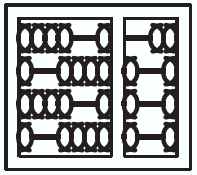
Desenvolvimento Baseado em Componentes
e
o Processo UML Components

Cecília Mary Fischer Rubira
Patrick Henrique da Silva Brito



Instituto de Computação (IC)
Universidade Estadual de Campinas
(Unicamp)
INF064'2008

Roteiro

(1) Desenvolvimento Baseado em Componentes;

(2) O Processo UML Components.

DBC e o Processo UML Components

Motivação

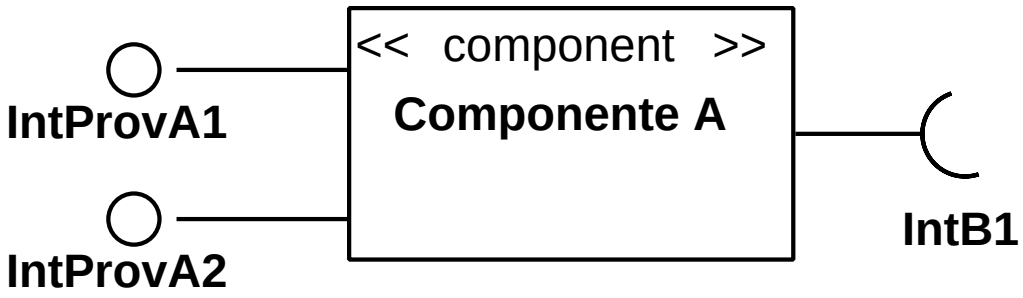
- Os sistemas de software estão cada vez mais complexos e "urgentes";
- Novos Requisitos:
  - Controle da complexidade do software;
  - Alta velocidade de desenvolvimento;
- Atendimento:
  - Elementos abstratos (granularidade alta);
  - Maior reutilização de código;

O Conceito de Componentes

Conceito

- Um componente de software é um conceito independente de tecnologia;
- Representa uma unidade de computação;
- Suas principais características são:
  - Define seus serviços oferecidos e as suas dependências $\Rightarrow$ **interfaces providas e requeridas**;
  - A comunicação entre componentes se baseia unicamente nas suas interfaces $\Rightarrow$ **baixo acoplamento**;

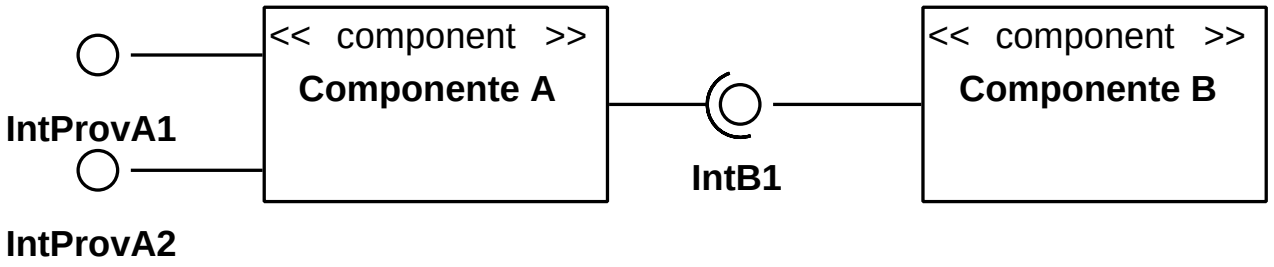
Representação de um Componente



Desenvolvimento Baseado em Componentes (DBC)

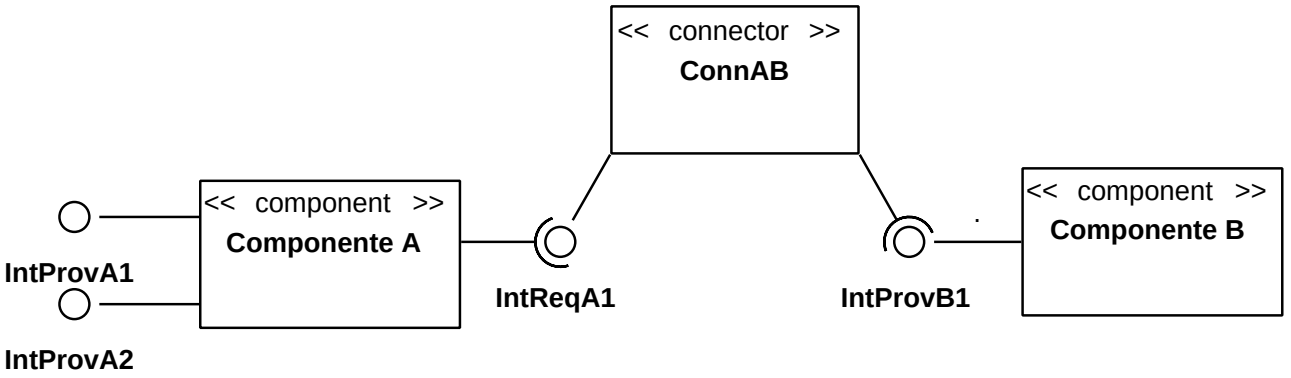
- No DBC, a aplicação é construída a partir da composição de componentes de software;
- Ganho de produtividade:
  - Reutilização de componentes existentes na construção de novos sistemas;
  - Possibilidade de utilizar componentes de prateleira;
- Ganho de qualidade:
  - Uso de componentes já empegados e testados em outros contextos;
  - Importante: Re-aplicar os testes no novo contexto de uso.

Um Sistema Baseado em Componentes (I)



- Componente A → *oferece* duas interfaces: `IntProvA1` e `IntProvA2`; *requer* as operações da interface `IntB1`.
- Componente B → *oferece* a interface `IntB1` e não *requer* nenhum serviço externo.

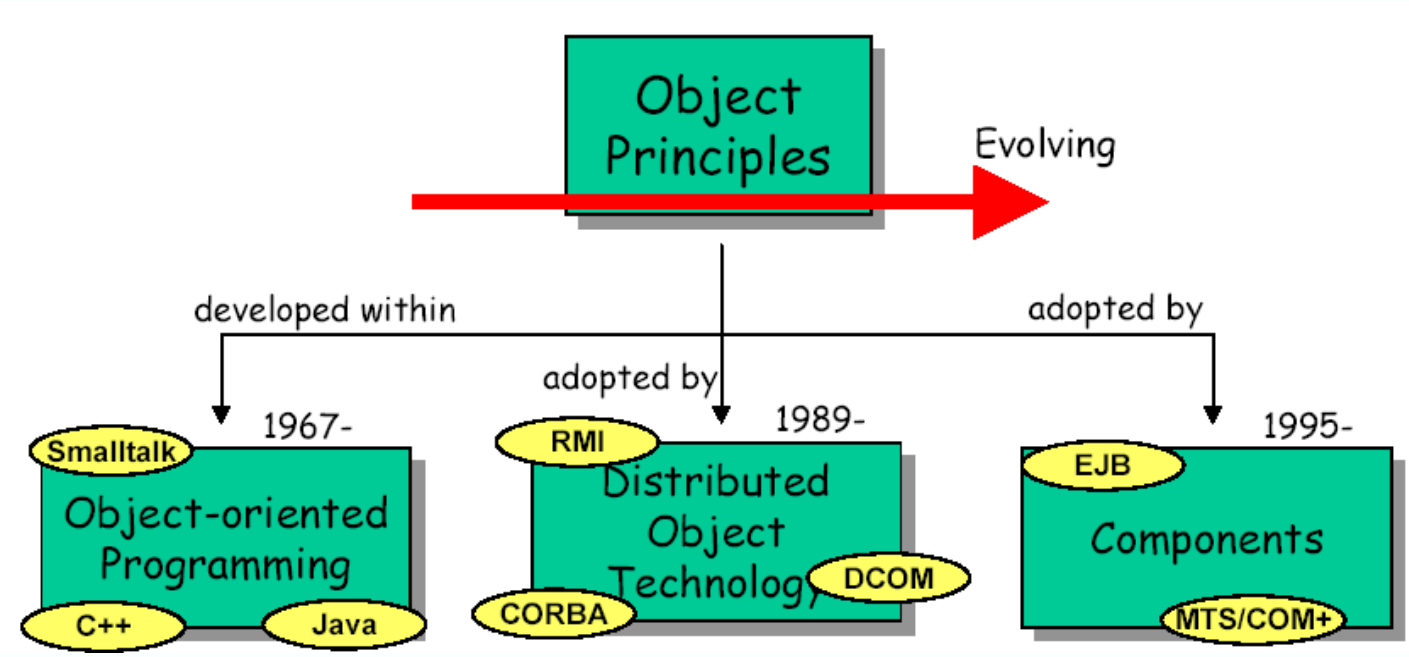
Um Sistema Baseado em Componentes (II)



- Componente A → *oferece* duas interfaces: `IntProvA1` e `IntProvA2`; *requer* as operações da interface `IntReqA1`.
- Componente B → *oferece* a interface `IntProvB1` e não *requer* nenhum serviço externo.
- ConnAB → intermedia a comunicação entre o `Componente A` e o `Componente B`.

Contextualização

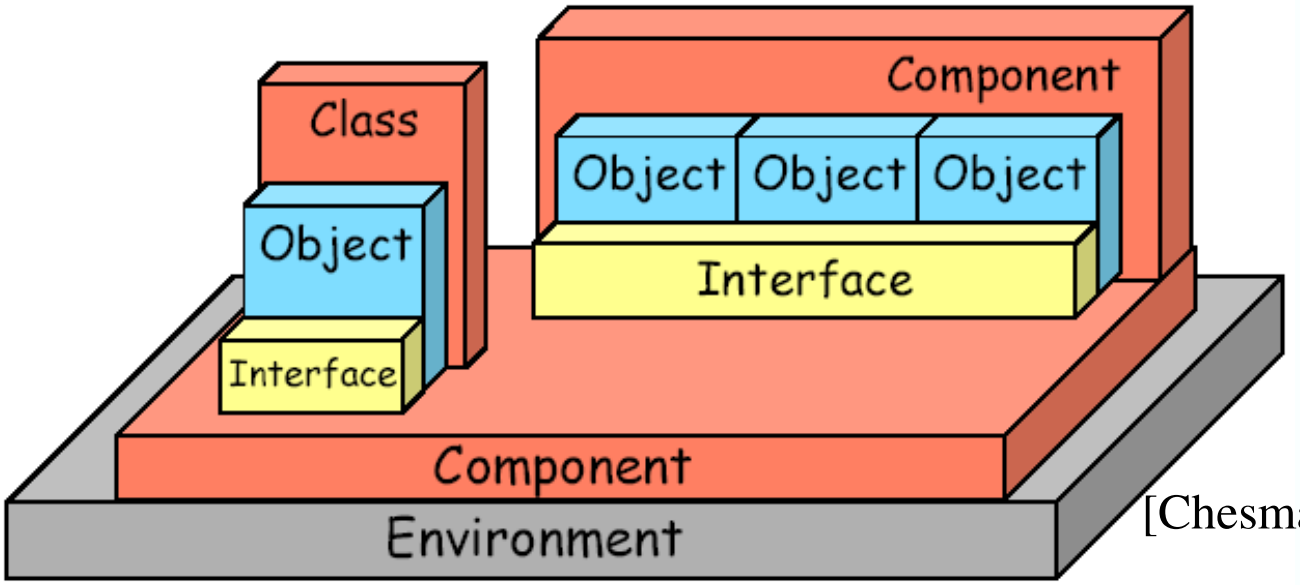
- OO surgiu em 1967 *(simula e smalltalk)*;
- A idéia de DBC para software surgiu em 1976, mas o marco inicial foi o WCOP'96 *(COM e DCOM)*;



[Chesman & Daniels]

Componentes e Classes

- Ambos necessitam se conectar ao ambiente;
- Ambos são unidades de desenvolvimento e evolução (cada um no seu paradigma);
- Granularidades diferentes.



[Chesman & Daniels]

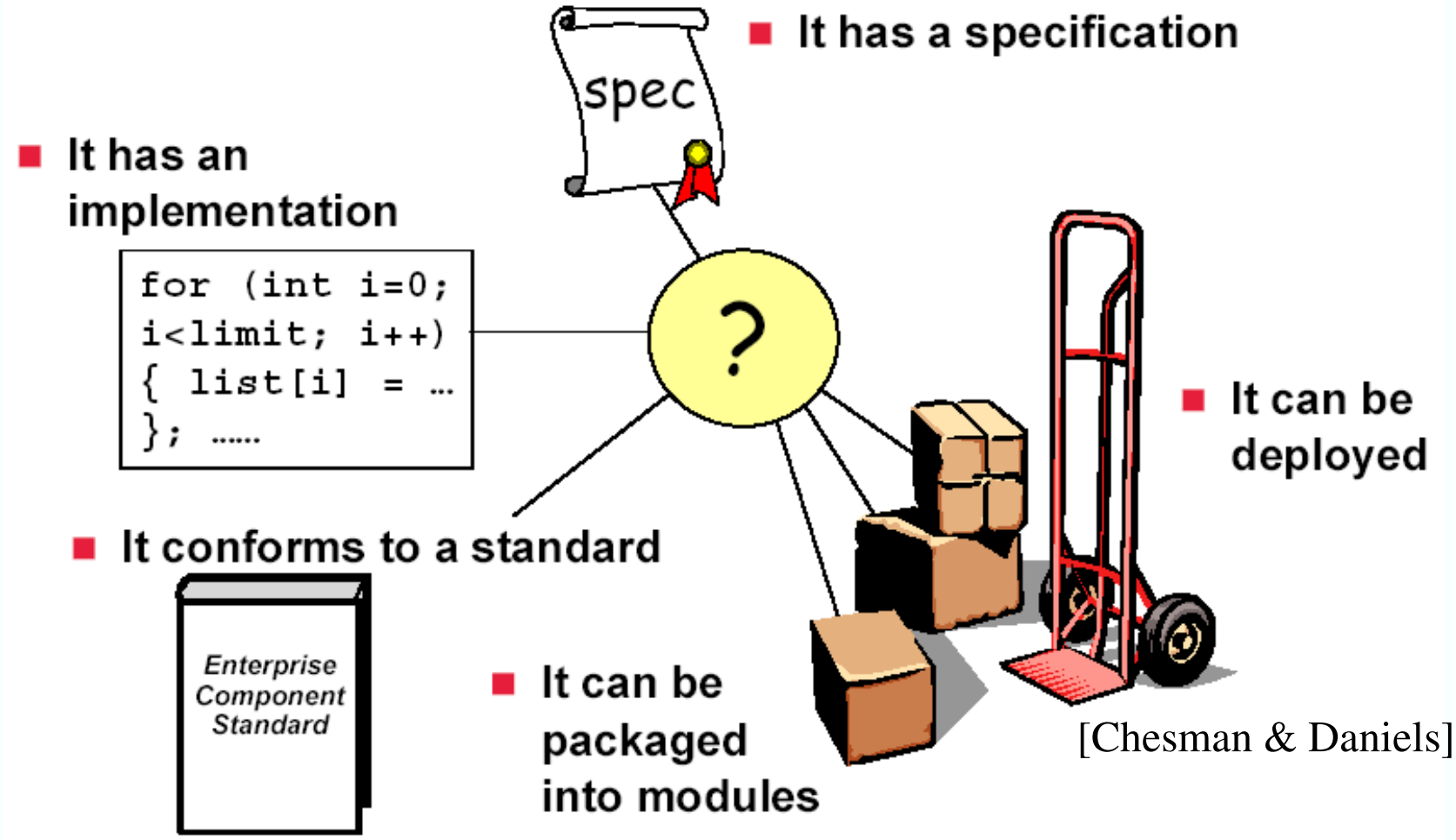
Comparação da Granularidade

- Classe:
  - Contrato de uso;
  - Lista de operações;
  - Define como as operações afetam o modelo da interface;
  - Efeito local.
- Componente:
  - Contratos de uso e de realização;
  - Lista de interfaces;
  - Define o relacionamento entre os vários modelos de interfaces;
  - Implementação em termos do uso de outras interfaces.

Ambientes e Modelos de Componentes

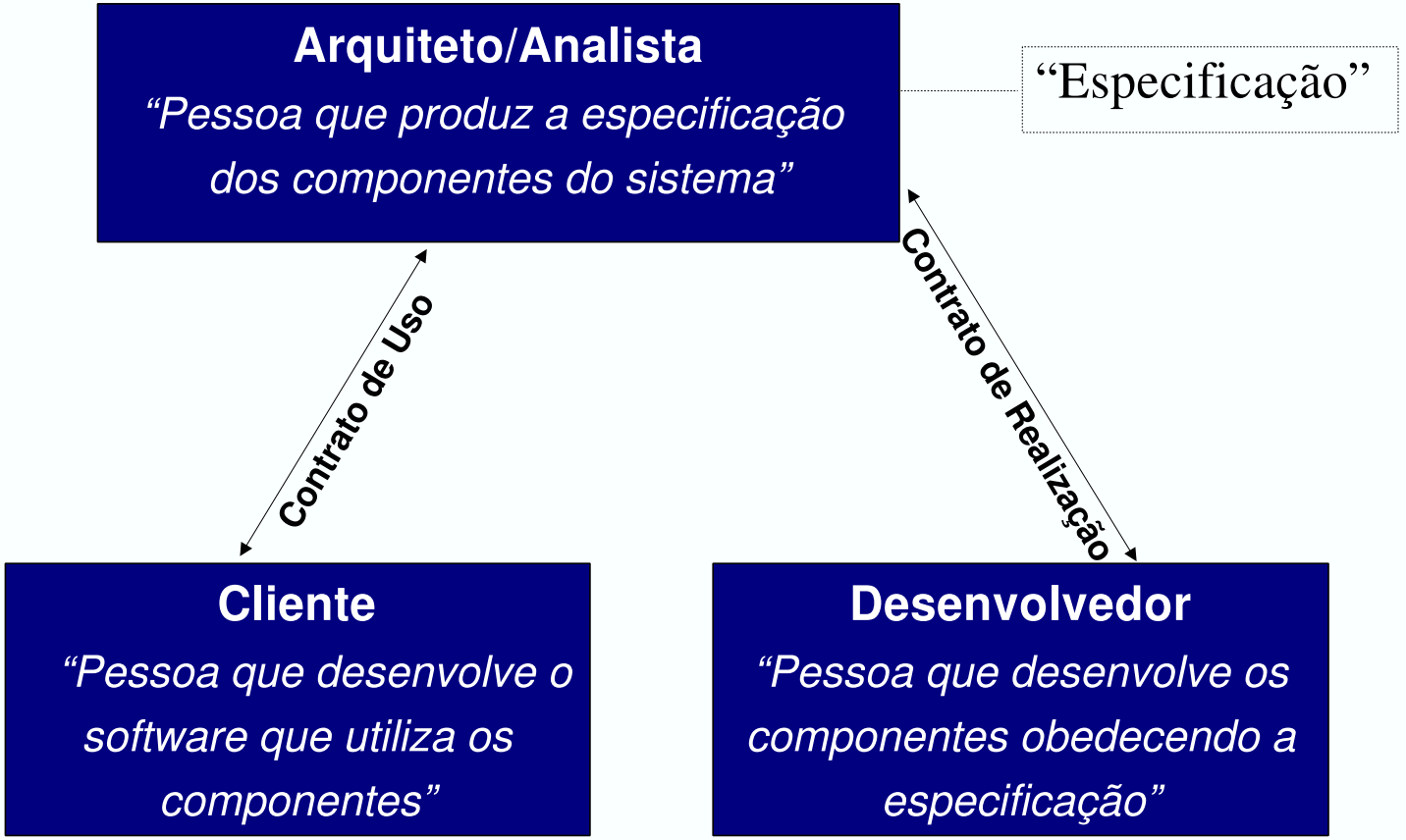
- Oferece um conjunto de serviços:
  - Normalmente configuráveis, não programáveis;
  - Ex.: acesso remoto, transação, persistência, etc.
- Define um conjunto de restrições:
  - Ex.: número máximo de interfaces.
- Alguns modelos existentes:
  - EJB (Enterprise JavaBeans) - Sun Microsystems;
  - CCM (Corba Component Model) - OMG;
  - COM/COM+ (Component Object Model+) - Microsoft.

Os Cinco Princípios Básicos



[Chesman & Daniels]

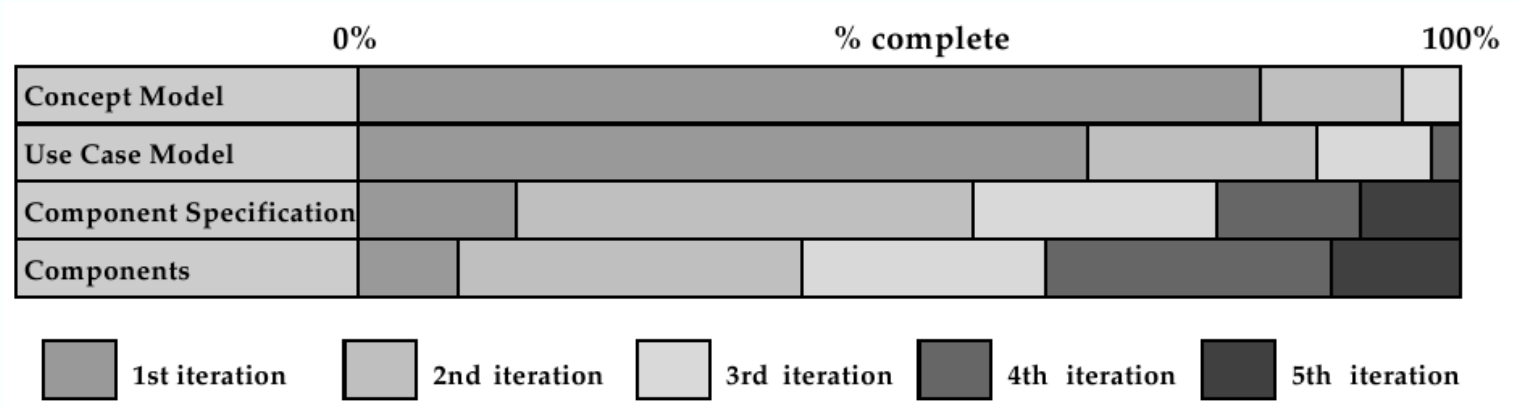
Papéis Envolvidos no DBC



O Processo UML Components

Visão Geral (I)

- Processo de DBC;
- Iterativo (mudança tardia nos requisitos)



Visão Geral (II)

- Voltado para sistemas de informação;

  Adota uma arquitetura específica;

  Estrutura o sistema em cinco camadas.

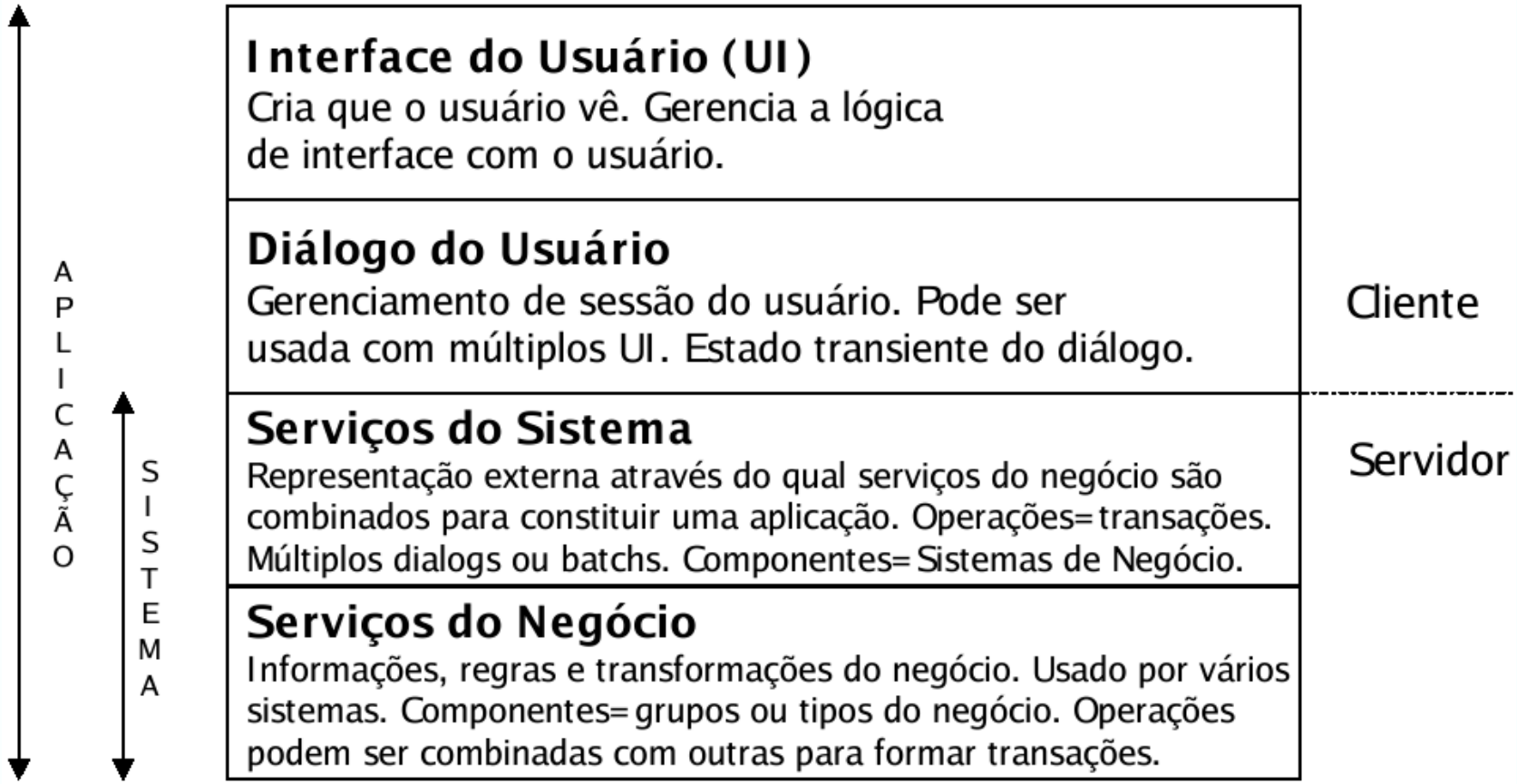
- De fácil entendimento e utilização;

- Foco na Especificação dos Componentes;

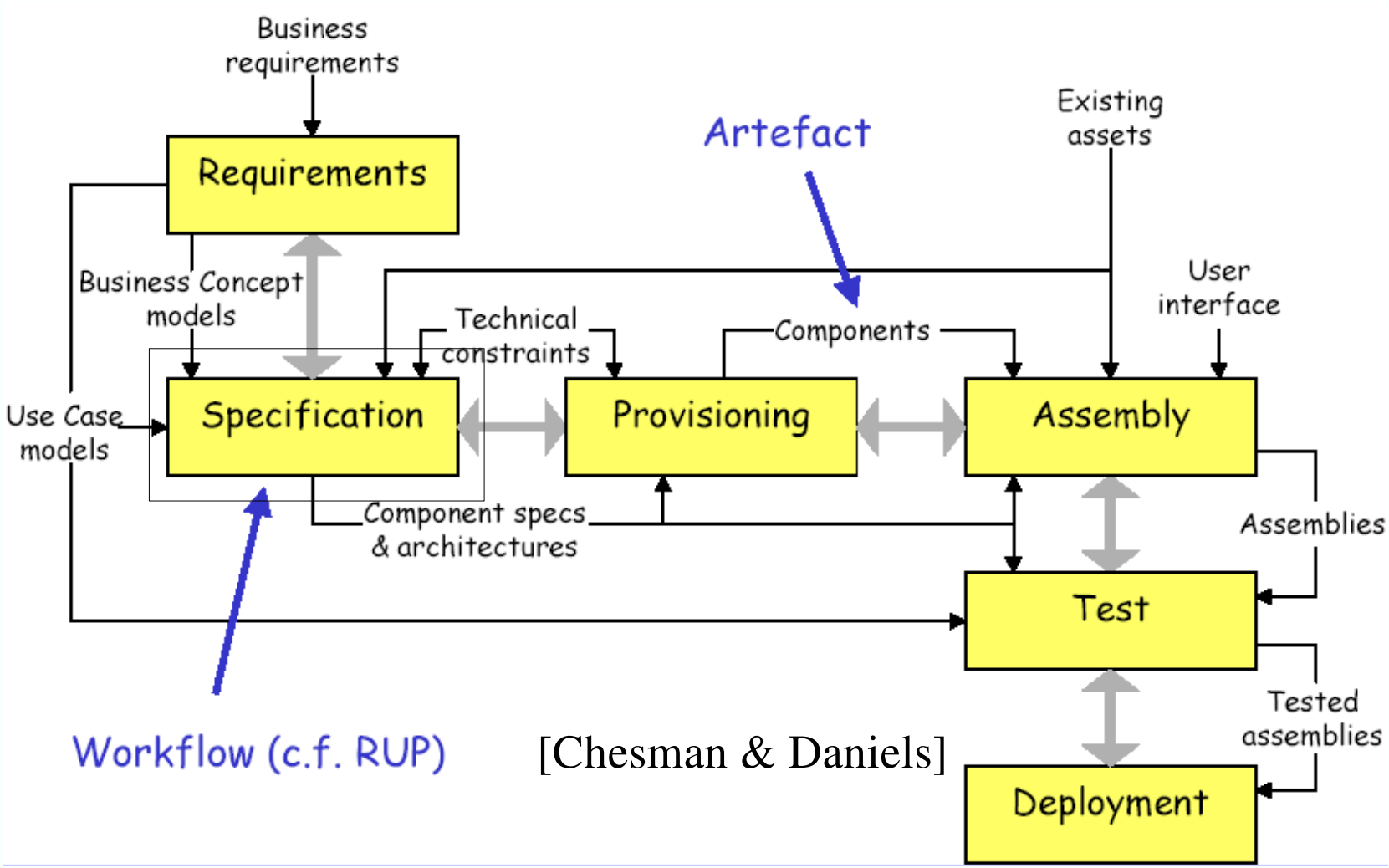
  (falta implementação, montagem e testes).

- Foco no detalhamento dos componentes internos.

Arquitetura Adotada



Workflow Geral do Processo



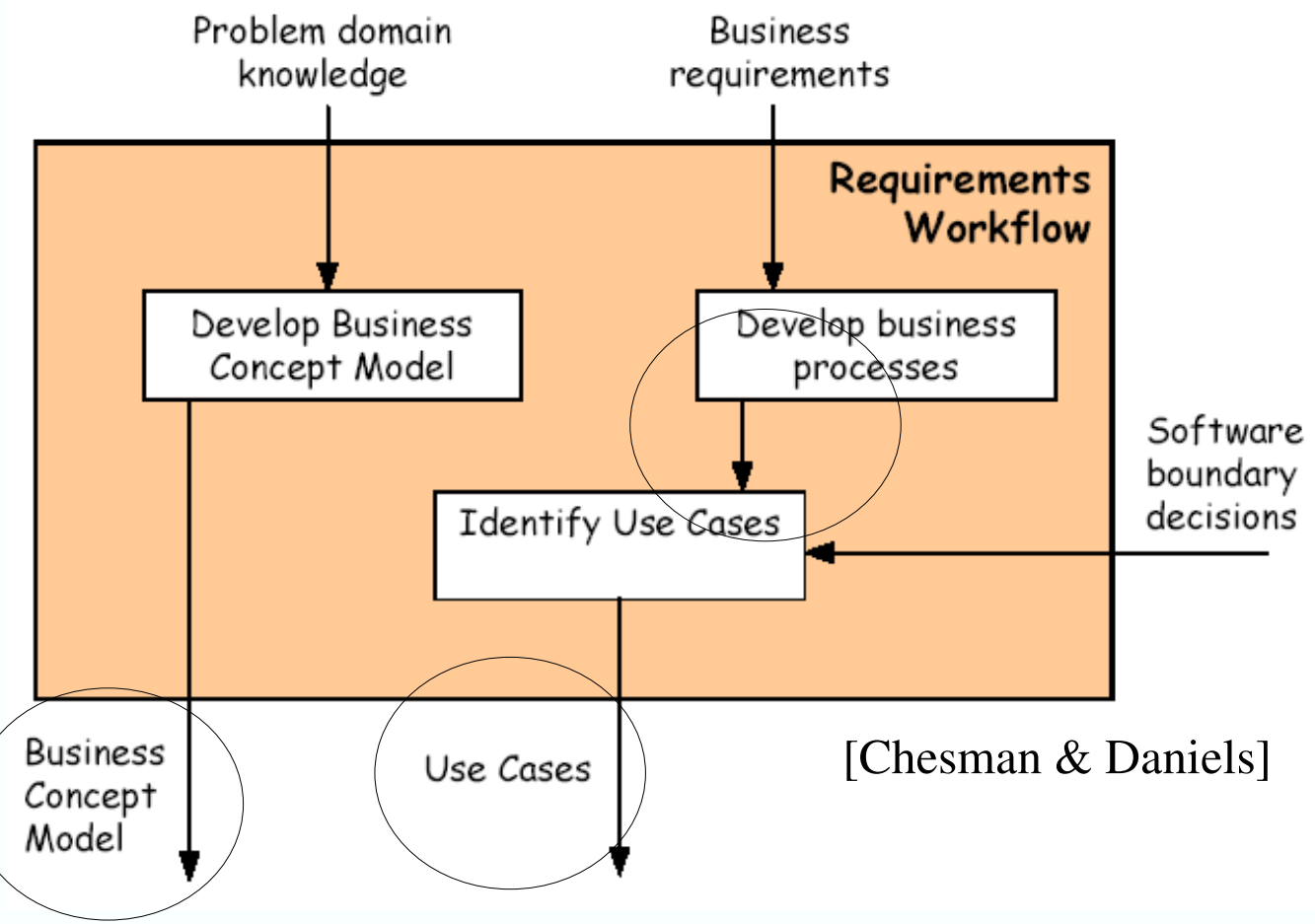
Fases do UML Components

- Será explicado o papel de cada fase do processo;
- A ênfase maior é na fase de especificação do sistema (análise e projeto);
- Serão utilizados exemplos de um sistema de gerenciamento de hotéis.

Especificação de Requisitos

- O processo UML Components lista os artefatos envolvidos nesta etapa, mas não detalha a forma de produzí-los.

Workflow de Especificação de Requisitos

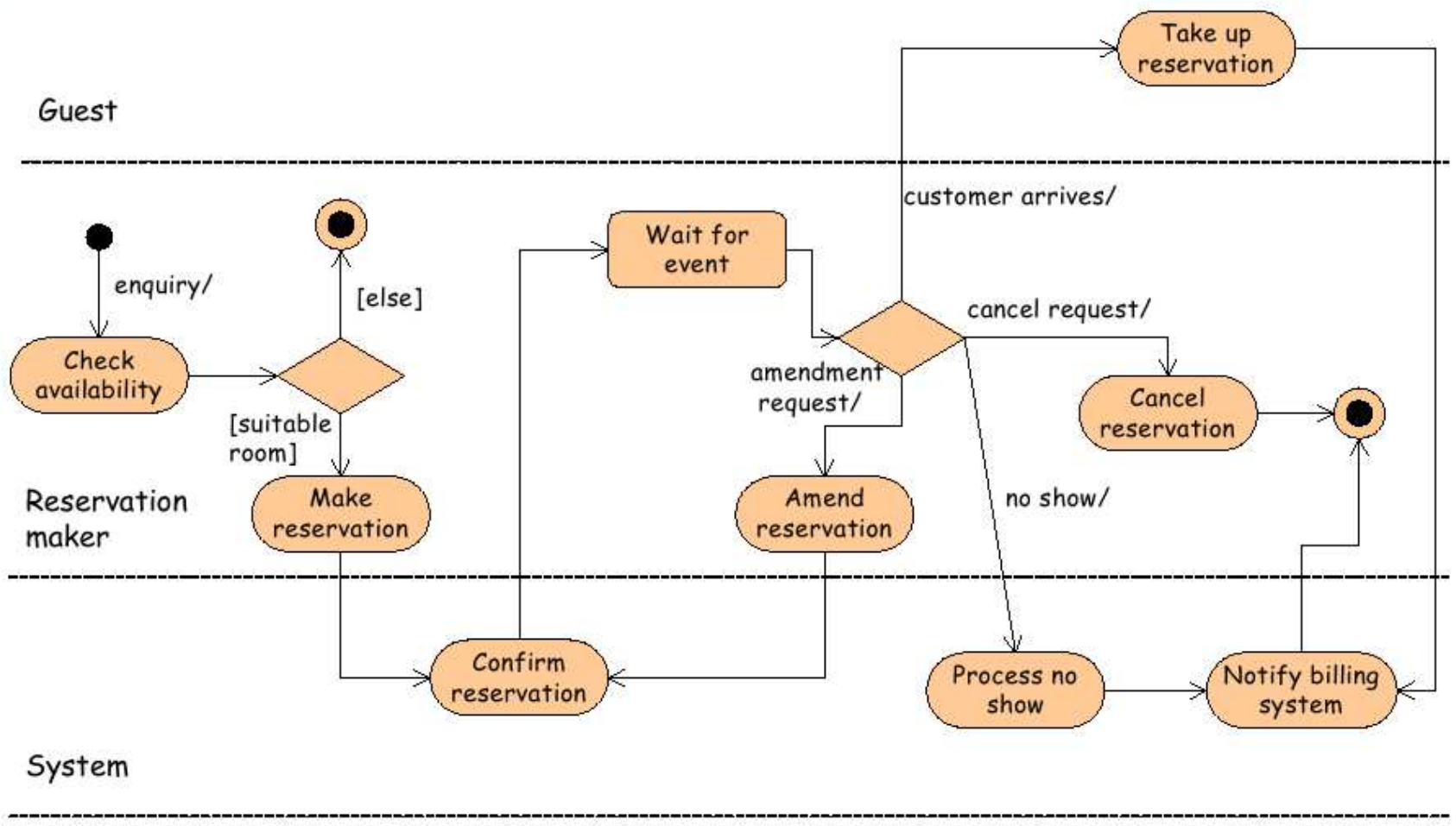


[Chesman & Daniels]

Processo do Negócio (Business Process) (I)

- Artefato interno (não é uma saída da fase de requisitos);
- Representação gráfica (diagrama de atividades UML) das atividades que representam o funcionamento do negócio em si, não apenas as atividades automatizadas;
- Envolve várias funcionalidades do sistema e mostra a relação (ordem de precedência) entre elas;
- Distinção clara entre as atividades manuais e automáticas do processo;
- Util para compreender o domínio.

Processo do Negócio (Business Process) (II)



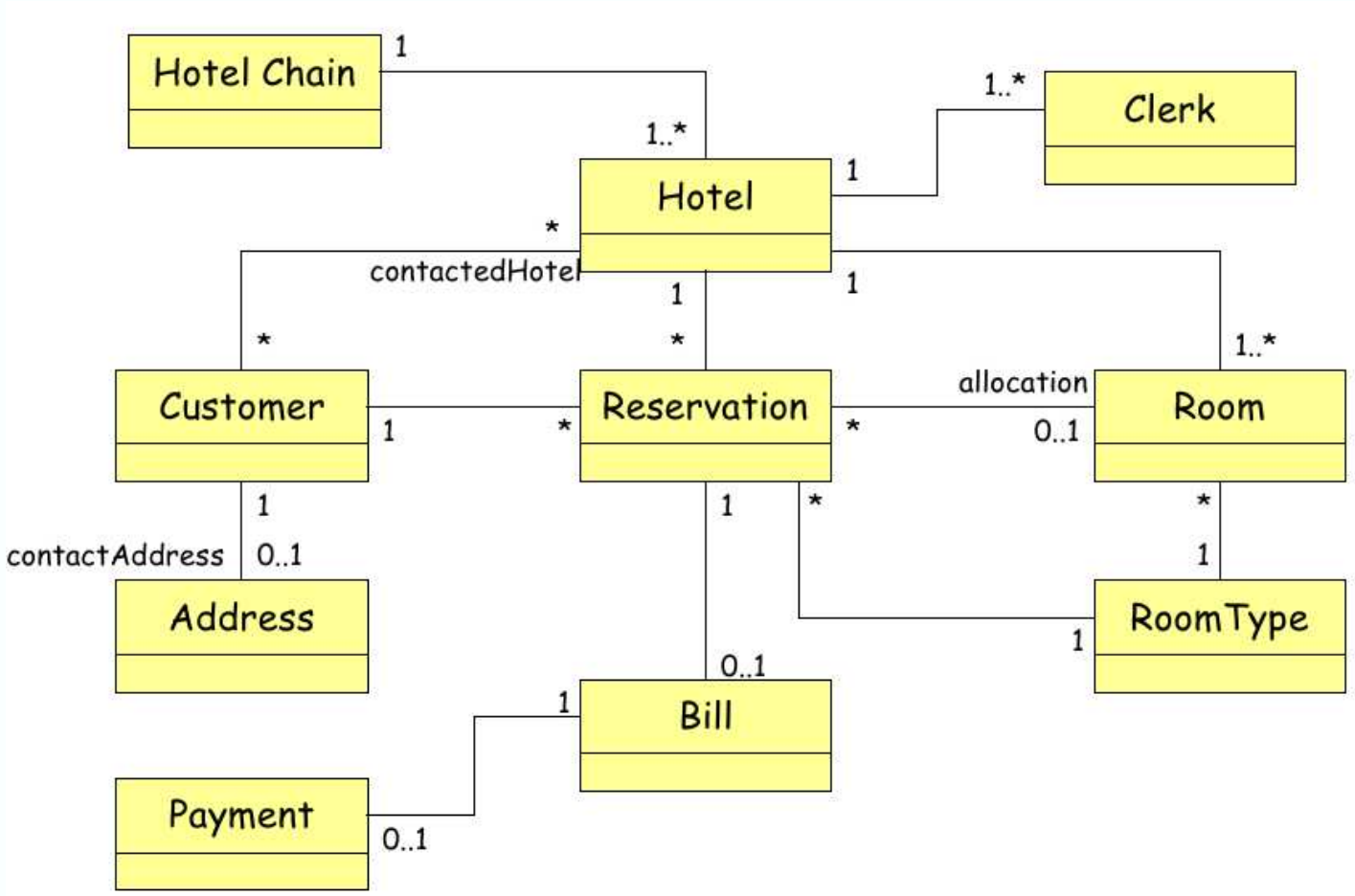
Modelo Conceitual do Negócio

(Business Concept Model) (I)

- Representa as entidades do negócio como um todo;
- Util para compreender o papel do sistema;
  - Contextualiza os desenvolvedores.
- Modelado a partir de entidades e os relacionamentos entre elas;
  - Semelhante a um MER de banco de dados.

Modelo Conceitual do Negócio

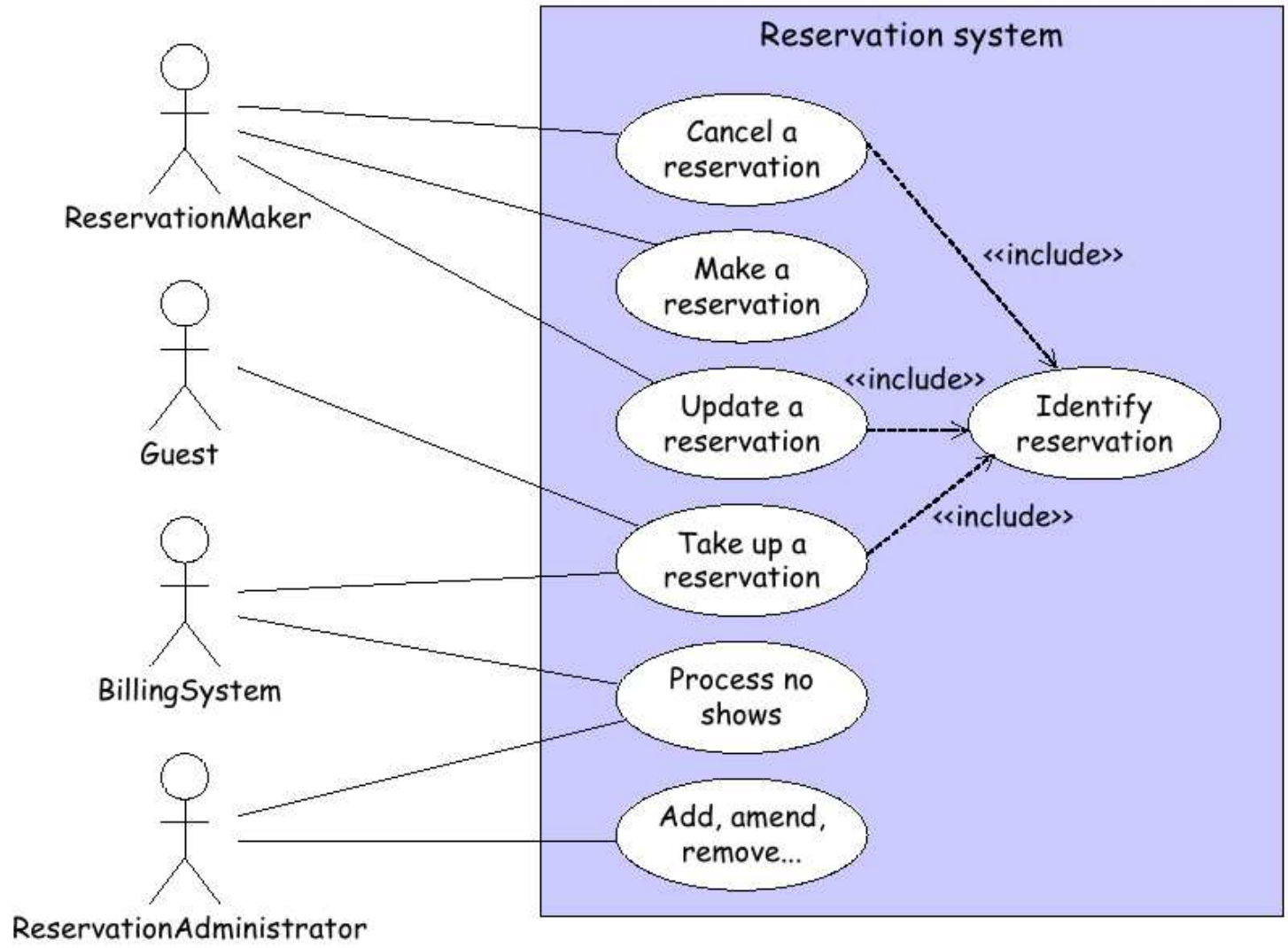
(Business Concept Model) (II)



Modelo de Casos de Uso

- Representação de requisitos funcionais adotada em UML;
- Detalhamento das funcionalidades do sistema e do seu relacionamento com os interessados externos (atores);
- Representa graficamente as principais funcionalidades do sistema e o relacionamento entre elas;
  - Diagrama de casos de uso.
- Representa textualmente o detalhamento das funcionalidades;
  - Especificação dos casos de uso.

Diagrama de Casos de Uso



Especificação dos Casos de Uso (I)

- "Baseado no Business Process";
- **Nome**: Make a reservation;
- **Iniciador**: Reservation Maker;
- **Objetivo**: Reserve room(s) at a hotel;
- **Atores envolvidos**: ReservationMaker;
- **Pré-condições**: O hóspede deve estar previamente cadastrado no sistema;
- **Pós-condições**: Existe uma reserva a mais no sistema;

Especificação dos Casos de Uso (II)

- **Fluxo principal**:
  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 postcode (zip code);
  6. Reservation Maker provides contact email address;
  7. System makes reservation and allocates tag to reservation;
  8. System reveals tag to Reservation Maker;
  9. System creates and sends confirmation by email.

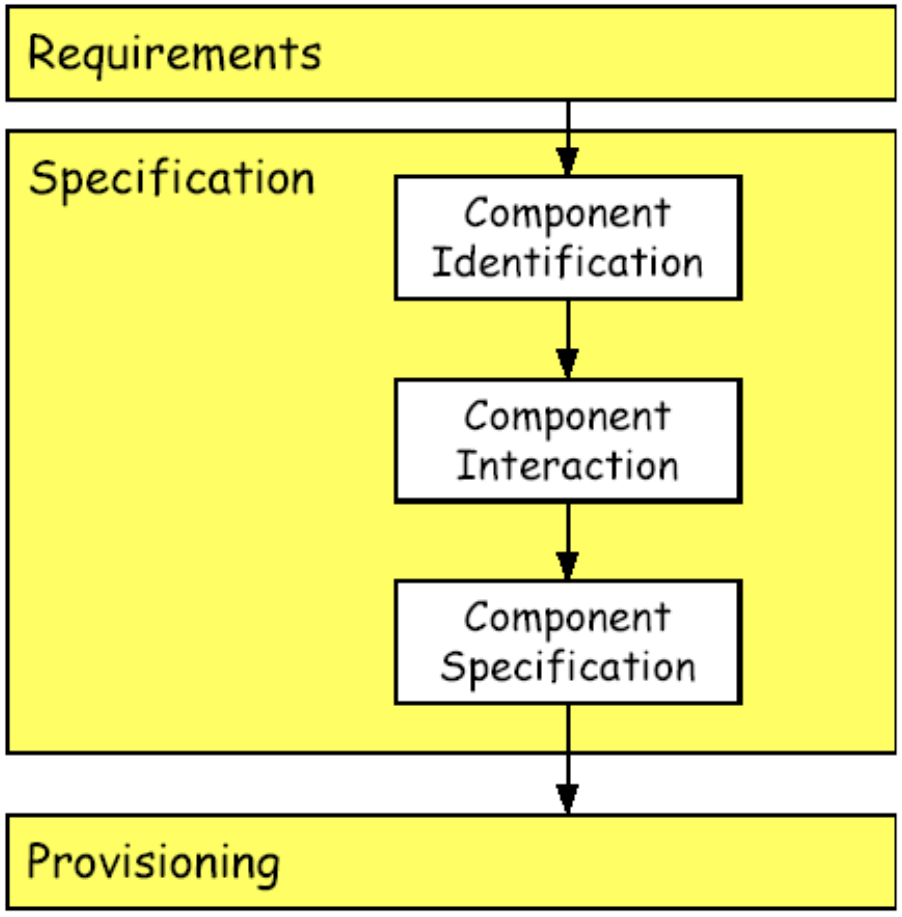
Especificação dos Casos de Uso (III)

- **Fluxo Alternativo 1**: Se no passo 3 do fluxo básico não houver nenhum quarto disponível, o sistema deve oferecer alternativas em hotéis próximos da mesma rede. O cliente pode selecionar uma das alternativas ou desistir.
- **Fluxo Alternativo 2**: No passo 4 do fluxo básico, o autor da reserva pode cancelar o procedimento de reserva.
- **Fluxo Alternativo 3**: No passo 6 do fluxo básico, o sistema pode detectar que o autor da reserva digitou um e-mail diferente do cadastrado e sugere a atualização do cadastro.

Especificação dos Componentes

- Além de apresentar os artefatos de entrada e saída dessa fase, o processo UML Components detalha a seqüência de atividades para produzí-los.

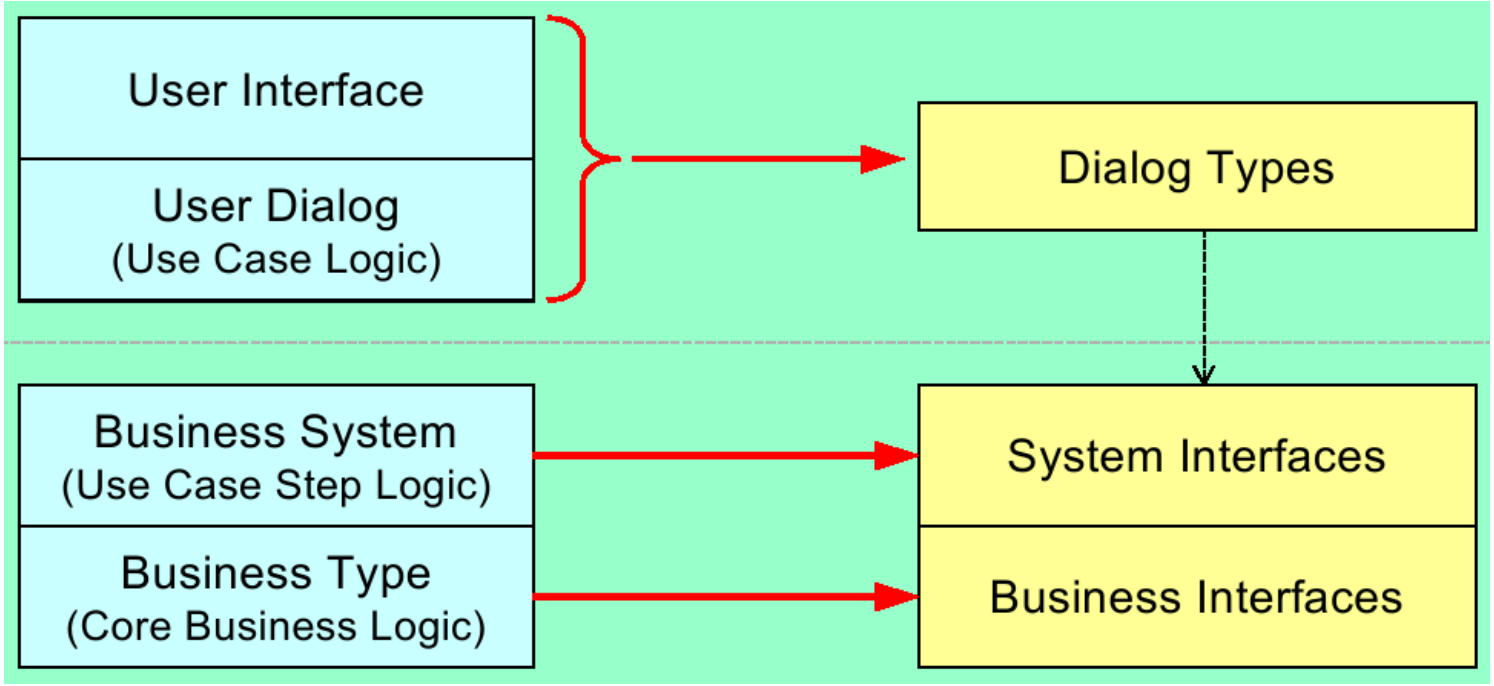
Workflow da Especificação dos Componentes



[Chesman & Daniels]

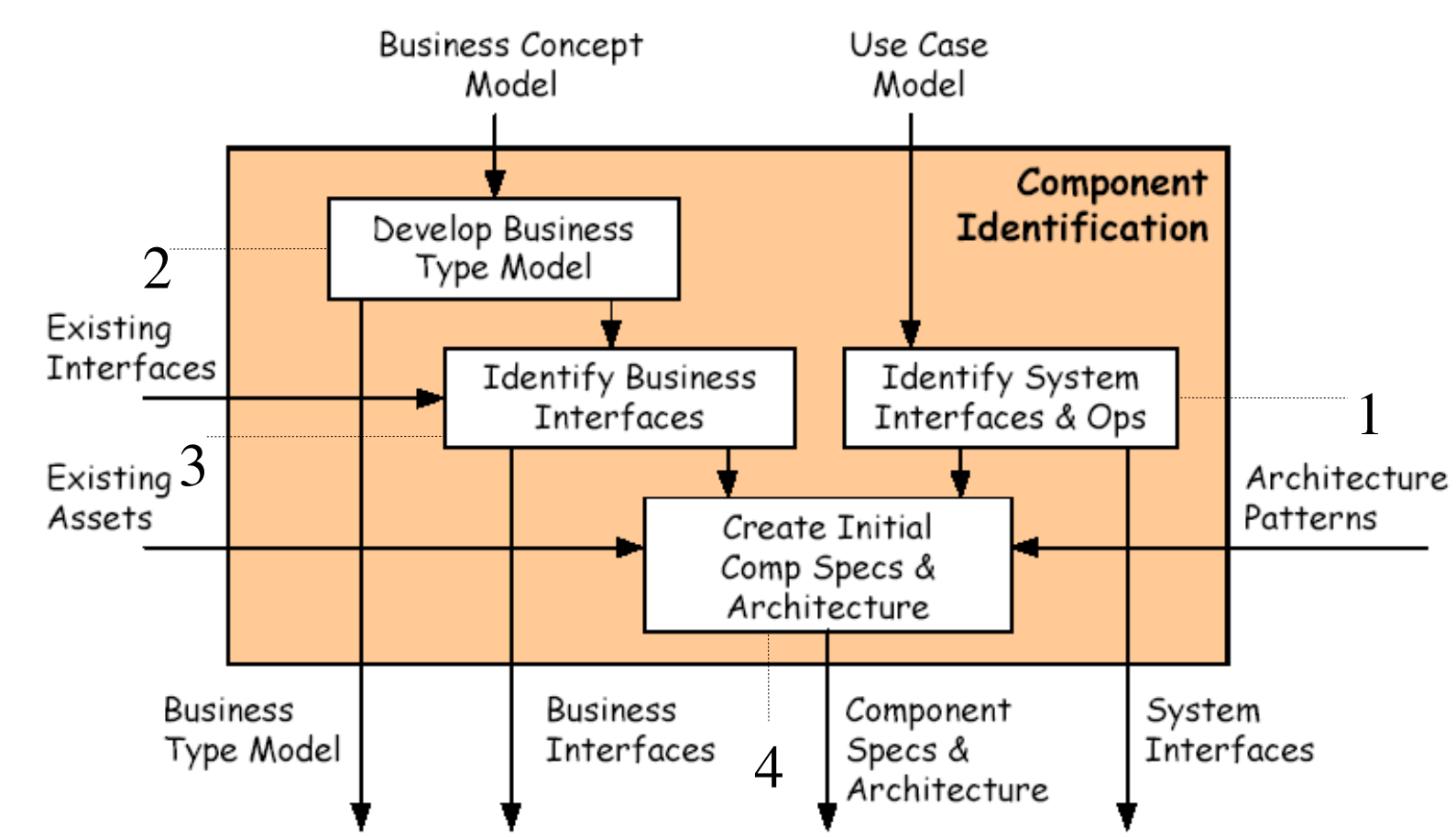
- Três etapas: identificação dos componentes, interação entre os componentes e especificação final.

Identificação dos Componentes



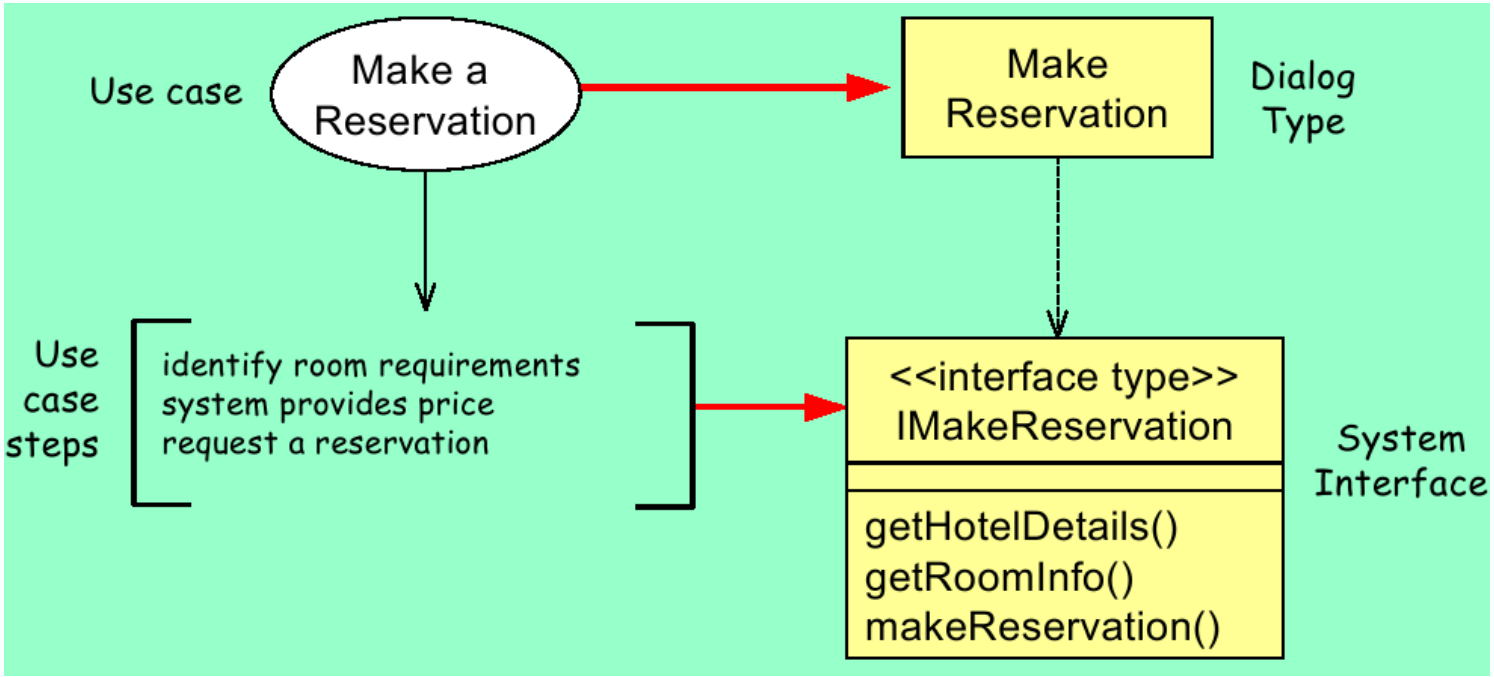
- Identificação de componentes a partir das interfaces do sistema (procedimento intuitivo);
- (i) identificação das interfaces; e (ii) agrupamento das interfaces em componentes.

Workflow de Identificação dos Componentes



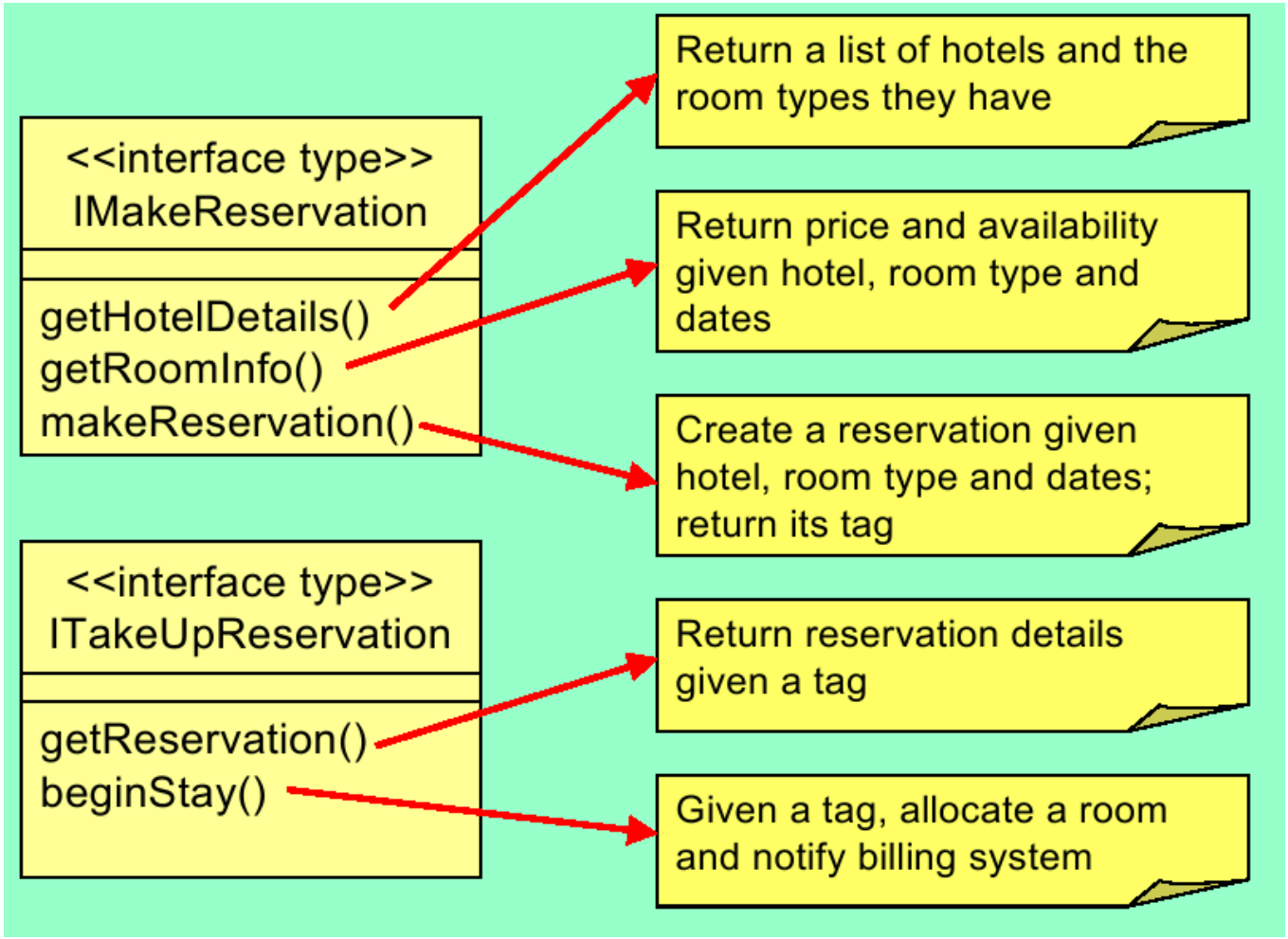
[Chesman & Daniels]

Interfaces e Operações de Sistema (I)



- A partir dos casos de uso especificados;
- Caso de uso ⇒ interface + operação;
- Passos do fluxo ⇒ podem ser agrupados em operações.

Interfaces e Operações de Sistema (II)



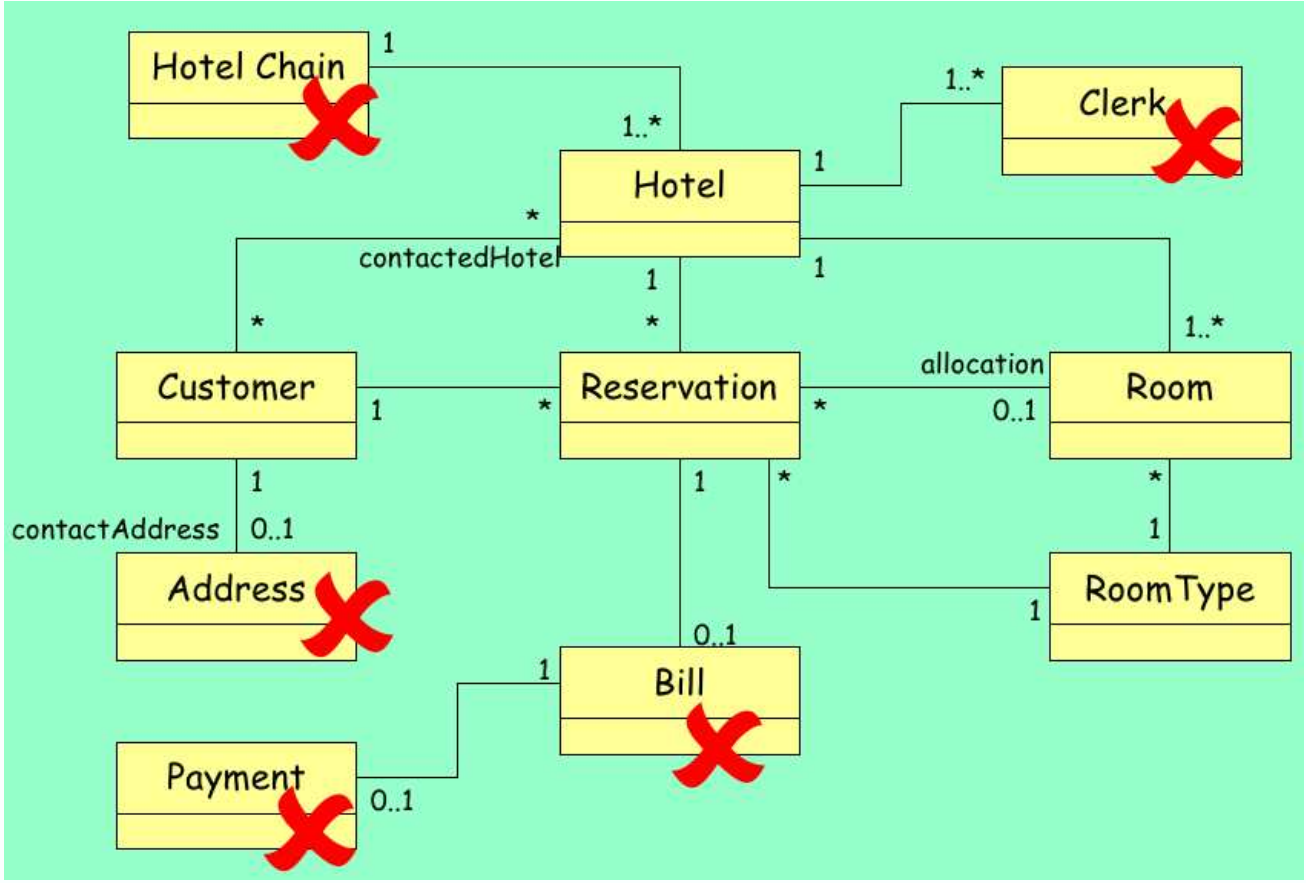
Modelo de Tipos do Negócio

(Business Type Model) (I)

- Derivado do modelo conceitual (*Business Concept Model*);
- Restringe as entidades relevantes para o domínio da solução;
- Pode ser visto como um "sub-modelo" conceitual do negócio que se refere aos casos de uso (responsabilidade do sistema) e não às responsabilidades exclusivas dos atores.

Modelo de Tipos do Negócio

(Business Type Model) (II)

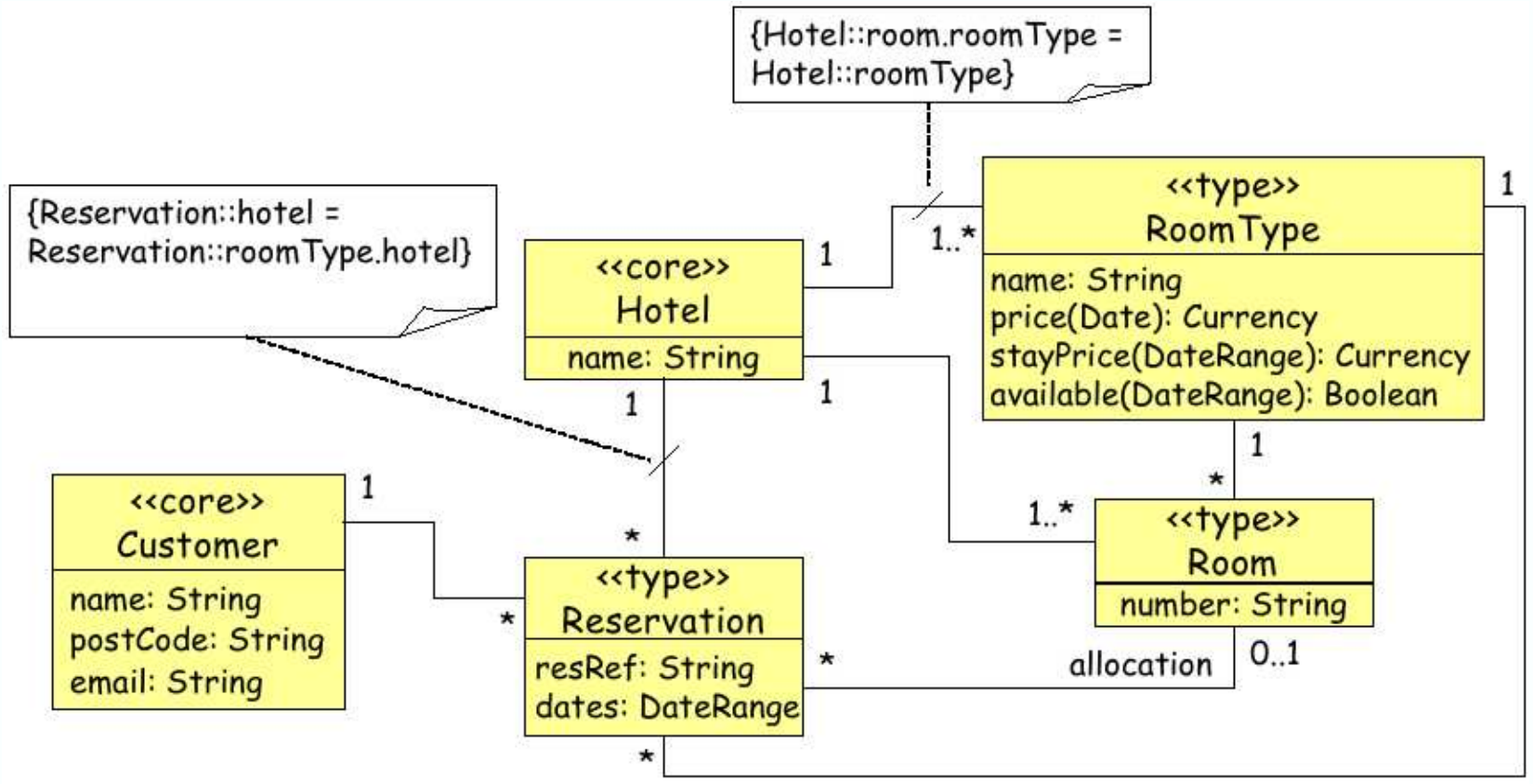


- Necessita classificar as entidades principais (*core types*) das entidades secundárias (*types*) do modelo.

Modelo de Tipos do Negócio

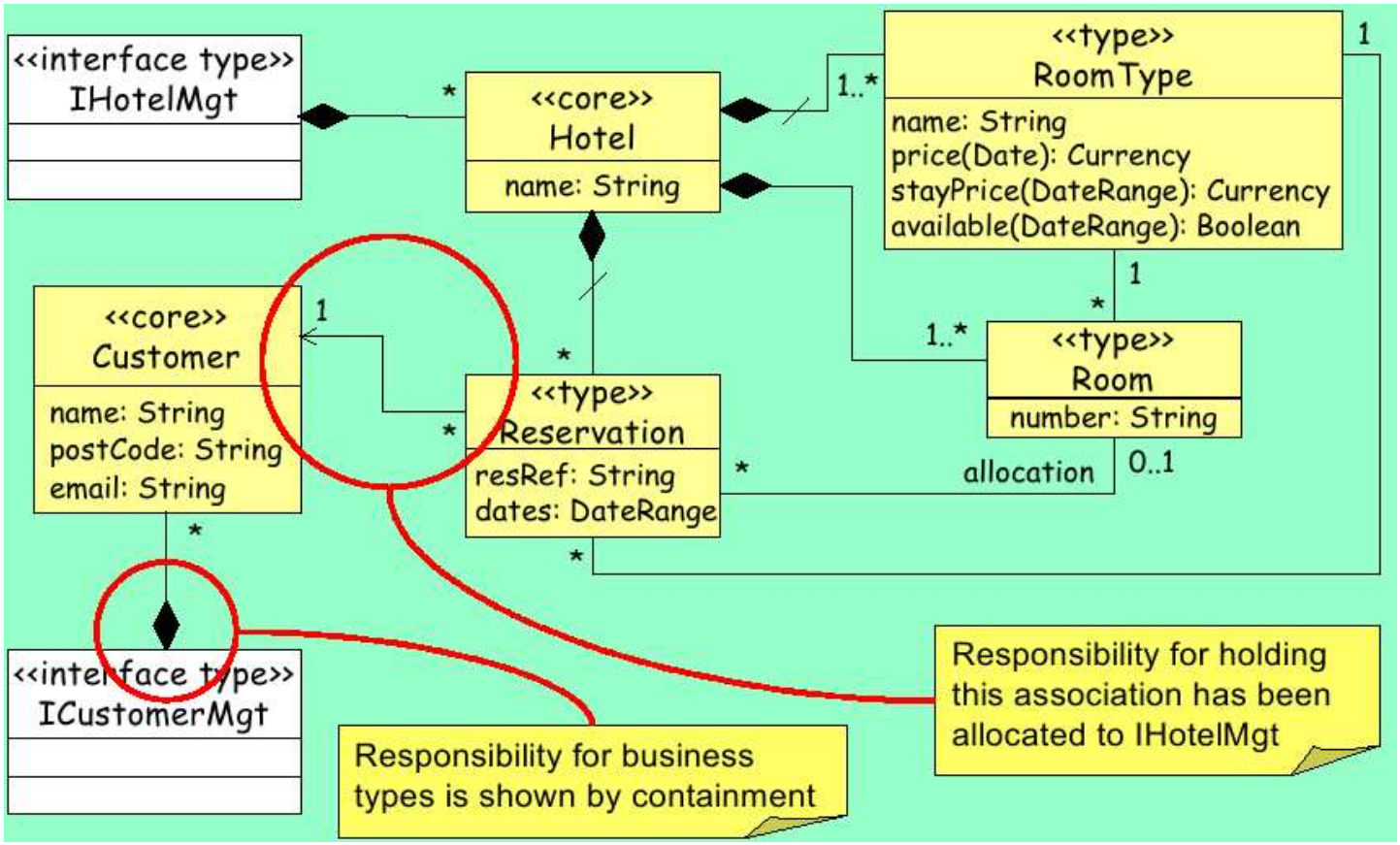
(Business Type Model) (III)

- Pode ser feita a seguinte analogia com o Modelo Entidade Relacionamento (MER) de banco de dados:
  - Entidade forte $\Rightarrow$ *core type*;
  - Entidade fraca $\Rightarrow$ *type*.
- *Core type*s normalmente se referem a entidades utilizadas pelos casos de uso (fluxos).

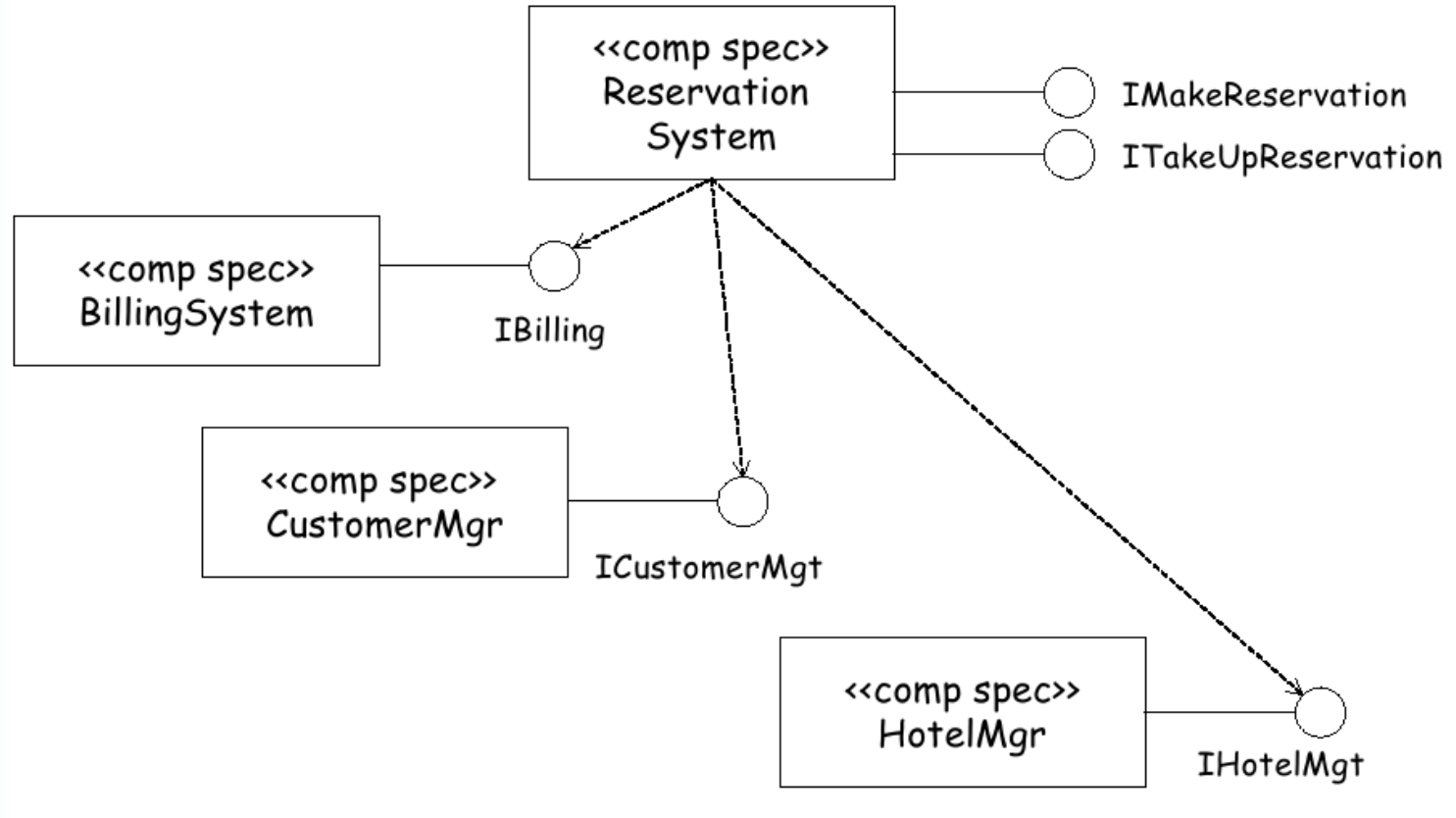

{Hotel::room.roomType = Hotel::roomType}
{Reservation::hotel = Reservation::roomType.hotel}
<<core>>
Hotel
name: String
<<type>>
RoomType
name: String
price(Date): Currency
stayPrice(DateRange): Currency
available(DateRange): Boolean
<<core>>
Customer
name: String
postCode: String
email: String
<<type>>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

Alocação de Responsabilidades e Identificação das Interfaces de Negócio

- A alocação de responsabilidade é necessária quando um *type* está associado a mais de um *core type*;
- Nesse ponto, as interfaces de negócio não possuem operações.

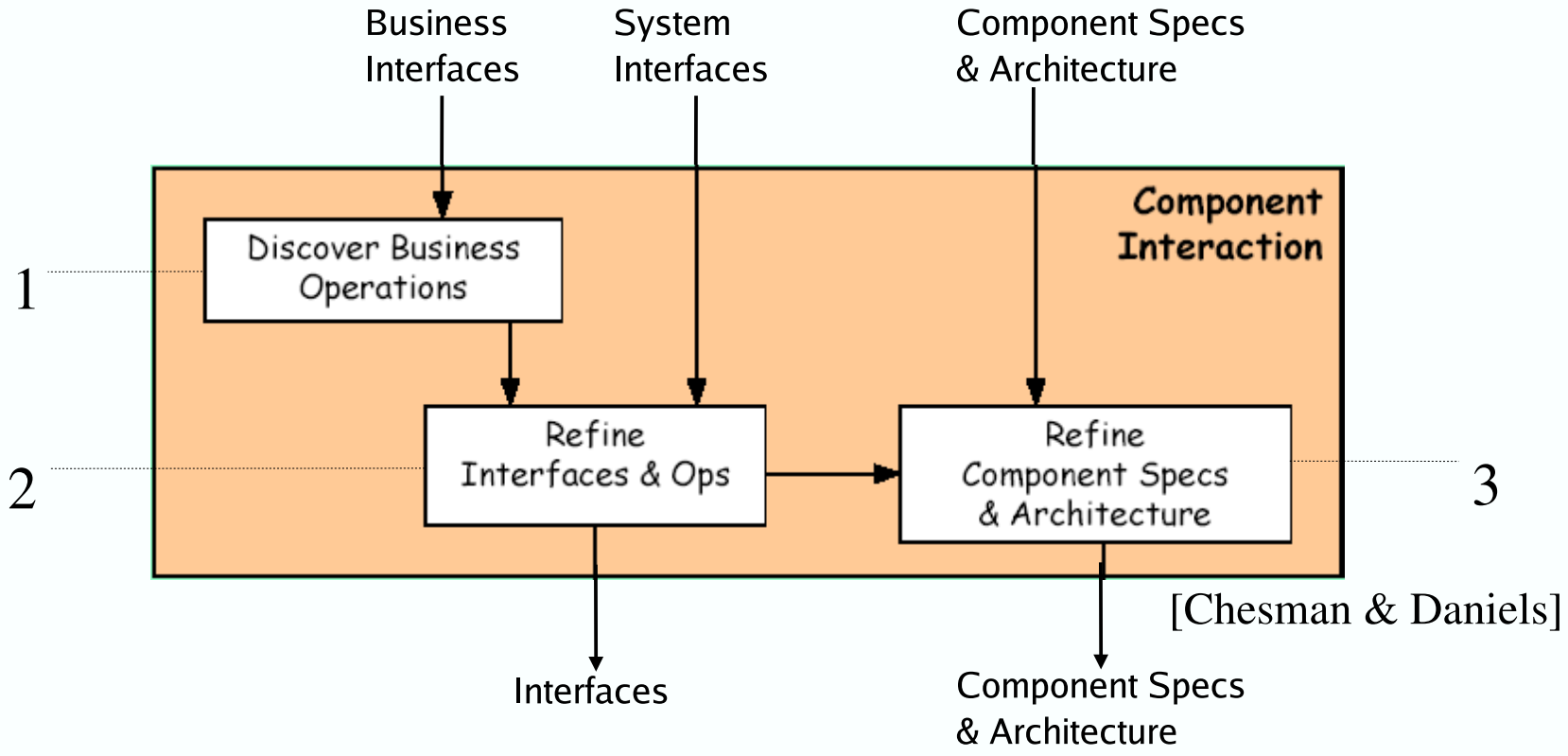

<<interface type>>
IHotelMgt
<<core>>
Hotel
name: String
<<type>>
RoomType
name: String
price(Date): Currency
stayPrice(DateRange): Currency
available(DateRange): Boolean
<<core>>
Customer
name: String
postCode: String
email: String
<<type>>
Reservation
resRef: String
dates: DateRange
<<type>>
Room
number: String
allocation
0..1
<<interface type>>
ICustomerMgt
Responsibility for business types is shown by containment
Responsibility for holding this association has been allocated to IHotelMgt

Arquitetura Inicial de Sistema



Interação Entre os Componentes

Workflow de Interação Entre os Componentes

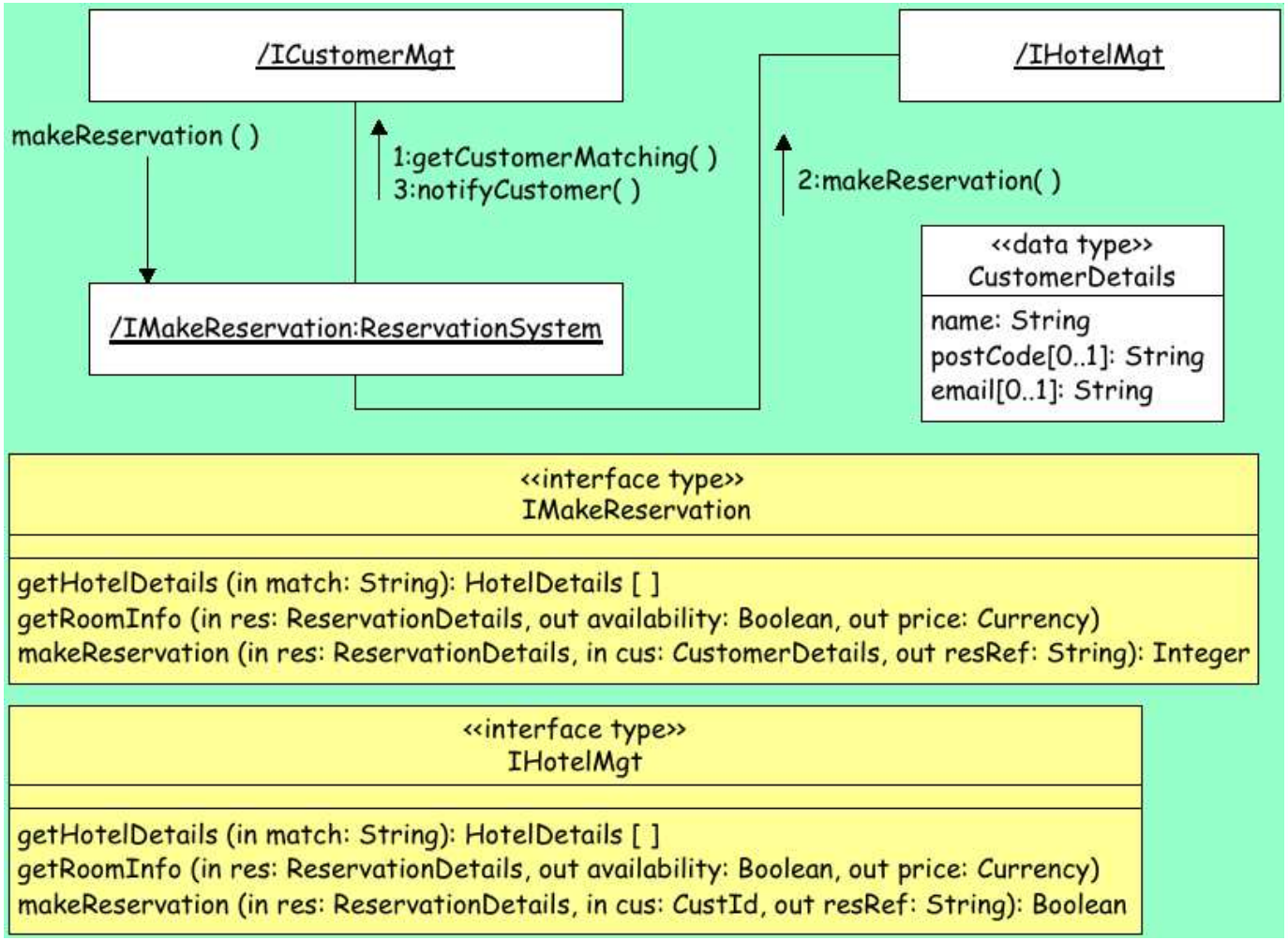


[Chesman & Daniels]

Descobrir Operações de Negócio

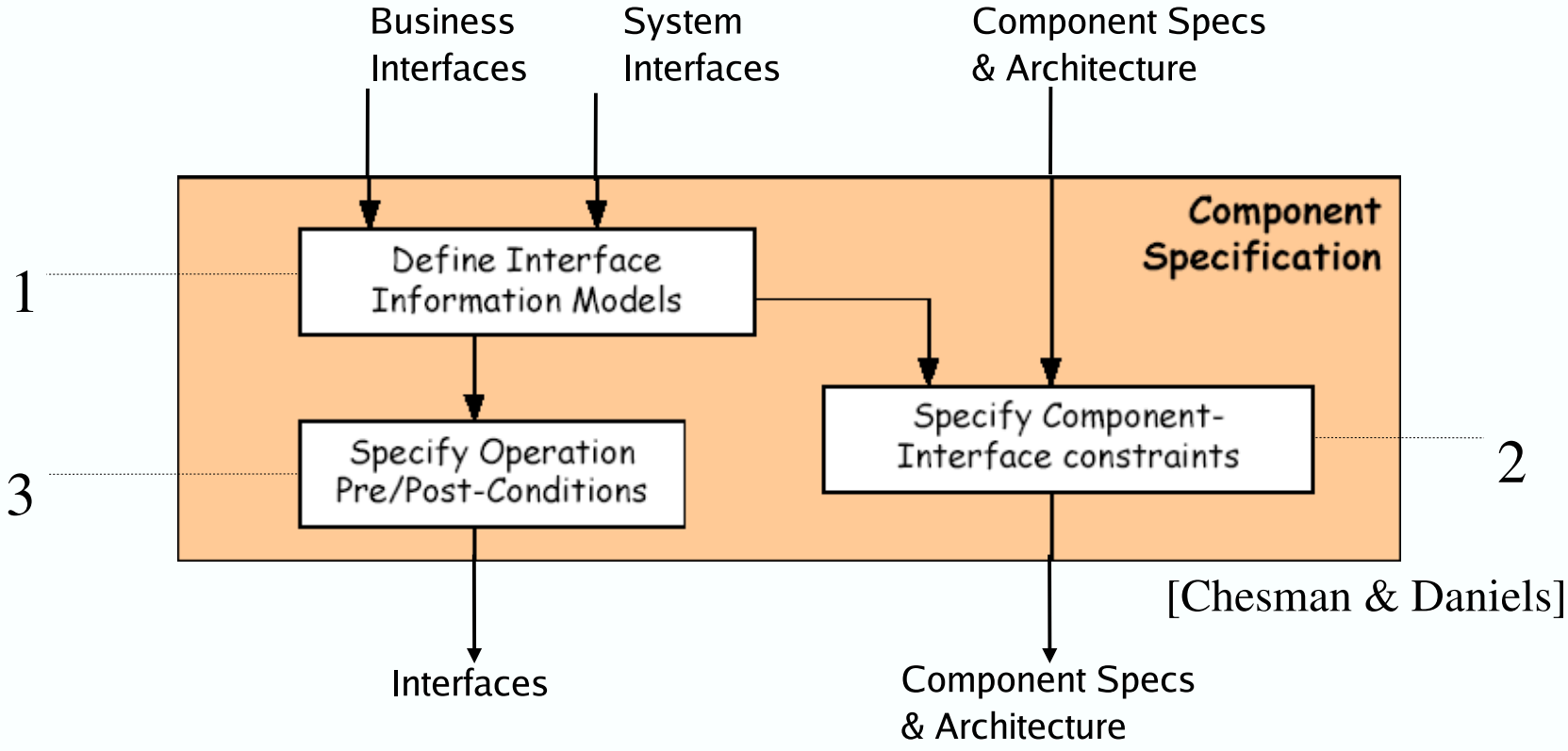
- Baseado na interação entre os componentes de sistema e de negócio;
- Construir diagramas dinâmicos (de colaboração, de seqüência ou de atividades) a partir dos fluxos especificados nos casos de uso;
- Descobrir operações das interfaces de negócio;
- Detalhar as assinaturas das operações das interfaces de sistema;
- Identificar os tipos de dados compostos que são utilizados ("entity beans").

Diagrama de Colaboração de Make Reservation



Especificação Final dos Componentes

Workflow de Especificação Final dos Componentes

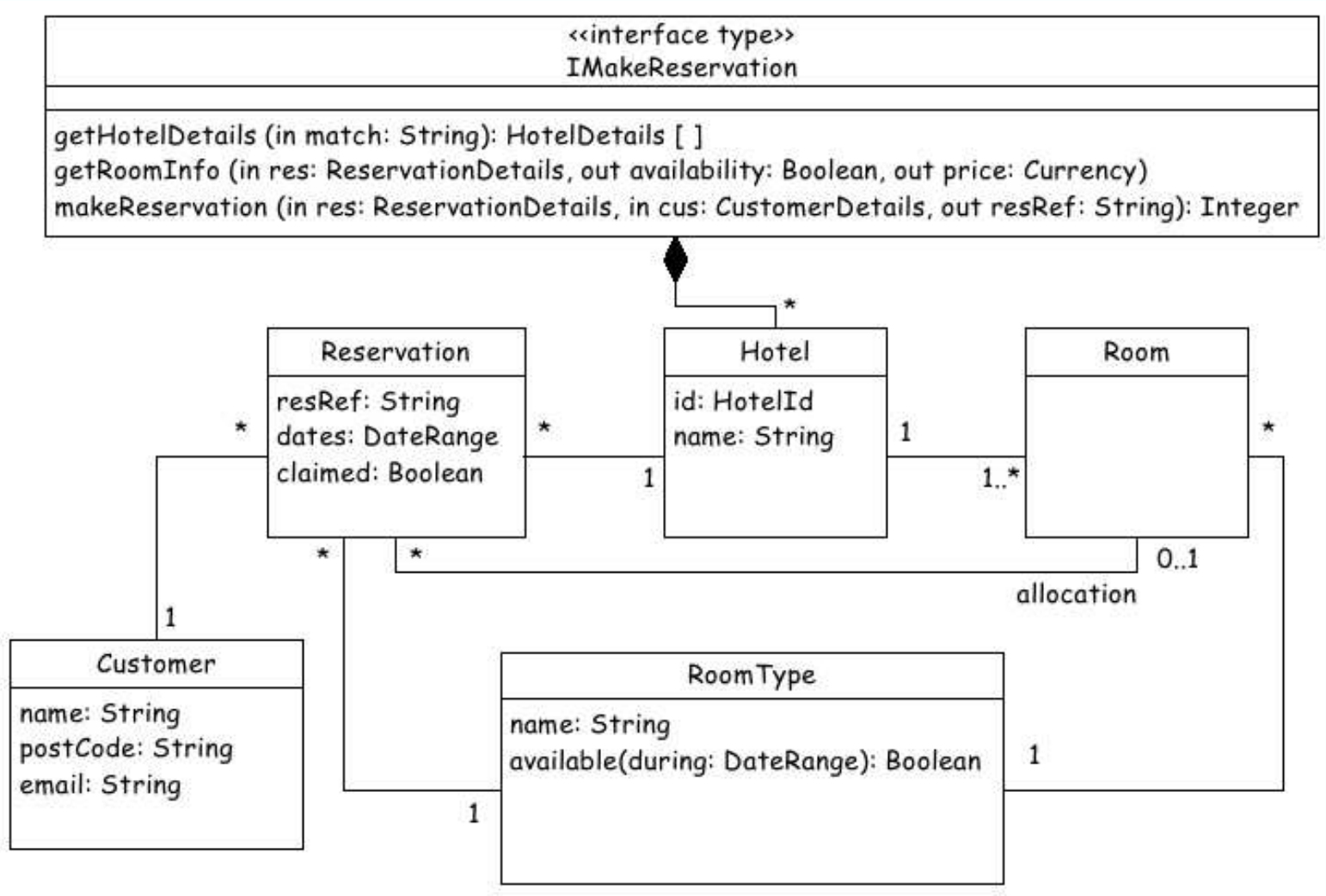


[Chesman & Daniels]

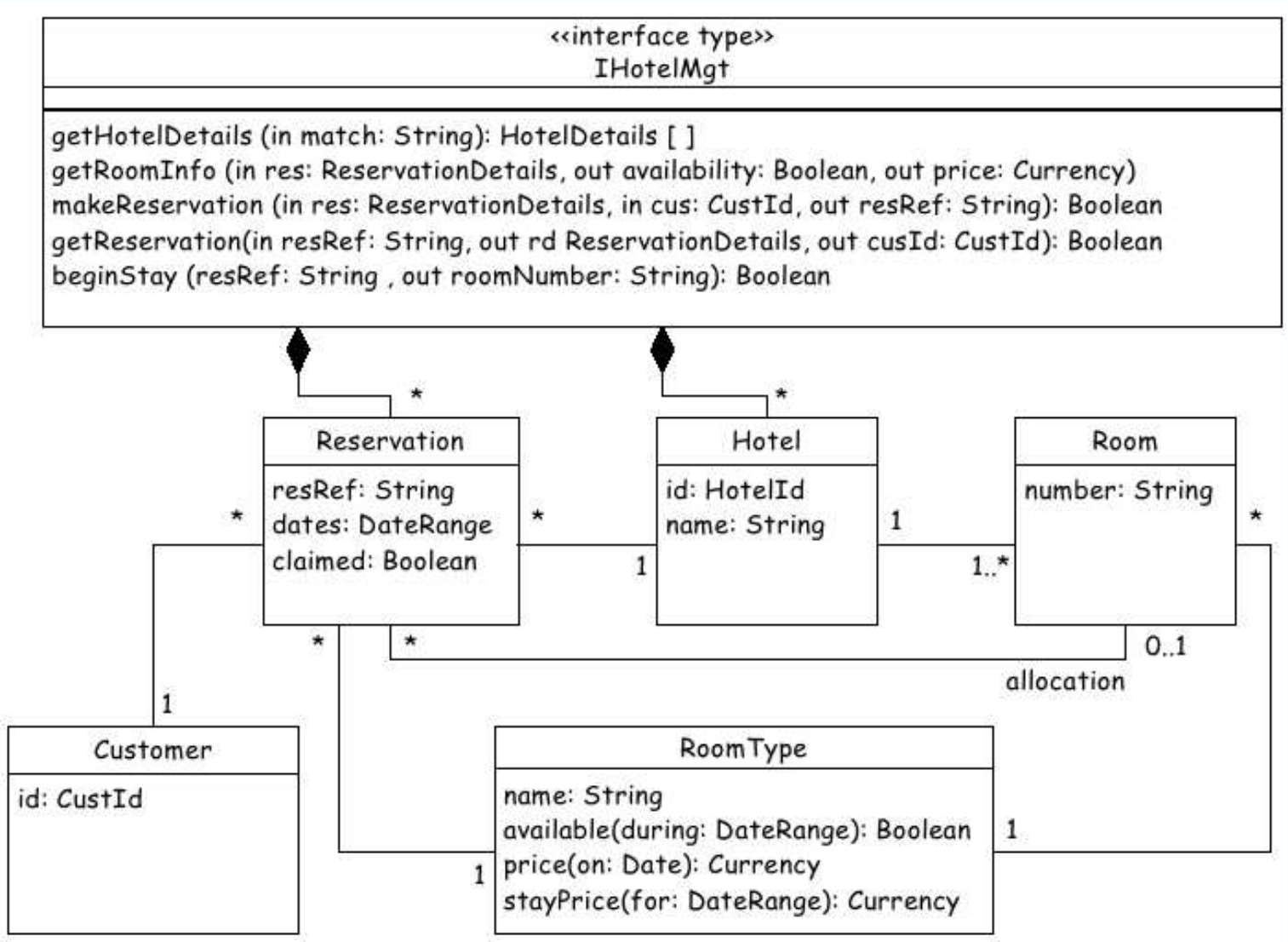
Modelo de Informação das Interfaces (Interface Information Model)

- Relação entre cada interface (de sistema e de negócio) e as entidades do modelo de tipos do negócio;
- Ajuda a entender o contexto de cada interface e auxilia na troca de conhecimento entre a equipe de desenvolvimento.

Modelo de Informação da Interface IMakeReservation


<<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
Reservation
resRef: String
dates: DateRange
claimed: Boolean
Hotel
id: HotelId
name: String
Room
Customer
name: String
postCode: String
email: String
RoomType
name: String
available(during: DateRange): Boolean
allocation
0..1
1..*
*
1

Modelo de Informação da Interface IHotelMgt


<<interface type>>
IHotelMgt
getHotelDetails (in match: String): HotelDetails [ ]
getRoomInfo (in res: ReservationDetails, out availability: Boolean, out price: Currency)
makeReservation (in res: ReservationDetails, in cus: CustId, out resRef: String): Boolean
getReservation(in resRef: String, out rd ReservationDetails, out cusId: CustId): Boolean
beginStay (resRef: String , out roomNumber: String): Boolean
Reservation
resRef: String
dates: DateRange
claimed: Boolean
Hotel
id: HotelId
name: String
Room
number: String
Customer
id: CustId
RoomType
name: String
available(during: DateRange): Boolean
price(on: Date): Currency
stayPrice(for: DateRange): Currency
allocation
0..1
1..*

Provisionamento dos Componentes (I)

- O processo UML Components lista as possíveis maneiras de se prover componentes de software, mas não detalha cada uma delas.

Provisionamento dos Componentes (II)

- Aquisição dos componentes especificados:
  - Reutilização de componentes prontos;
  - Implementação de novos componentes
- Para localização de componentes prontos:
  - Deve-se ter heurísticas de busca por serviço;
  - Pode-se levar em consideração a semelhança entre os *core types*.

Provisionamento dos Componentes (III)

- Para reutilizar os componentes:
  - Pode ser necessário adaptar os componentes reutilizados ou até mesmo as funcionalidades do sistema (re-negociação dos requisitos).
- Para a implementação:
  - Deve-se utilizar um modelo de componentes:

    EJB, CCM, COM+, COSMOS

Montagem do Sisitema (I)

- O processo UML Components destaca a importância e o papel desta fase, mas não detalha como as suas atividades podem ser executadas.

Montagem do Sisitema (II)

- Implementação dos conectores arquiteturais;
  - Componente responsável pela "cola" entre os componentes;
  - Preocupação com a adaptação do comportamento excepcional (*architectural mismatches*);
  - Implementação dos requisitos de qualidade:
    - Distribuição, disponibilidade, confiabilidade, escalabilidade, etc.
- Implementar a rotina de inicialização do sistema.
  - Programa principal;
  - "Liga" os componentes aos conectores.

Resumo (UML Components)

- Especificação dos requisitos (**independente de tecnologia**);
  - Modelagem da lógica do negócio.
- Especificação dos componentes (**independente de tecnologia**);
  - Descoberta das Interfaces de sistema (providas e requeridas);
  - Descoberta das Interfaces de negócio (providas).
- Provisionamento dos componentes (**dependente de tecnologia**);
  - Aquisição dos componentes (reutilização ou implementação).
- Montagem do sistema (**dependente de tecnologia**);
  - Implementação dos conectores;
  - Ligação entre os componentes e os conectores do sistema.

Resumo (DBC)

- OO e DBC são complementares;
  - Um componente pode ser desenvolvido usando OO.
- Componentes declaram explicitamente os seus serviços e as suas dependências:
  - Interfaces providas e requeridas.
- Os principais benefícios do conceito de DBC são a abstração da complexidade e o baixo acoplamento entre os módulos do sistema:
  - Outros benefícios são conseqüências desses.