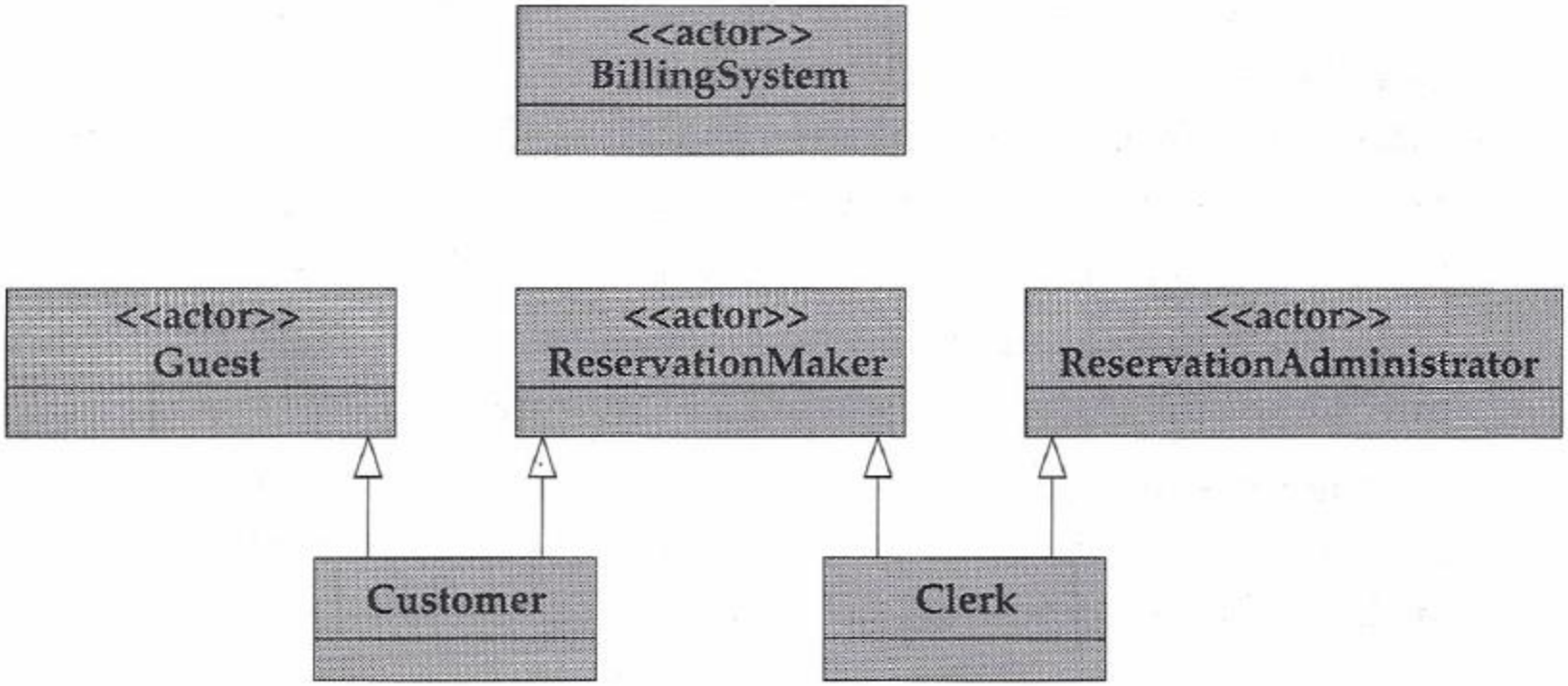
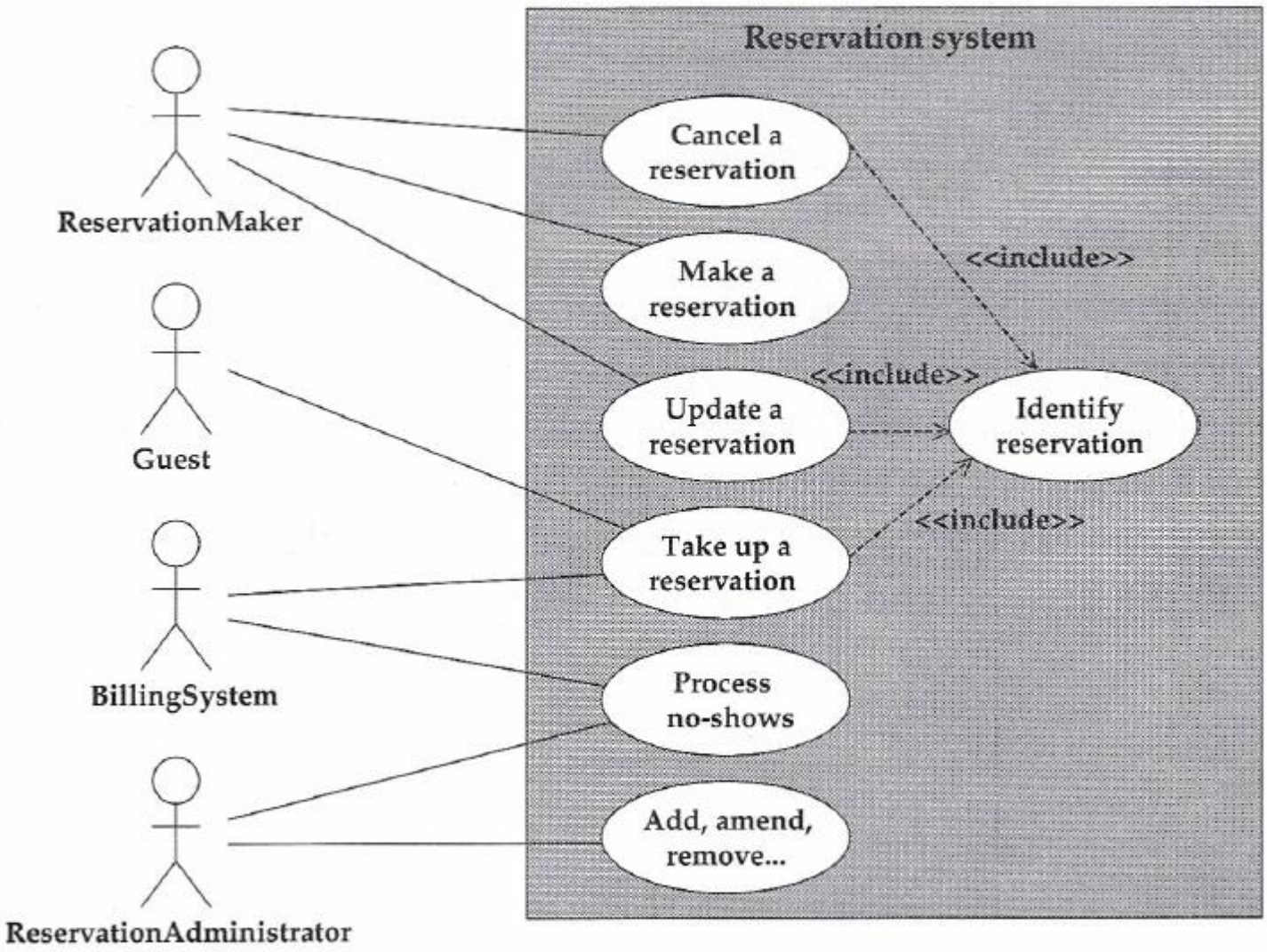


Figure 4.2 Business concept model for hotel reservation


Guest
Take up reservation
customer arrives/
Wait for event
enquiry/
[else]
Check availability
cancel request/
amendment request/
Cancel reservation
[suitable room]
Reservation Maker
Make reservation
Amend reservation
no-show/
Confirm reservation
Process no-show
Notify billing system
System


<<actor>>
BillingSystem
<<actor>>
Guest
<<actor>>
ReservationMaker
<<actor>>
ReservationAdministrator
Customer
Clerk


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	**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

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

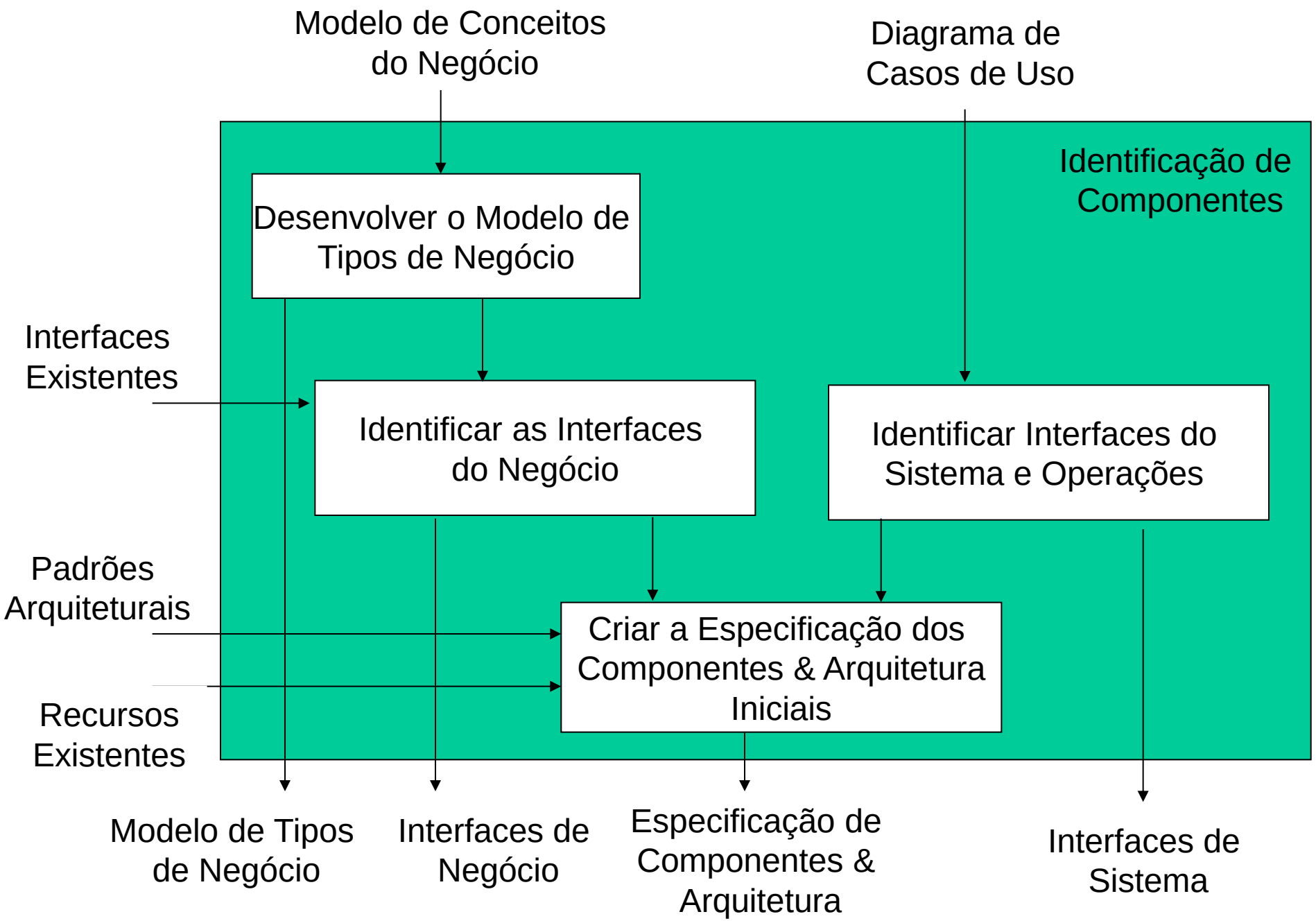
2b. No active reservations at this hotel for this customer.
   a. Fail

Identificação de Componentes

Cap. 5

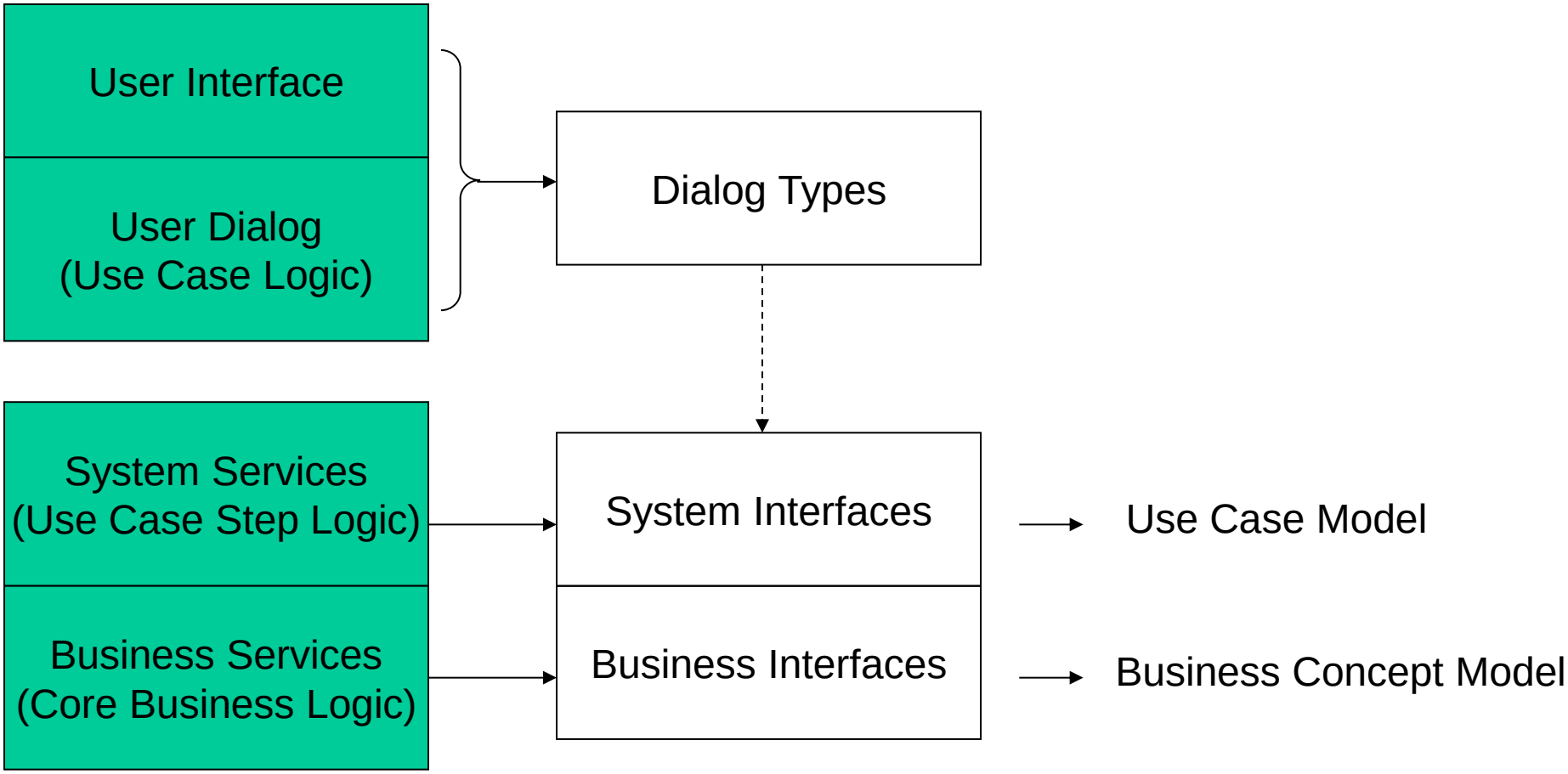
Introdução

- 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
Criar a Especificação dos Componentes & Arquitetura Iniciais
Recursos Existentes
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.


User Interface
User Dialog
(Use Case Logic)
Dialog Types
System Services
(Use Case Step Logic)
Business Services
(Core Business Logic)
System Interfaces
Business Interfaces
Use Case Model
Business Concept Model

Identificação das Interfaces (cont.)

- Como funciona durante a execução:
  - quando o usuário inicia o caso de uso, sua lógica faz com que a IU apropriada seja criada e mostrada;
  - o usuário é guiado pelos passos de acordo com a lógica do caso de uso;
  - Sempre que a lógica do caso de uso necessita mostrar alguma informação ou notificar o sistema, ela chama uma operação apropriada e esta usa operações definidas na lógica de negócios interna (*core*) para executar essa função

Identificação das Operações e das Interfaces do Sistema

- Como regra geral, são definidos um tipo de diálogo e uma interface de sistema por caso de uso.
- Examinar cada caso de uso e, para cada passo, identificar se há responsabilidades do sistema que precisam ser modeladas.
- Em caso positivo, representá-las como uma ou mais operações apropriadas da interface do sistema.
- Vários passos consecutivos do caso de uso que são de responsabilidades do sistema **podem** ser colapsados em uma única operação

Exemplo: Make up Reservation (Fazer reserva)

- Cria-se uma interface inicial do sistema chamada IMakeReservation.
- No cenário principal de sucesso, passo 2, o sistema permite que o funcionário obtenha detalhes de diferentes hotéis e depois, para uma dada seleção (no passo 3) mostra a disponibilidade e o preço para um dado período →getHotelDetails() e getRoomInfo().
- O passo 7 permite inferir a necessidade de makeReservation()

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.

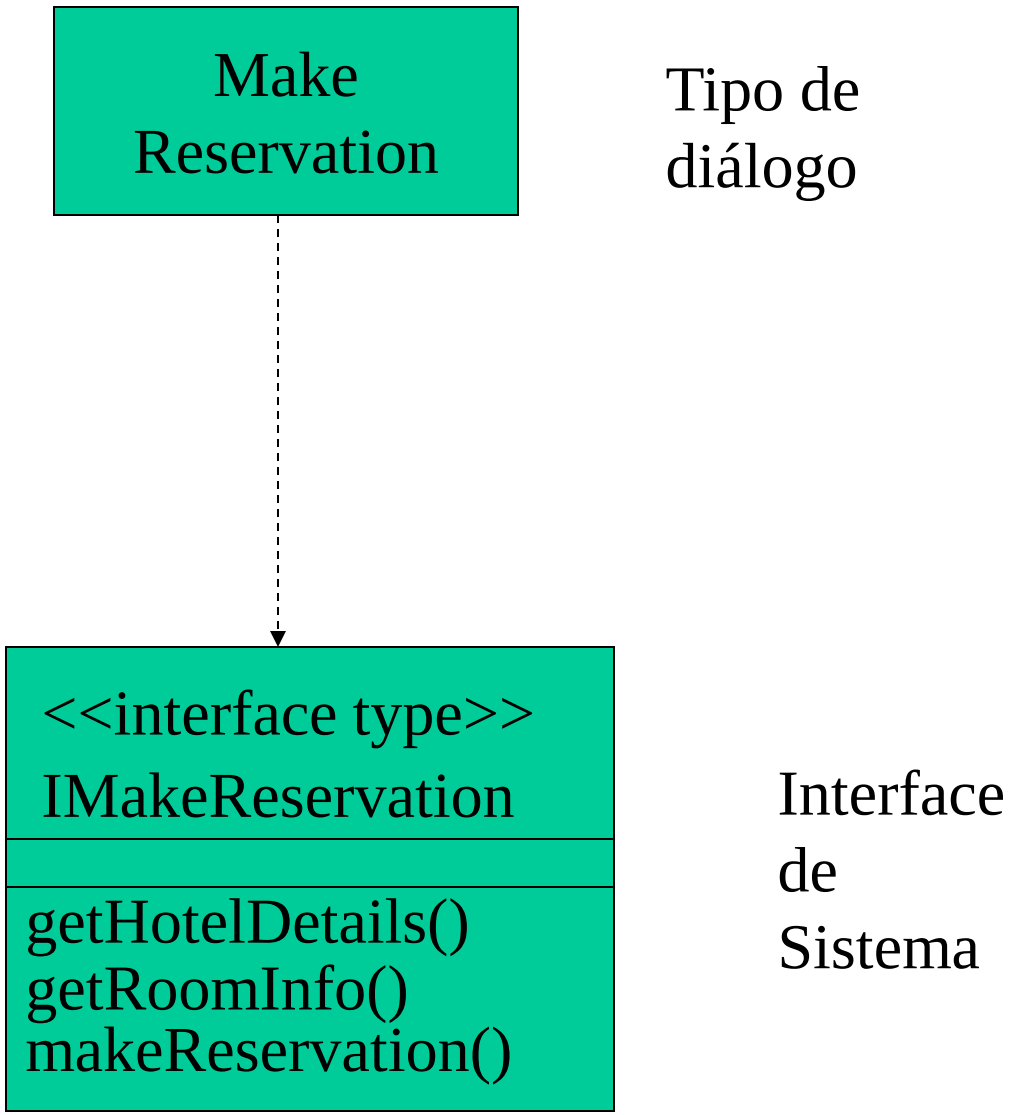
Extensions

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.


Make Reservation
Tipo de diálogo
<<interface type>>
IMakeReservation
getHotelDetails()
getRoomInfo()
makeReservation()
Interface de Sistema

Make up reservation (cont.)

- As extensões descrevem comportamentos alternativos sob certas condições.
- A extensão “acomodação não disponível” leva o usuário a selecionar datas e tipos de acomodações alternativas.
- Mas isso não implica em novas operações do sistema: a projeção (na tela) e a seleção de informação serão tratadas pela lógica do diálogo com o usuário.

Exemplo:Take up Reservation (Ocupar Reserva)

- Neste caso de uso, o hóspede chega e registra-se no hotel. Ele fornece um número (código) de reserva e o sistema recupera a reserva (passo 3) → getReservation()
- Os detalhes da reserva são confirmados com o hóspede. Para iniciar a estada, o sistema aloca um quarto (apto) e notifica o sistema de faturamento que a estada iniciou →beginStay()

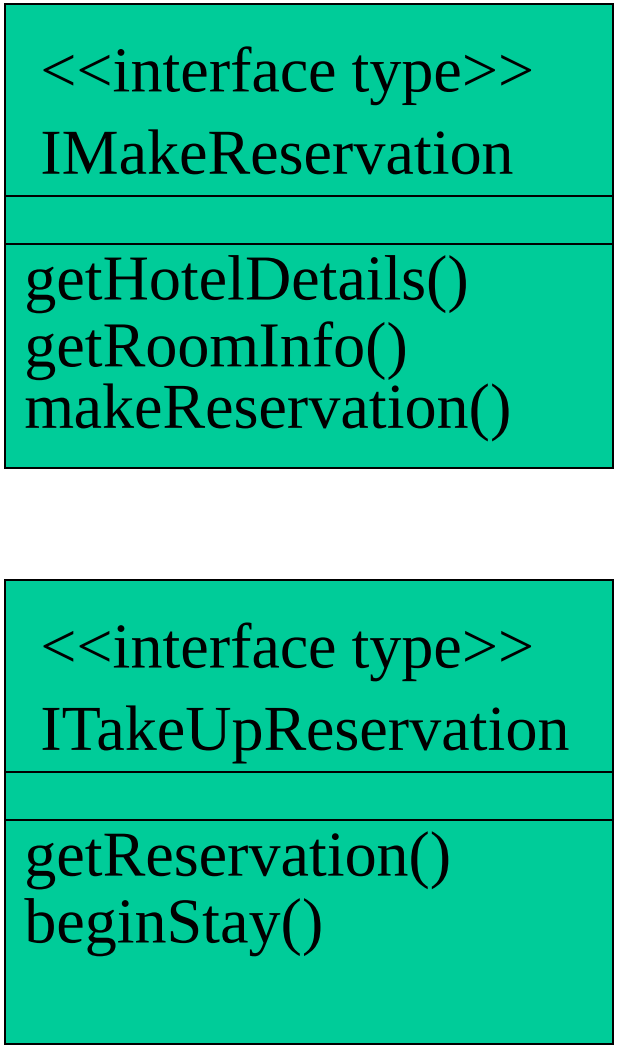
Name	**Take up reservation**
Initiator	**Guest**
Goal	**Claim a reservation and check in to the hotel**

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


<<interface type>>
IMakeReservation
getHotelDetails()
getRoomInfo()
makeReservation()
<<interface type>>
ITakeUpReservation
getReservation()
beginStay()

Take up Reservation (cont.)

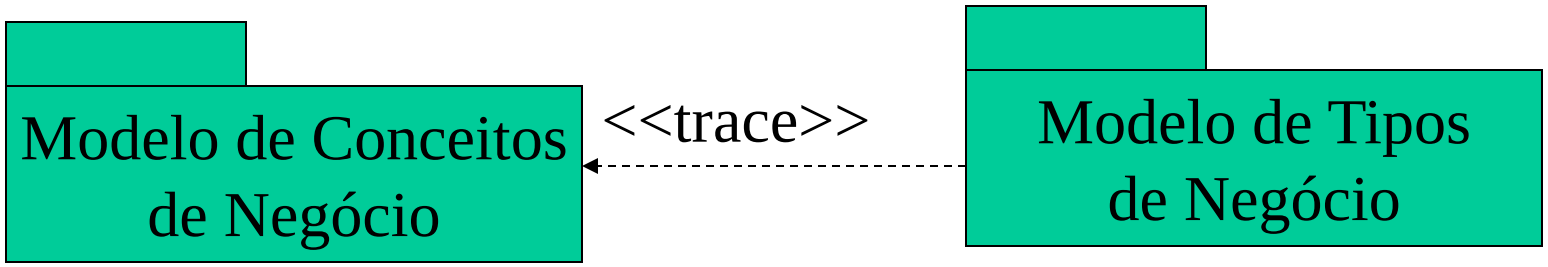
- Os parâmetros das operações ainda não foram definidos → isso será feito quando as interações entre objetos for analisada.
- As interfaces definidas nesta atividade são específicas do sistema e não naturalmente reusáveis por sistemas diferentes.
- O reuso de interface entre diferentes sistemas é o propósito das interfaces de negócios, que devem ter o objetivo de serem independentes de sistema.

Identificar as Interfaces de Negócio

1. Produzir uma cópia delimitada do modelo de conceitos de negócio como o modelo de tipos de negócio.
2. Refinar o modelo de tipos de negócio e especificar todas as regras de negócio adicionais.
3. Identificar os tipos de negócio básicos
4. Criar interfaces de negócio para os t.n.b. e adicioná-las ao m.t.n.
5. Refinar o m.t.n. para indicar as responsabilidades das interfaces de negócio

Criar o Modelo de Tipos de Negócio

- O modelo de tipos de negócio contém as informações específicas do negócio que devem ser mantidas pelo sistema especificado.
- Define os dados e estados que a empresa precisa manter e monitorar.
- Tipos de negócios podem ser físicos ou abstratos



Refinar o Modelo de Tipos de Negócio

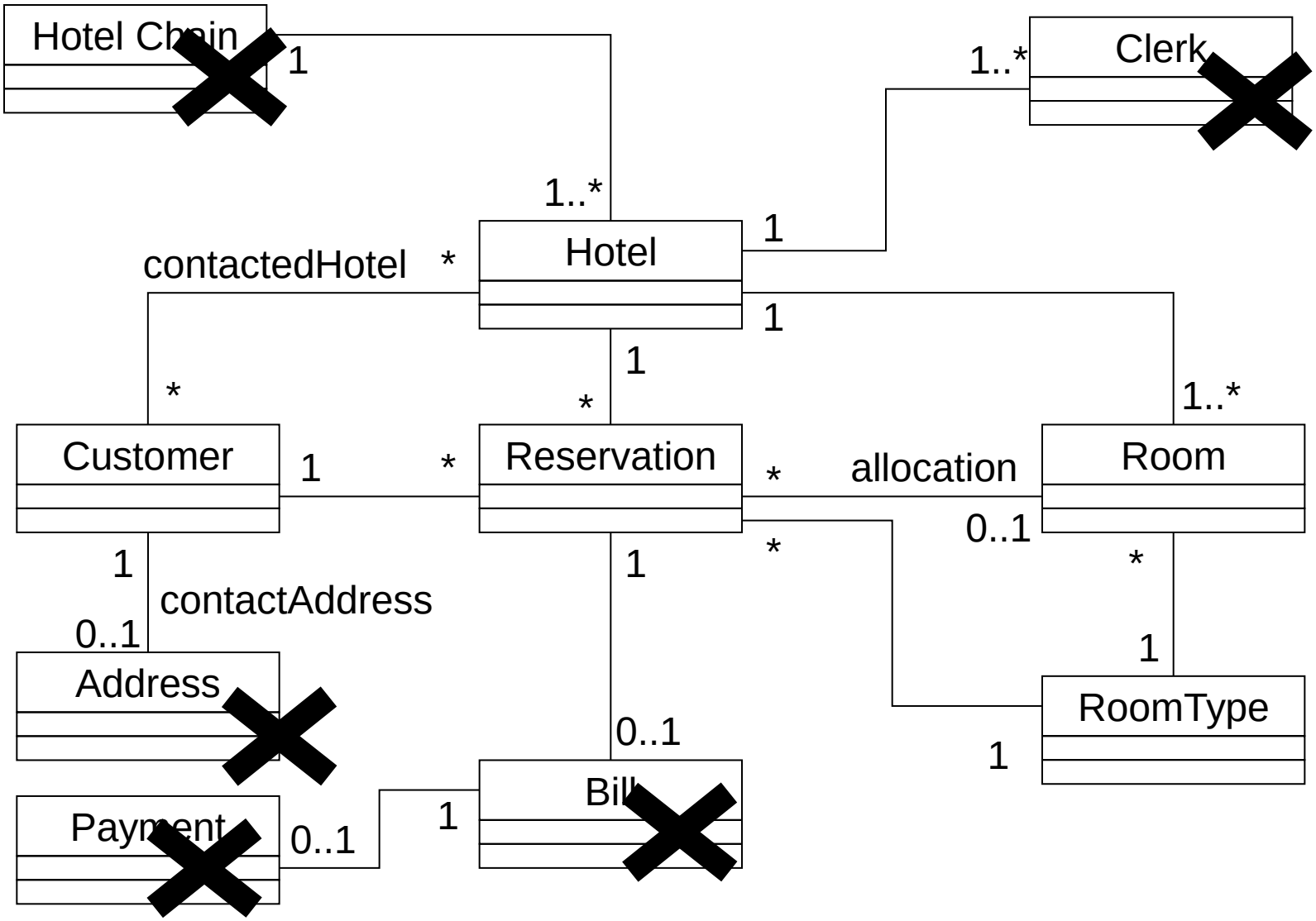
- Parta do modelo de conceitos de negócio e adicione ou remova elementos até que a abrangência (escopo) do modelo se torne correta.

Análise Criar/Destruir

- HotelChain – o requisito é uma sistema para uma única cadeia e, portanto, cadeias nunca são criadas ou destruídas.
- Hotel – podem ser adicionados ou removidos, mesmo que com baixa frequência, portanto são precisos casos de uso para esses eventos.
- Room – pode ser adicionados ou removidos...
- RoomType – pode ser adicionado ou removido...
- Clerk – eles vêm e vão ...(assumindo que o sistema precisa tomar conhecimento de clerks – e nós assumimos)

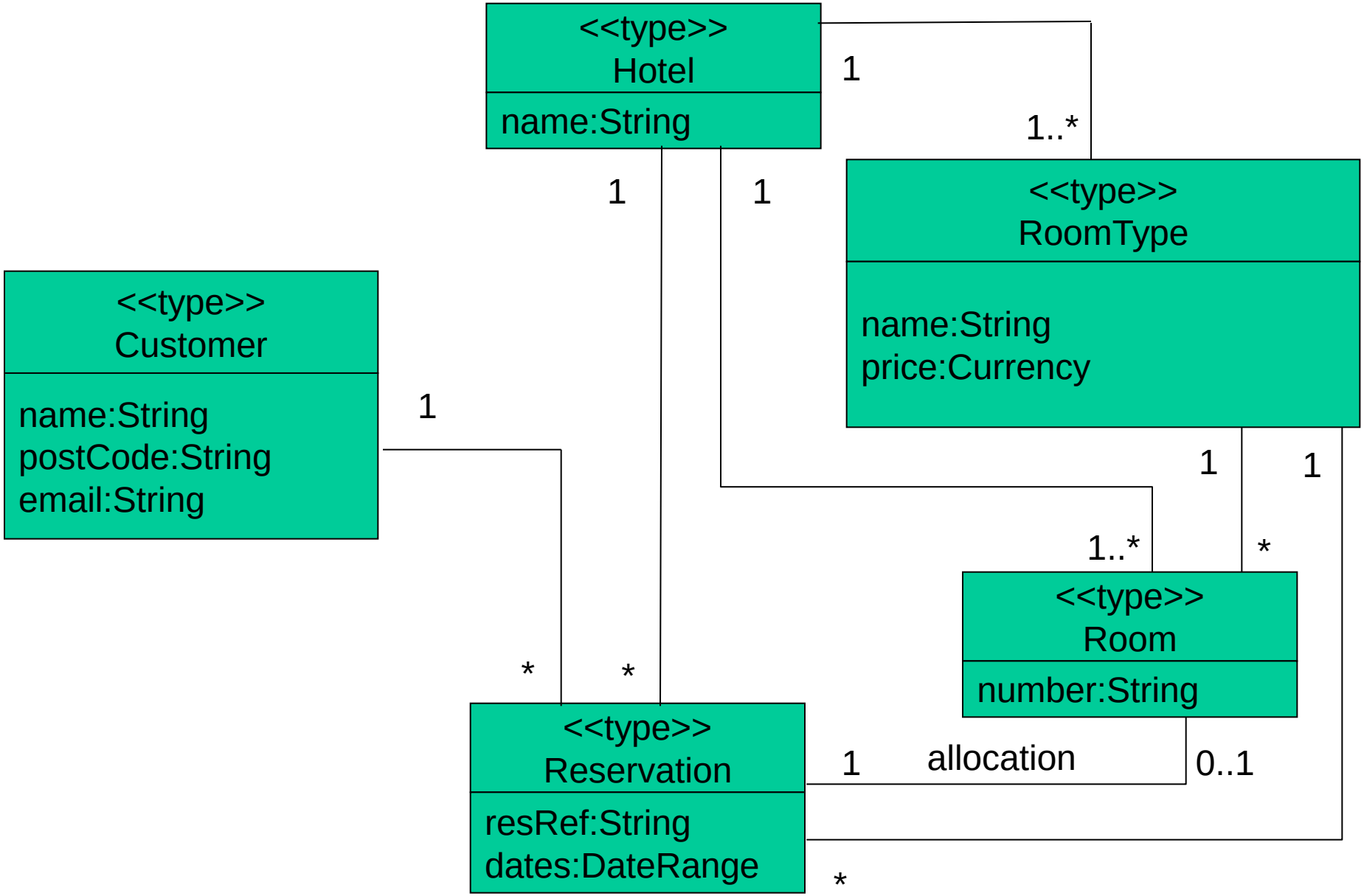
Checagem das associações

- HotelChain-Hotel – Nunca muda
- Hotel-Room – Nunca muda (não se pode mudar um quarto de um hotel para outro)
- Hotel-Clerk – Podem ser movidos de um hotel para outros, mas decidimos que o sistema não dará suporte a essa função.
- Hotel-Customer – Não será mantida pelo sistema.
- Hotel-reservation – Pode ser mudada como parte de uma alteração na reserva. ...


Hotel Chain
1
1..*
Clerk
1..*
1
contactedHotel
*
Hotel
1
1
*
*
1
1..*
Customer
1
*
Reservation
*
allocation
Room
0..1
1
*
1
*
contactAddress
0..1
1
Address
RoomType
0..1
1
Bill
Payment
0..1
1

Definir as regras de negócio

- Adicionar novas regras de negócio requeridas, além dos conceitos e associações: introduzir novos atributos e restrições
- Neste estágio as restrições podem ser especificadas em linguagem natural.

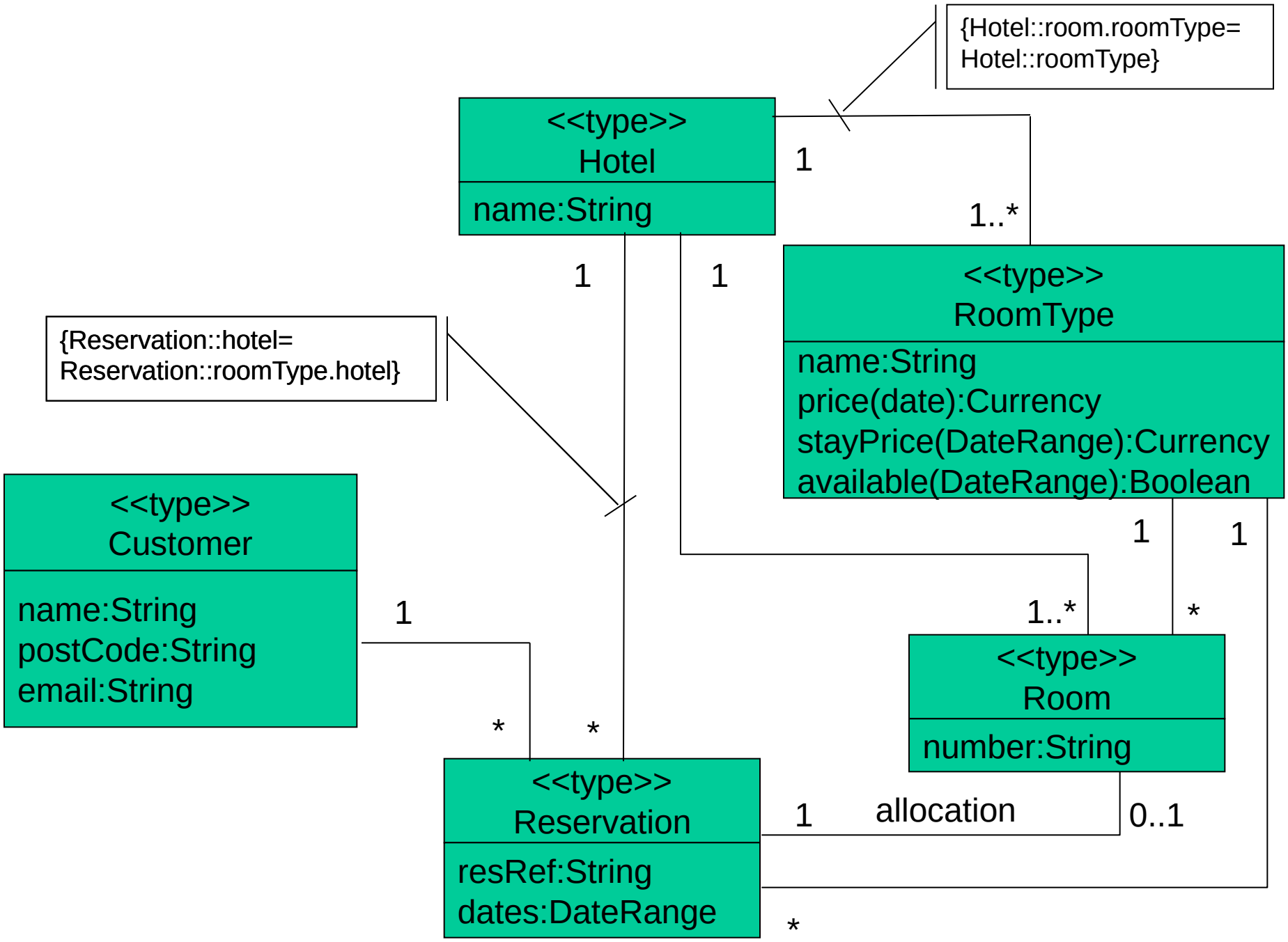

<<type>>
Hotel
name:String
1
1..*
1
1
<<type>>
RoomType
name:String
price:Currency
<<type>>
Customer
name:String
postCode:String
email:String
1
1
1
1..*
*
<<type>>
Room
number:String
*
*
<<type>>
Reservation
resRef:String
dates:DateRange
1
allocation
0..1
*

Definir as regras de negócio

- Inicialmente, identificar quais associações podem ser derivadas de outras. Exemplo: uma reserva deve ser feita para quartos do mesmo hotel e o tipo de quarto deve estar disponível no mesmo hotel.

Definir as regras de negócio

- Regra de disponibilidade
  - O número de reservas no intervalo < número de quartos → novo parâmetro: available(DateRange)
  - Não pode haver mais reservas para uma data do que quartos.
- Regra de preço
  - O preço da estada é a soma dos preços para os dias da estada
  - O preço deve ser então parametrizado pela data


{Hotel::room.roomType=
Hotel::roomType}
<<type>>
Hotel
name:String
1
1..*
1
1
<<type>>
RoomType
name:String
price(date):Currency
stayPrice(DateRange):Currency
available(DateRange):Boolean
{Reservation::hotel=
Reservation::roomType.hotel}
1
1
1..*
*
<<type>>
Customer
name:String
postCode:String
email:String
1
*
*
<<type>>
Room
number:String
<<type>>
Reservation
resRef:String
dates:DateRange
1
allocation
0..1
*

Identificar os Tipos Básicos

- Um tipo básico de negócio tem existência independente dentro do negócio, caracterizada por:
  - Um identificador de negócio, usualmente independente de outras entidades
  - Uma existência independente – nenhuma associação obrigatória, exceto para um tipo categorizador
- No exemplo: H*otel* e C*ustomer*

Identificar os Tipos Básicos

- Os tipos básicos são indicados pelo estereótipo <<core>>.
- Como a UML não permite mais do que um estereótipo por classe, <<core>> substitui <<type>> e incorpora o seu significado.
- Room não é um tipo básico.
  - Room tem uma associação obrigatória com Room Type, mas este é um categorizador de Room.
  - Room não tem existência independente, porque tem uma associação obrigatória com Hotel.

Criar as Interfaces de Negócio e Atribuir Responsabilidades

- Regra geral: criar uma interface para cada tipo básico do modelo de tipos de negócio.
- Tipicamente, cada interface gere a informação representada pelo tipo básico e seus tipos detalhe (ou tipos de detalhamento)
- Estas interfaces são normalmente denotadas por IxxxMgt (Ex. ICustomerMgt, significando interface for Customer Management)
- Em situações especiais, instâncias dos tipos básicos poderiam formar componentes de objetos separados, mas isso não deve ser feito neste passo.

Criar as Interfaces de Negócio e Atribuir Responsabilidades (cont.)

- O significado da interface neste momento é que uma objeto componente, por ex. ICustomerMgt, gerencia muitos clientes.
- Um particular C*ustomer* deve ser referenciado por uma chave identificadora.
- Os diagramas de tipos de negócio passam agora a ser chamados de diagramas de responsabilidade das interfaces.

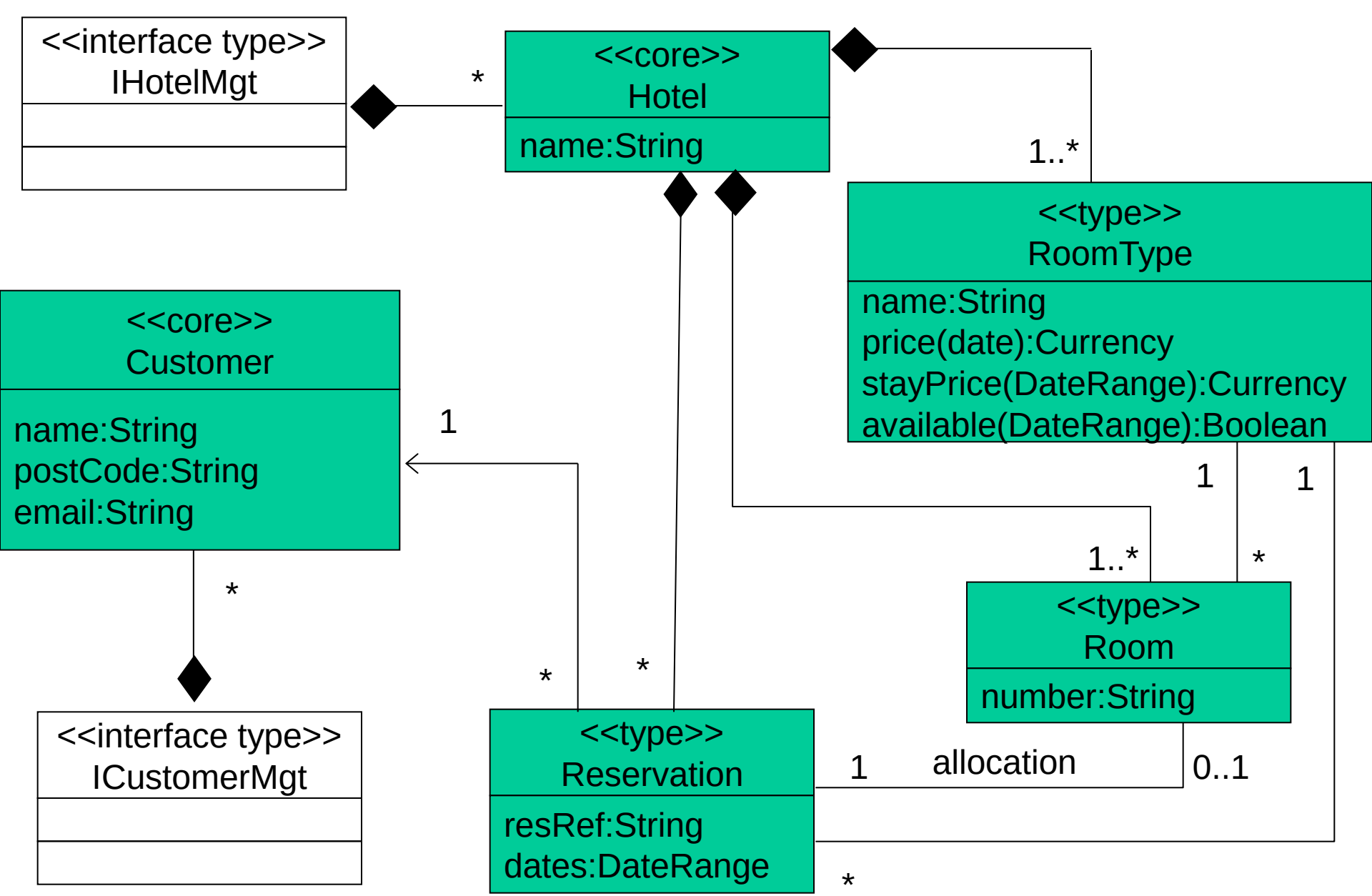


Diagrama de Responsabilidade das Interfaces do Modelo de Tipos de Negócio

Criar as Interfaces de Negócio e Atribuir Responsabilidades (cont.)

- O diagrama de responsabilidade das interfaces mostra que informação vai ser gerida por qual interface.
- Os tipos de detalhe são alocado à sua interface, que deve ser única e identificados pela relação de composição (agregação)
- Se o tipo de detalhe detalha mais do que um tipo e estes estão alocados para diferentes interfaces, ele deve ser alocado de forma a diminuir o acoplamento.

Criar as Interfaces de Negócio e Atribuir Responsabilidades (cont.)

- O tipo *Reservation* foi alocado para IHotelMgt por que tem apenas uma referência para *Customer* e está mais acoplado com outras informações geridas por IHotelMgt.
- Isso é indicado no diagrama tornando a associação entre *Customer* e *Reservation* navegável apenas na direção de *Customer*.

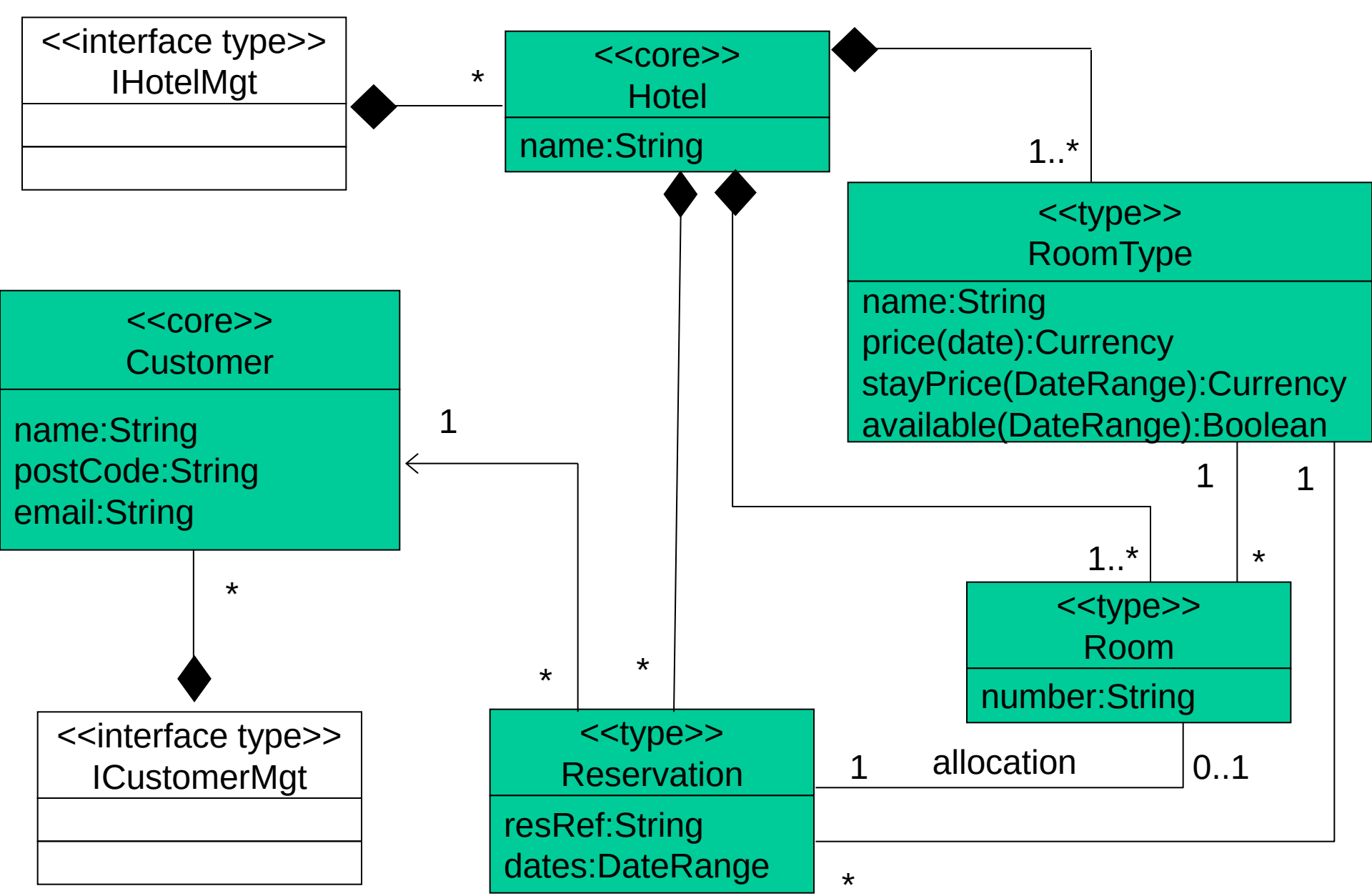


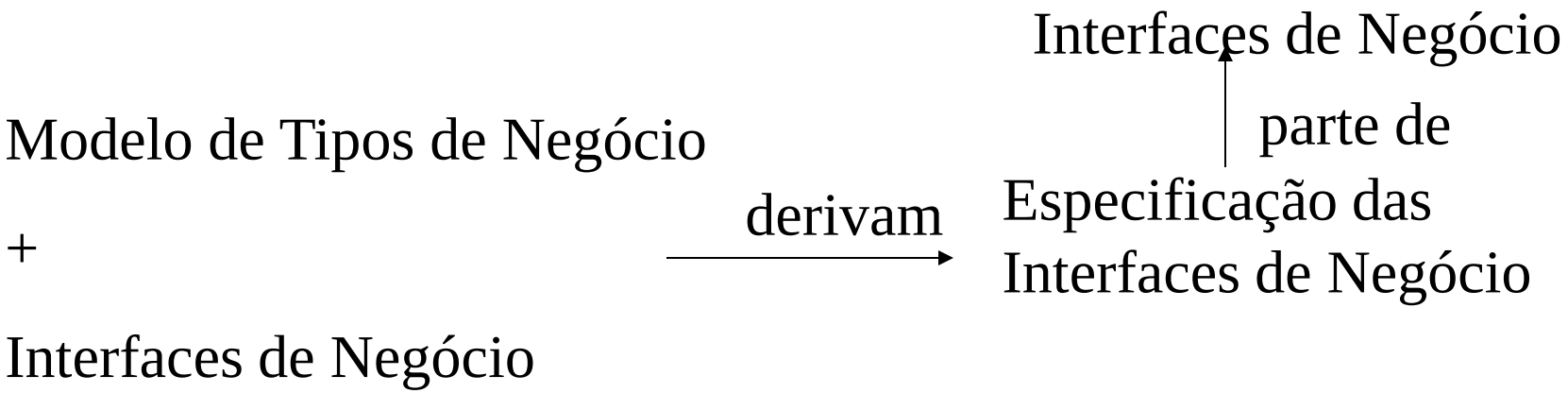
Diagrama de Responsabilidade das Interfaces do Modelo de Tipos de Negócio

Alocação de Responsabilidade a Associações

- As associações que existem entre tipos geridos por diferentes interfaces são chamadas de **associações entre-interfaces**.
- Associações entre-interfaces pode ser vista como um tipo específico de dependência.
- No exemplo, IHotelMgt é responsável por registrar a referência para *Customer*.

Criação das especificações iniciais das interfaces

- O modelo de tipos de negócio é um artefato interno e não um produto final, podendo ser descartado ou não. Os autores preferem mantê-lo.



Sistemas e Interfaces Existentes

- As interfaces do sistema podem ser ampliadas por interfaces que são parte do ambiente onde o sistema será implantado:
  - Interfaces de uso obrigatório
  - Sistemas com os quais se integrar, fora do escopo deste projeto etc
- No exemplo, existe o sistema de faturamento.

Arquitetura da Especificação de Componentes

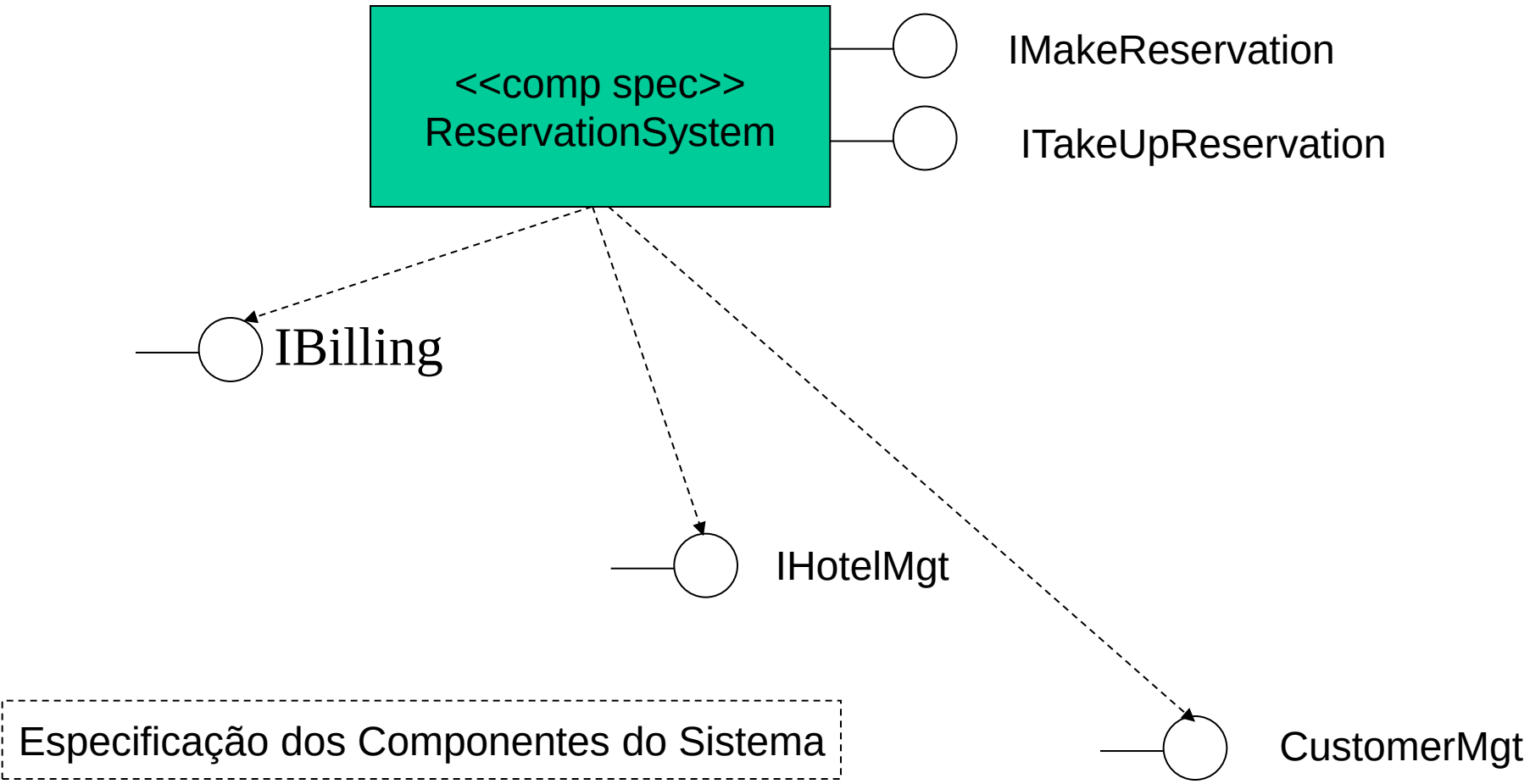
- Um conjunto inicial de Especificações de Componentes é criado, formando-se uma idéia de como eles são integrados.
- Os componentes são as unidades de realização.
- Devemos escolher componentes que fazem sentido ao se construir ou comprar uma funcionalidade.
- Um sistema pode ser melhorado mudando-se componentes seletivamente.

Arquitetura da Especificação de Componentes

- Na maior parte dos casos serão criadas especificações de componentes para cada especificação de interface identificada.
- Múltiplas interfaces ou especificação de um único componente poderão ser criadas se:
  - Os conceitos representados pelas diferentes interfaces têm o mesmo tempo de vida.
  - As interações entre as interfaces são complexas, frequentes ou envolvem grande quantidade de dados.
  - A granularidade dos componentes deve ser mantida em tamanho razoável.
  - As implementações das interfaces devem ser substituídas simultaneamente, como uma unidade.

Especificações dos Componentes do Sistema

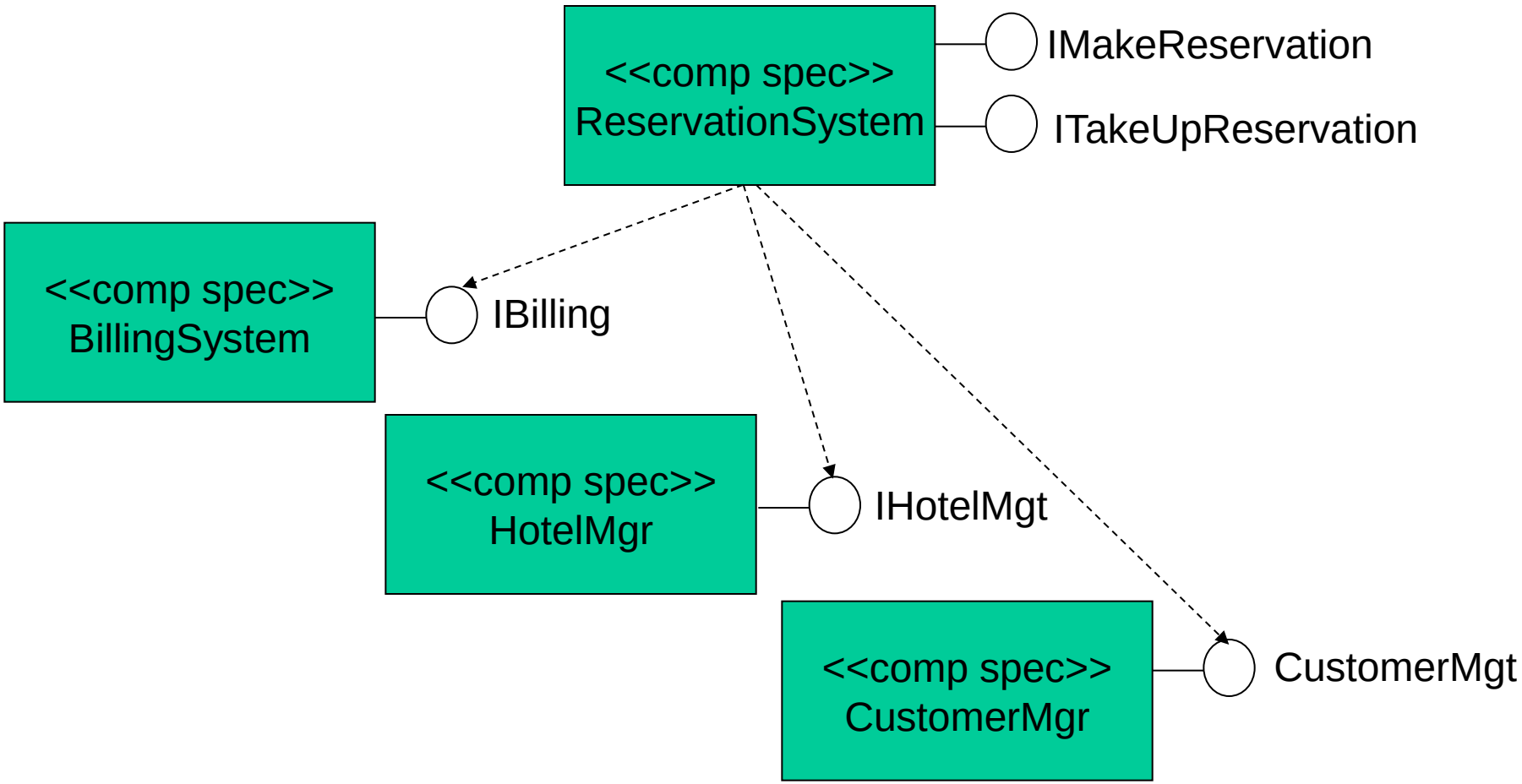
- No caso de estudo, as especificações derivadas dos casos de uso são fortemente sobrepostas e gerem conceitos que têm o mesmo tempo de vida.
- Elas poderiam ser apoiadas por uma única especificação de componente. Mas a conexão do sistema de faturamento não faz sentido para as outras especificações


<<comp spec>>
ReservationSystem
IMakeReservation
ITakeUpReservation
IBilling
IHotelMgt
CustomerMgt
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