

# Testes caixa preta - 2



## Tabela de Decisão Casos de Uso

Criação: Abr/2001

Reformulação: Abr/2013

# Referências

---

- M. Pezzè, M. Young. **Teste e Análise de Software.**  
Bookman Companhia Editora, 2008, cap. 10 e 11.
- P. Ammann, J. Offutt. **Introduction to Software Testing.**  
Cambridge University Press, 2008, cap.4.
- R.Binder. “Testing OO Systems. Models, Patterns and Tools”, Addison-Wesley, 2000, cap14

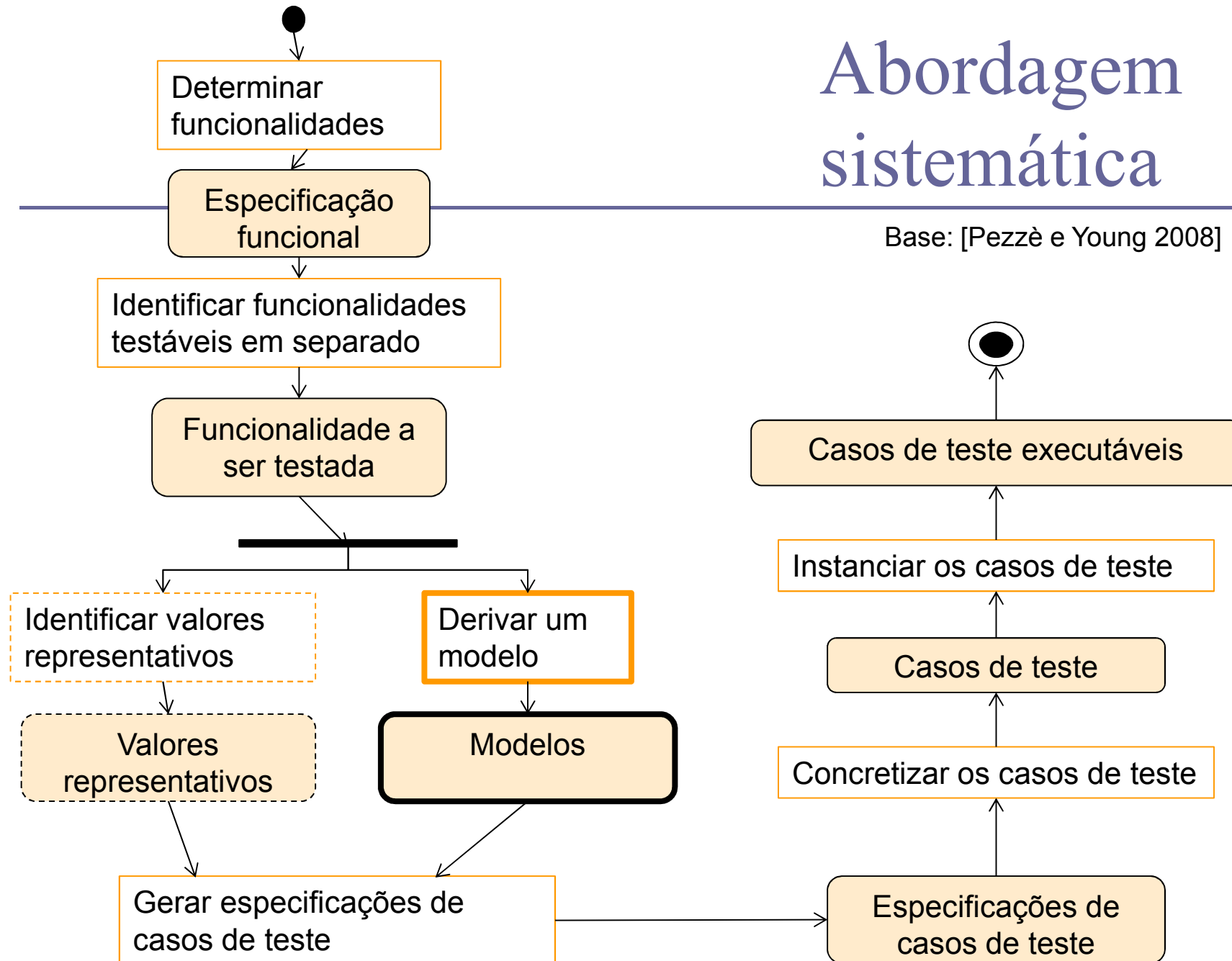
# Tópicos

---

- Visão Geral
- Abordagens
  - Árvore de decisão
  - Tabela de decisão
  - Gramática

# Abordagem sistemática

Base: [Pezzè e Young 2008]



# Algumas técnicas de testes caixa preta

---

Especificação:  
Requisitos  
Projeto

Abordagens combinatórias

Partição de equivalência

Valores Limite

Particionamento em categorias

Testes Aleatórios

Baseados em especificação estruturada

Árvore de decisão

Tabela de decisão

Gramática

# Motivação - 2

---

- Limitações das abordagens combinatórias:
  - Partição em classes de equivalência e Análise de Valores-Limite:
    - Não levam em conta combinações de valores → difícil testar situações em que diferentes combinações levam a diferentes saídas do sistema
  - Partição por categorias e Árvore de classificação
    - Visam as combinações de valores (com restrições)
    - Partem de especificações informais → não levam em conta modelos de especificação
- Como gerar casos de teste que levem em conta uma representação estruturada do sistema?

# Análise causa - efeito

---

- Útil quando especificações são representadas como estruturas de decisão:
  - Conjuntos de condições sobre valores de entradas e as ações correspondentes do sistema
- Modelos de teste:
  - árvores de decisão
  - tabelas de decisão
    - grafo causa-efeito como modelo auxiliar

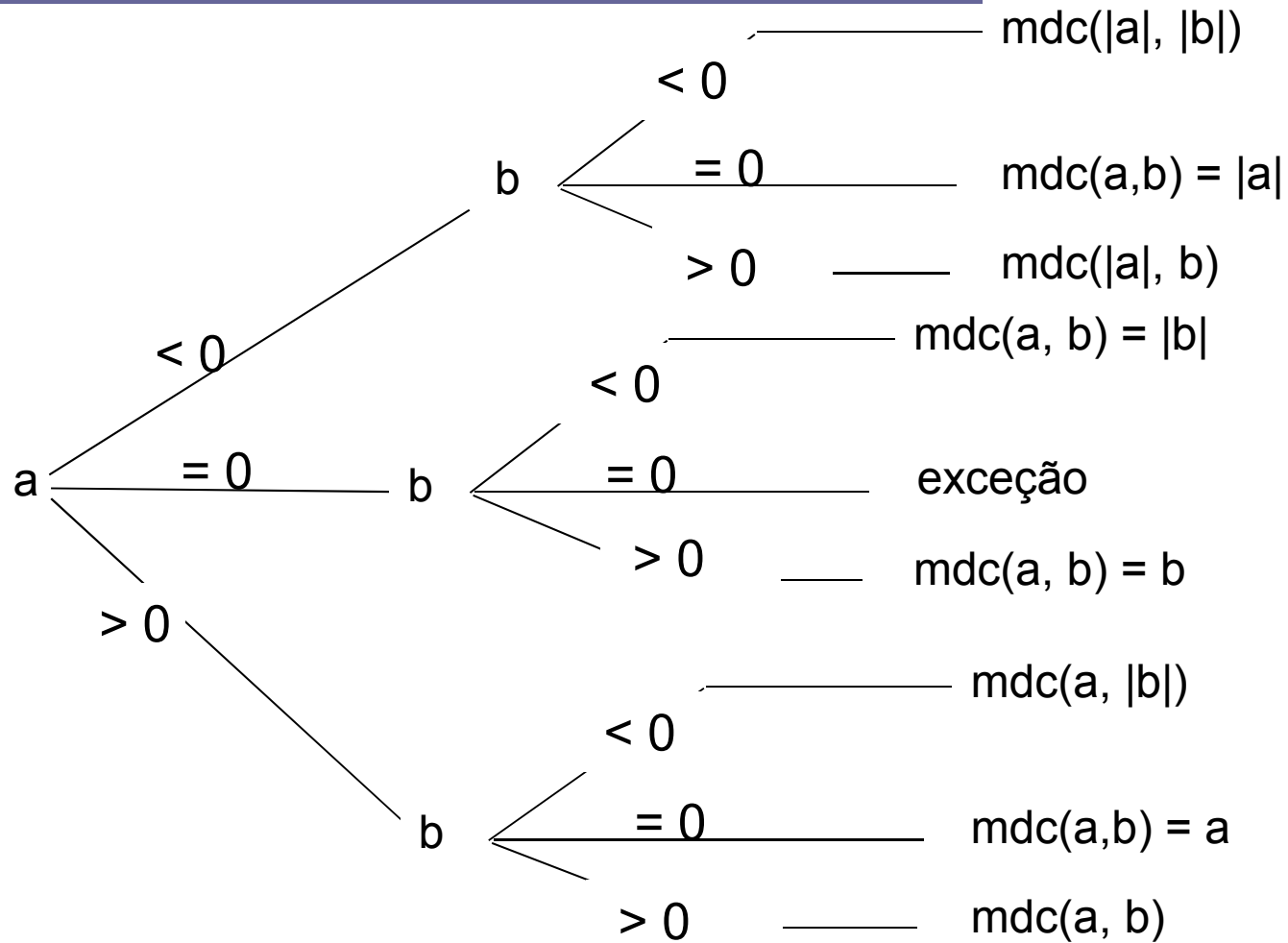
# Definições

---

- **Causas:**
  - condições de entrada (valor lógico)
- **Efeitos:**
  - ações realizadas em resposta às diferentes condições de entrada



# Árvore de decisão: exemplo do mdc (a, b)



# Tabela de decisão

---

c1	V	...	F																	
...																				
e1	X																			
...																				

Condições de entrada  
(causas)

Ações  
(efeitos)

regra



# Construção da tabela de decisão

---

$a \neq 0$ $b \neq 0$	V	V	F	F
	V	F	V	F
$\text{mdc}(a, b) = a$		✓		
$\text{mdc}(a, b) = b$			✓	
$\text{mdc}(a, b)$ exceção	✓		✓	

# Utilidade da tabela de decisão

---

- A tabela de decisão:
  - Facilita a determinação de quais testes aplicar.
  - Permite que se analise a especificação para determinar:
    - **Redundâncias**: duas regras iguais, i.e, mesmas causas levando aos mesmos efeitos
    - **Contradições**: duas regras com as mesmas causas levando a efeitos diferentes
    - **Omissões**: não há regras para todas as combinações de causas.
      - Redundâncias e contradições não são necessariamente erros: podem indicar concorrência.
      - Omissões podem indicar situações irrelevantes ou até mesmo impossíveis

→ é preciso fazer uma análise

# Limitação das tabelas de decisão

---

## □ Tamanho:

- 3 causas  $\Rightarrow 2^3$  combinações (regras)

- 5 causas  $\Rightarrow 2^5$  regras

- ...

- 8 causas  $\Rightarrow 2^8$  regras

- ...

## □ Será que vale a pena testar todas as regras?



# Exemplo

---

Supor um sistema bancário que trate somente duas transações:

depósito    n° da conta    quantia

saque        n° da conta    quantia

Requisitos:

- se o comando é depósito e o n° da conta é válido então a quantia é depositada
- se o comando é saque e o n° da conta é válido e a quantia é válida ( $0 < \text{quantia} \leq \text{saldo}$ ) então a quantia sacada
- se o comando ou n° da conta ou a quantia for inválido então exibir mensagem de erro apropriada


# Causas e efeitos

---

- Causas:

c1. Comando é depósito

c2. Comando é saque

c3. N° da conta é válido 

c4. Quantia é válida

n° de regras =  $2^4 = 16$

será que todas interessam ?

- Efeitos:

e1. Exibir “comando inválido”

e2. Exibir “n° da conta inválido”

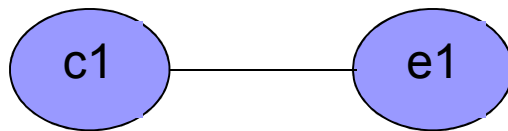
e3. Exibir “quantia inválida”

e4. Depositar a quantia

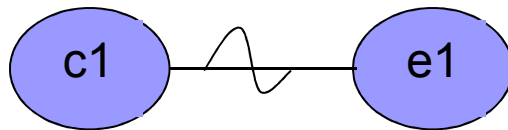
e5. Sacar a quantia

# Grafo causa-efeito: notação básica

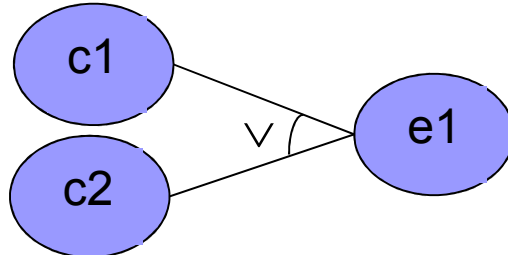
---



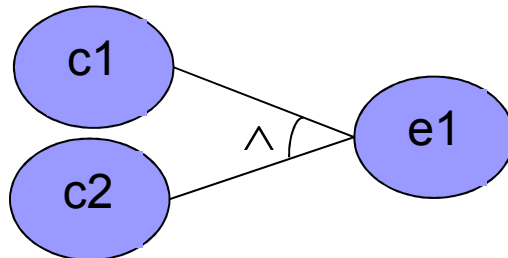
Identidade



Negação



Ou

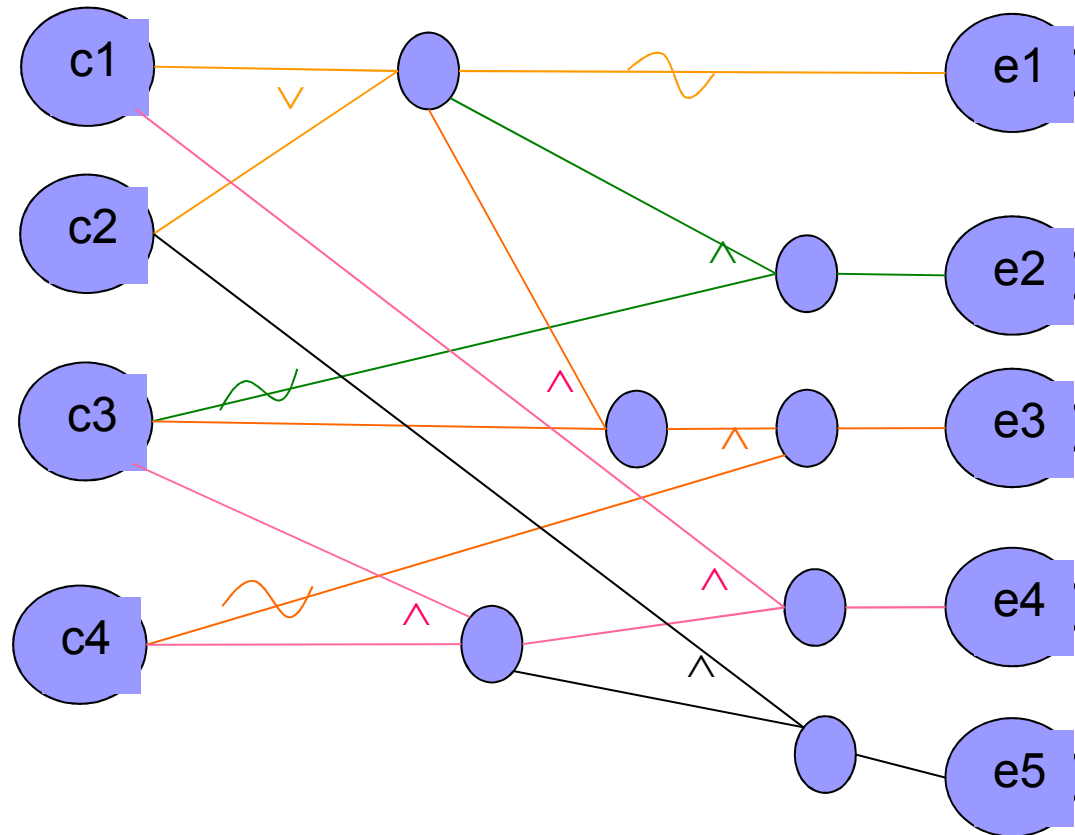


E



# Exemplo: grafo causa-efeito

- Causas:
  - c1. Comando é depósito
  - c2. Comando é saque
  - c3. N° da conta é válido
  - c4. Quantia é válida
- Efeitos:
  - e1. Exibir “comando inválido”
  - e2. Exibir “n° da conta inválido”
  - e3. Exibir “quantia inválida”
  - e4. Depositar a quantia
  - e5. Sacar a quantia



# Conversão em tabela de decisão

---

- ① Escolher um efeito como ação a ser executada, ie, marcar um “✓” na regra correspondente a este efeito.
- ② Rastrear no grafo quais as combinações de causas que levam a esse efeito e marcar um “V” ou “F” na posição correspondente na tabela
- ③ Para cada combinação criada, verificar se ocorrem ou não os outros efeitos

# Conversão: OU

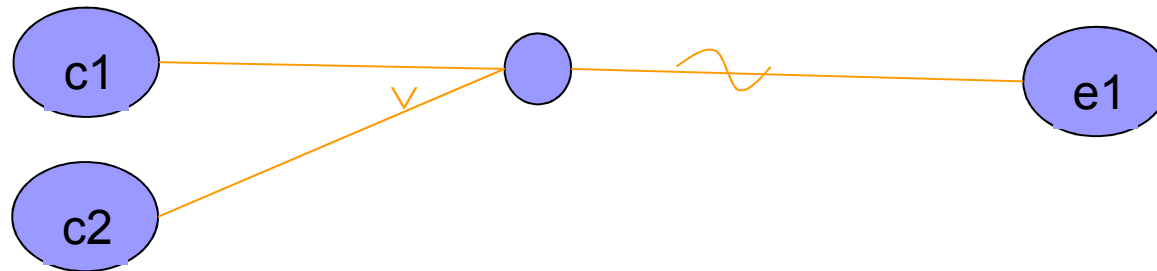
---

- Se  $e1 = x1 \vee x2$ :
  - não escolha  $x1 = x2 = V$
- Se  $e1 = \neg (x1 \vee x2)$ :
  - considere todas as combinações que façam com que  $x1 \vee x2 = F$

$x1$  e  $x2$  podem ser causas ou nós intermediários

# Exemplo: tabela de decisão

---



Id.	1	2	3	4	5
c1	F				
c2	F				
c3	x				
c4	x				
e1	✓				
e2					
e3					
e4					
e5					

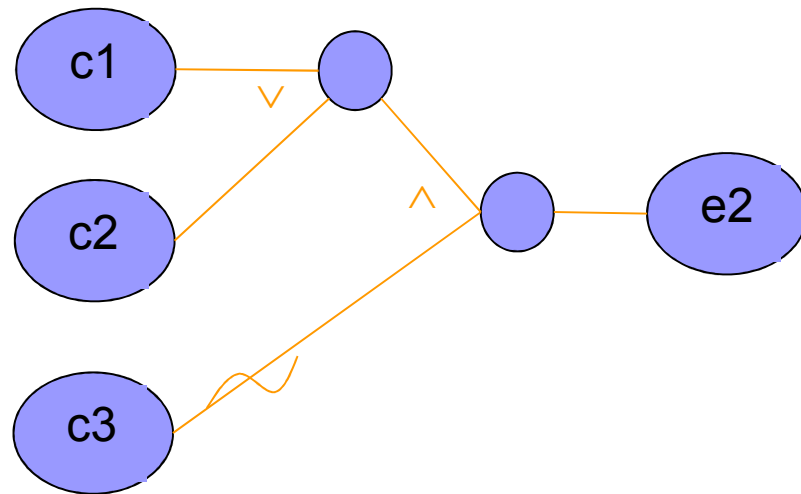
# Conversão: E

---

- Se  $e1 = x1 \wedge x2$ :
  - considere todas as combinações que façam com que  $x1 = x2 = V$
- Se  $e1 = \neg (x1 \wedge x2)$ :
  - considere somente uma combinação que faça com que  $x1 \wedge x2 = F$
  - para a combinação escolhida inclua uma e somente uma combinação que leve ao resultado desejado

$x1$  e  $x2$  podem ser causas ou nós intermediários

# Exemplo: tabela de decisão



Id.	1	2	3	4	5	6
7						
c1	F	V	F			
c2	F	F	V			
c3	x	F	F			
c4	x	x	x			
e1	✓					
e2		✓	✓			
e3						
e4						
e5						

# Exemplo: tabela de decisão

---

Id.	1	2	3	4	5	6	7
c1	F	V	F	V	F	V	F
c2	F	F	V	F	V	F	V
c3	x	F	F	V	V	V	V
c4	x	x	x	F	F	V	V
e1	✓						
e2		✓	✓				
e3						✓	✓
e4					✓		
e5							✓

# Geração de testes

---

- Tabela de decisão

critério: exercitar cada regra pelo menos 1 vez

- Árvore de decisão

critério: exercitar cada caminho da raiz até a folha pelo menos 1 vez

- ☞ Eliminar os casos de teste que não fazem sentido ou que são redundantes.

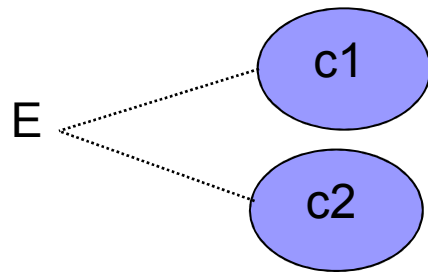


# Exemplo: casos de teste

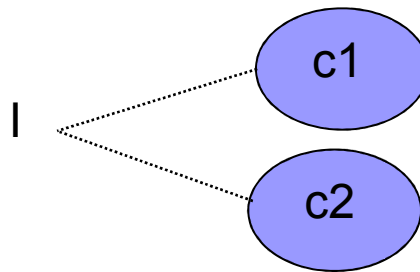
---

- **Regra 1:**  
comando  $\notin$  {depósito, saque},  $\forall$  nº conta,  $\forall$  quantia
- **Regra 2:**  
comando = depósito, nº de conta inválido,  $\forall$  quantia
- **Regra 3:**  
comando = saque, nº de conta inválido,  $\forall$  quantia
- ...

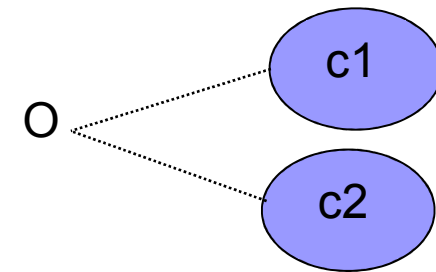
# Restrições



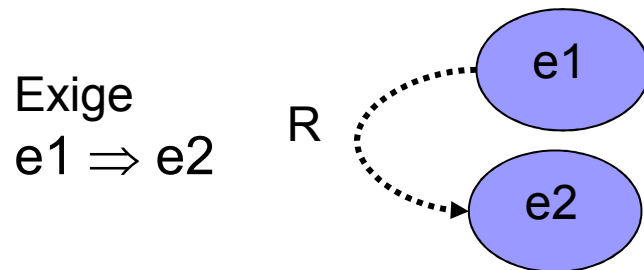
Exclusivo  
no máximo 1  
(0+)



Inclusivo  
no mínimo 1  
(1+)

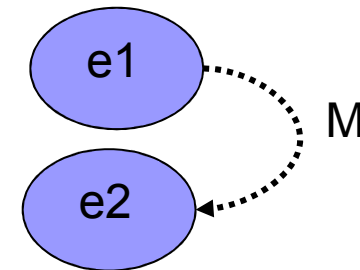


Somente um  
1 e somente 1  
(1)

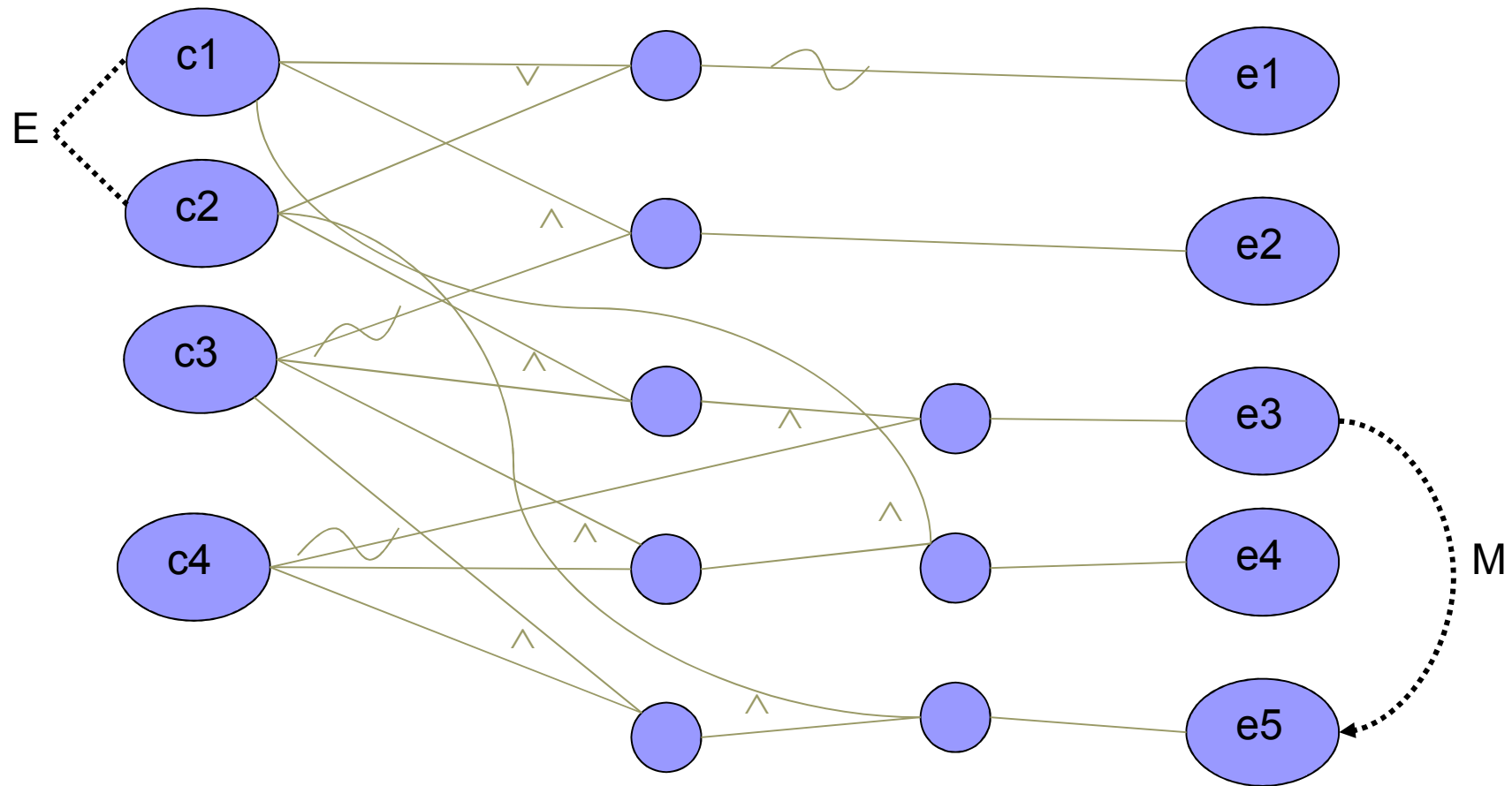


Exige  
 $e1 \Rightarrow e2$

Mascara  
 $e1 \Rightarrow \neg e2$



# Exemplo: uso de restrição



# Exercício 1

---

Considere uma função com duas variáveis de entrada:

Cliente e Qtd, e uma variável de saída, Desconto.

Cliente pode ser do tipo A, B ou C e Qtd pode variar de 1 a 1000. A função calcula Desconto de acordo com as seguintes regras:

- Clientes do tipo A não recebem desconto se n° de itens comprados for inferior a 10; recebem 5% desconto para compras entre 10 e 99 itens; 10% de desconto acima de 100 itens.
- Clientes do tipo B recebem 5% de desconto para compras abaixo de 10 itens; 15% de desconto entre 10 e 99 itens; 25% de desconto acima de 100 itens.
- Clientes do tipo C não recebem desconto se n° de itens comprados for inferior a 10; 20% de desconto entre 10 e 99 itens; 25% de desconto acima de 100 itens.

# Exercício 2

---

Considere a tela de login em um sistema mostrada ao lado. O usuário deve fornecer:

- **login:** código alfanumérico de 8 caracteres. Se o código é inválido ou não é reconhecido pelo sistema, este solicita ao usuário que o forneça novamente, até que um código válido seja fornecido.
- **senha:** código alfanumérico de 5 caracteres. Se a senha é incorreta, o usuário tem uma chance a mais para fornece-la. Se ambas as tentativas falharem, o usuário deve recomeçar todo o processo.



The image shows a screenshot of a login window titled "Sistema - Tela de Entrada". The window has a light blue background and a dark blue title bar with standard window controls (minimize, maximize, close). The login form consists of two sections: "Login" and "Senha". Each section has a text input field and two buttons labeled "Entrar" and "Cancelar".

# Testes de sintaxe

---

- Uso de gramáticas para geração de testes
- Gramáticas são adequadas para representar:
  - Entradas de tamanho variável e não limitado
  - Estruturas recursivas
  - Condições-limite
- Exemplos:
  - Entradas textuais complexas
  - Árvores
    - ex.: documentos XML e HTML são árvores descritas textualmente
  - Estrutura de programas
    - Também podem ser consideradas como árvores descritas textualmente
    - Útil para testar compiladores

# Exemplo de gramática

Símbolo inicial

Símbolo não terminal

stream ::= action\*

action ::= actG | actB

actG ::= "G" s n

actB ::= "B" t n

s ::= digit<sup>1-3</sup>

t ::= digit<sup>1-3</sup>

n ::= digit<sup>2</sup> "." digit<sup>2</sup>

digit ::= "0" | "1" | "2" | "3" | "4" | "5"

produção  
ou regra

Símbolo terminal

Cadeias válidas

```
G 17 03.01
B 13 15.20
G 1 04.23
B 123 45.34
```

- : zero ou mais repetições
- m-n: no mínimo m e no máximo n repetições
- n: exatamente n repetições

# Testes de Sintaxe - critérios

---

- Casos de teste = *cadeias* geradas a partir da gramática
- Alguns critérios:
  - **Cobertura de produções**: um caso de teste deve exercitar pelo menos uma produção
  - **Cobertura de terminais**: um caso de teste deve conter pelo menos um terminal
  - **Condições-limite**: casos de teste devem exercitar cada produção recursiva:
    - Número mínimo de vezes
    - Número mínimo + 1
    - Número máximo - 1
    - Número máximo de vezes



# Exemplo de derivação de testes

---

Cobertura de produções

**stream** → **action**<sup>2</sup> ← nro estabelecido de repetições  
→ **action action**  
→ actB **action**  
→ G s n **action**  
→ G **digit**<sup>1-3</sup> **digit**<sup>2</sup> "." **digit**<sup>2</sup> **action**  
→ G **digitdigitdigit digitdigit.digitdigit action**  
→ G 143 21.01 **action**  
...

# Ainda a geração de testes de sintaxe

---

- Cobertura probabilística
  - Pode-se associar probabilidades às produções, para indicar qual produção selecionar a cada passo
  - Prioriza produções mais utilizadas
- Geração de dados inválidos:
  - Obtidos simplesmente aplicando-se mutações às produções ou aos terminais
  - Objetivo: determinar se o programa reage adequadamente a entradas inválidas
  - Ex.:
    - Mutação de produção: B 123 45.34 → 15 123 45.34
    - Mutação de terminal: B 123 45.34 → B 123 01.34

# Ferramentas para testes caixa preta

---

- O site abaixo contém várias ferramentas open source:
  - <http://www.opensourcetesting.org/functional.php>
    - Último acesso: set/2010

# Exercício - 3

- UF em teste: Busca por elementos na base de dados de um site.
- A chave de busca é uma cadeia de caracteres que pode ser:
  - Uma **cadeia simples**, i.e., uma seqüência simples de caracteres
  - Uma **cadeia composta**:
    - Uma cadeia terminada com “\*”, usado como coringa ou
    - Uma string composta de sub-cadeias entre chaves e separadas por “,”, usadas para indicar alternativas
  - Uma **combinação de cadeias**, i.e, cadeias combinadas com operadores booleanos AND, OR, NOT, que podem estar entre parênteses para alterar a prioridade dos operadores

- Exemplos de chaves de busca:
  - Laptop → a UF busca por essa cadeia
  - {DVD\*, CD\*} → busca por cadeias começando por DVD ou CD
  - NOT (A2010) AND A20\* → busca por cadeias começando por A20, exceto a sub-cadeia A2010

- Escreva a gramática que representa uma chave de busca
- Derive casos de teste que cubram todas as produções

# Algumas técnicas de testes caixa preta

---

Especificação:  
Requisitos  
Projeto

Abordagens combinatórias

Partição de equivalência

Valores Limite

Particionamento em categorias

Testes Aleatórios

Baseados especificação estruturada

Árvore de decisão

Tabela de decisão

Gramática

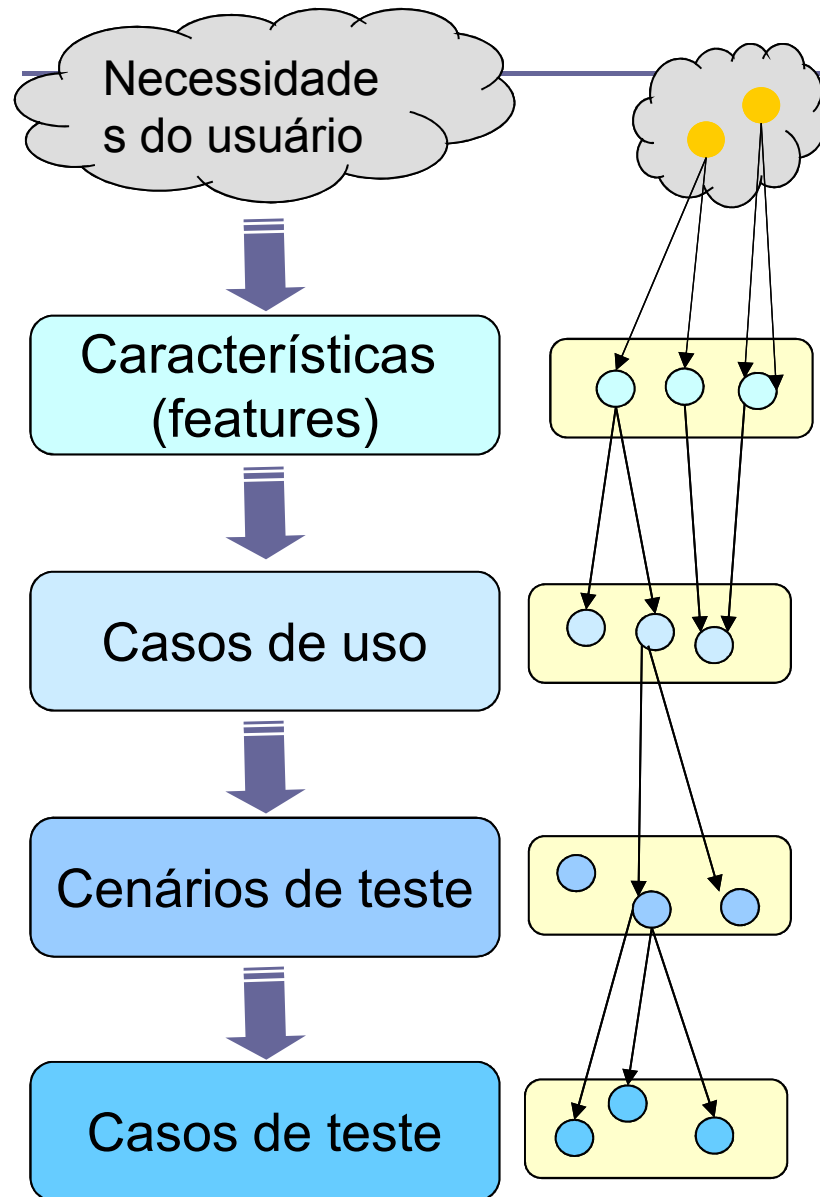
Casos de uso

# Testes a partir de casos de uso

---

- Motivação:
  - Casos de uso permitem representar o sistema do ponto de vista de seus usuários
  - Casos de uso podem ser usados em vários níveis de abstração, podendo representar:
    - requisitos funcionais (ou não funcionais) de um sistema ou incremento
    - interfaces de componentes ou objetos
    - interfaces com usuários
  - Casos de uso são descritos por fluxos (normais, alternativos e de exceção) → diferentes cenários → cenários  $\approx$  casos de teste
  - Facilidade de manter rastreabilidade entre os testes e os requisitos

# Relação entre requisitos, casos de uso e casos de testes



Fonte: Peter Zielczynski , obtido em out/2010

# Matriz de rastreabilidade dos testes

---

	CT1	CT2	CT3	...	CT1500
caso uso 1	✓	✓			
caso uso 2		✓	✓		
caso uso 3	✓		✓		✓
...					
caso uso 99	✓	✓			✓

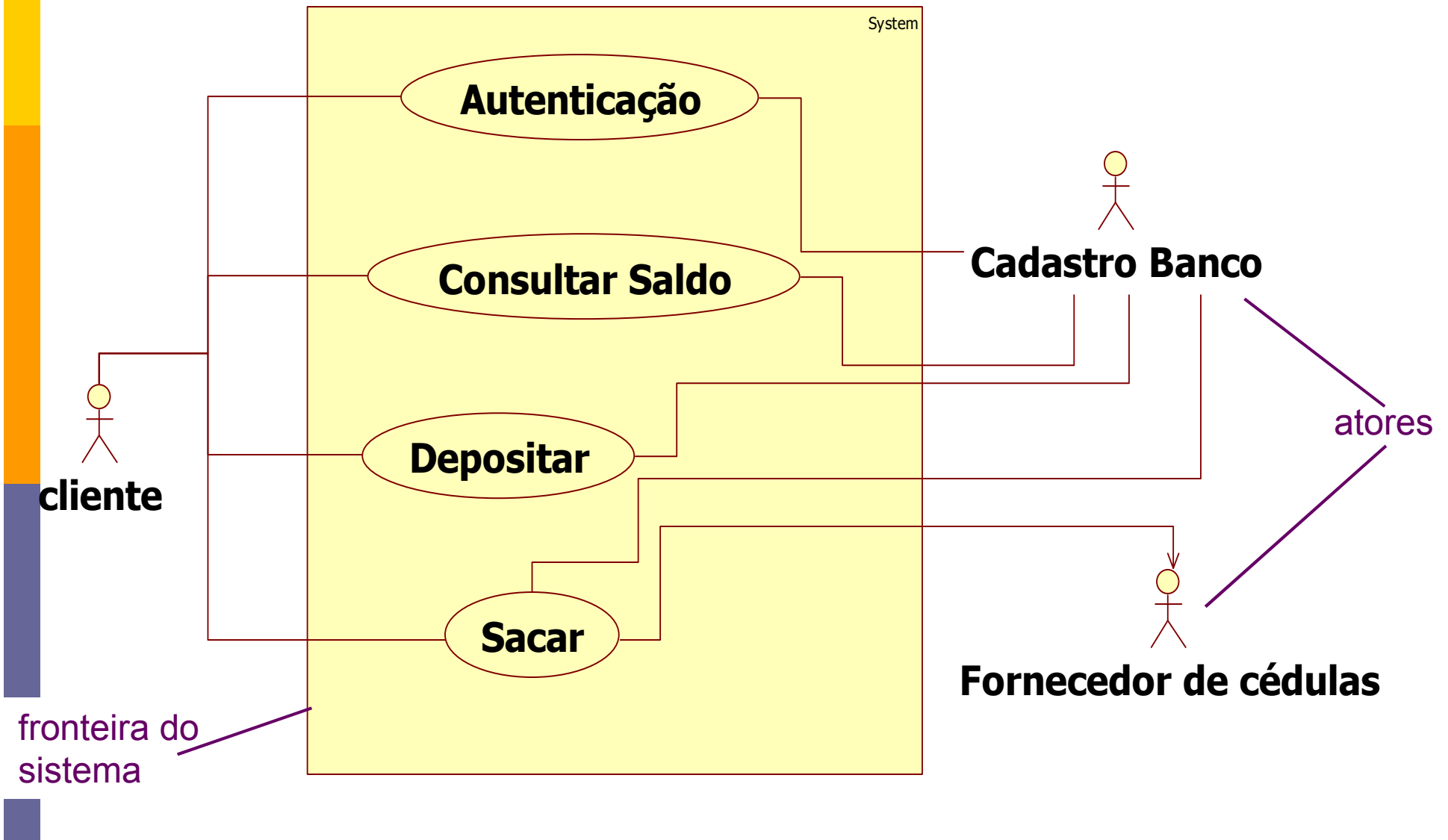


# Características

---

- Casos de uso representam o diálogo entre um sistema e **atores** externos
- Uma **instância** de um caso de uso define valores de entrada e saída específicos
- Um caso de uso é composto de **operações** iniciadas por um estímulo externo
  - uma operação = cenário de uso do sistema

# Exemplo - Diagrama de Caso de Uso



# De onde derivar os casos de teste?

---

- Diagramas de casos de uso
- Descrição dos casos de uso
  - inclui outros diagramas, como os de interação

mas ...

... a maioria das metodologias OO não exigem que os casos de uso possuam informações necessárias para os testes:

- Quais as entradas? Qual o domínio de cada entrada?
- Quais as saídas? Qual o domínio de cada saída?
- Em que condições os fluxos normais, alternativos e de exceção são usados?
- Qual a seqüência de execução dos casos de uso?

# Testes baseados em casos de uso estendidos

---

[Binder00, 14.2]

- Baseiam-se em **casos de uso estendidos** = caso de uso + variáveis operacionais
- Procedimento:
  - ① Identificar as variáveis operacionais
  - ② Definir os domínios das variáveis operacionais
  - ③ Estabelecer as relações operacionais
  - ④ Derivar os casos de teste

# Testes baseados em casos de uso

## ① Identificar as **variáveis operacionais**:

Fatores que variam de um cenário para outro e determinam as diferentes respostas do sistema:

- ▣ entradas e saídas explícitas (na interface do caso de uso)
- ▣ condições ambientais que afetem o comportamento dos atores
- ▣ abstrações do estado do sistema em teste

Exemplo: caixa eletrônico

Caso de uso	Ator	Cenários
Estabelecer sessão	cliente	(1) Senha inválida; corrige senha; mostra menu (2) senha OK; sem conexão com banco; mostra: “Tente mais tarde”
Sacar	cliente	(1) Pede R\$50; conta aberta; saldo de R\$1000; fornece os R\$50
Suprir dinheiro	operador	(1) abrir o caixa; inserir R\$15000; fechar o caixa

# Testes baseados em casos de uso

---

- ② Definir domínios das variáveis operacionais:
  - estabelecer valores válidos e inválidos para cada variável
  
  - para o caso de uso Autenticação:

Variáveis	Domínio
nº do cartão (NC)	entre 0000 e 9999
senha (S)	entre 0000 e 9999
conexão com o banco(CB)	{no ar, fora do ar}
status da conta (SC)	{aberta, fechada}

# Testes baseados em casos de uso

## ③ Estabelecer as relações operacionais:

- estabelecer combinações de valores das variáveis que levem a diferentes respostas do sistema ⇒ **tabela de decisão**

- para o exemplo dado:

↪ **variante** de um caso de uso

NC ∈ [0000 .. 9999]	F	V	V	
S ∈ [0000 .. 9999]		—	F	V
CP = no ar	—	—	F	
SC = aberto	—	—	—	
msg.: cartão inválido	✓			...
ejetar cartão	✓	✓	✓	
msg.: senha inválida		✓		
msg.: selecione transação				
msg.: tente mais tarde			✓	
msg.: conta fechada				

# Testes baseados em casos de uso

---

- ④ Derivar os casos de teste:
  - **Critério**: cada variante deve ser exercitada pelo menos uma vez
    - **Requisitos de testes**: conjunto de variantes
    - **Conjunto de testes adequado**: pelo menos 1 caso de teste para cada variante
  
  - para o exemplo dado:
    - ... completar



# Testes baseados em casos de uso

---

- Condições de entrada:
  - casos de uso estendido estão disponíveis
  - o sistema ou componente é minimamente operacional, i.e., já passou por testes que demonstraram que pode ser usado
  
- Condições de saída dos testes:
  - todos os casos de uso selecionados foram testados pelo menos uma vez
  - para cada caso de uso selecionado, cada variante foi exercitada pelo menos uma vez
  - matriz de rastreabilidade dos testes

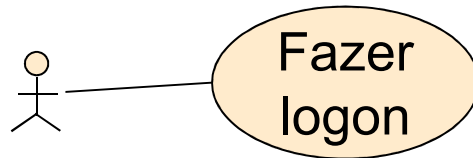
# Considerações sobre a abordagem

---

- ☺ Estabelece critério para geração de testes a partir de casos de uso
- ☹ O que fazer quando há relação de precedência entre os casos de uso?
  - A técnica baseada nos casos de uso estendidos não deixa explícito o que fazer nesse caso
- ☹ O que fazer quando fluxos internos a um caso de uso são complexos?
  - A técnica não dá diretrizes explícitas quanto a esse ponto também.
  - Recomenda desenvolver diagramas de estado representando o comportamento.

# Exercício 1: Derivar casos de teste para o caso de uso a seguir.

Diagrama de caso de uso



Protótipo da interface

A screenshot of a web login form titled 'Logon'. It features a blue header bar with the title and window control icons. Below the header, there are two input fields: 'Username' and 'Password'. At the bottom, there is a 'Remember me!' checkbox and a yellow 'Logon' button.

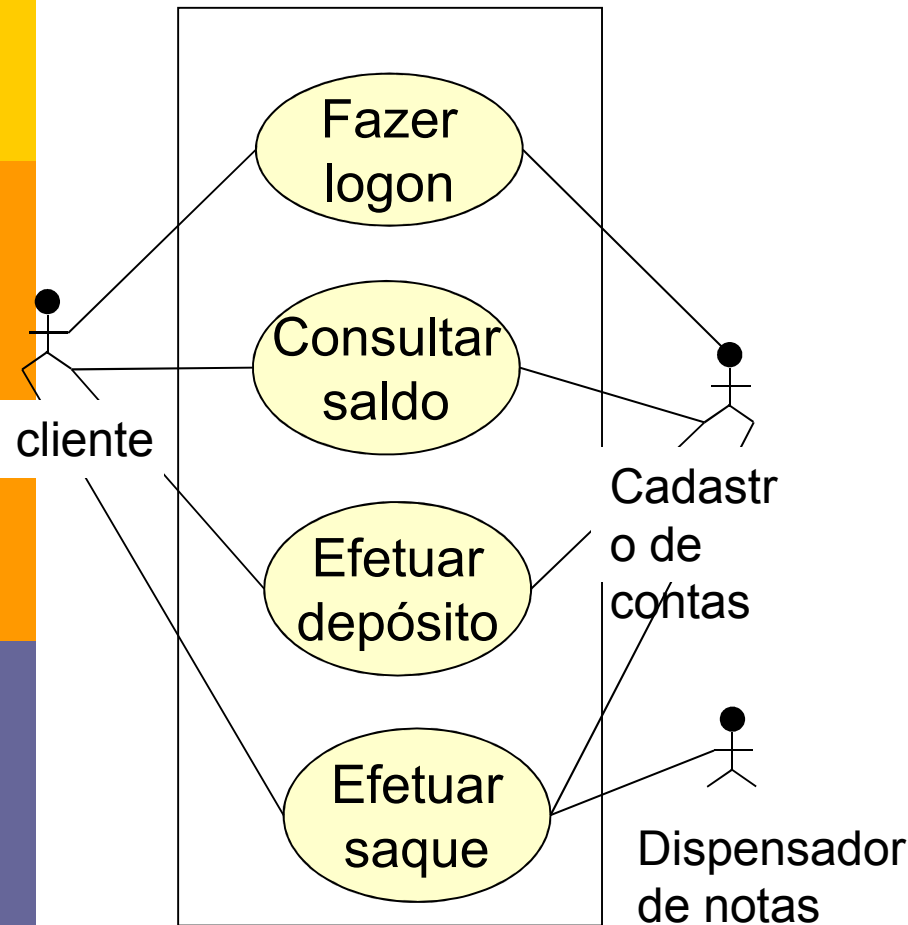
## ■ Breve descrição:

- Usuário entra com *username* e *senha* e aperta *Logon*
- Usuário também pode marcar a caixa ao lado de *Remember me!* Nesse caso, é necessário habilitar um *cookie* no seu sistema.
- Ao entrar no sistema, cria-se uma sessão no servidor para controlar suas atividades durante o uso da aplicação web

Shams Mukhtar 2004: <http://www.codeproject.com/aspnet/ModelViewController.asp>

# Exercício 2 (1)

Baseado em [Rubira2006]



- Elabore o diagrama de atividades representando as seqüências válidas de casos de uso desse sistema.
- Elabore os procedimentos e os casos de teste para o caso de uso Efetuar saque, descrito resumidamente a seguir e no próximo slide:
- Descrição: O cliente, já autenticado, escolhe a opção Efetuar Saque, informa a quantia desejada e, caso o saldo da conta seja suficiente e o caixa tenha o dinheiro necessário, a quantia é liberada.
- **Pré-condições:** (i) o cliente deve ter efetuado o caso de uso Efetuar logon para entrar no sistema; (ii) a conta deve estar ativa; (iii) a quantia deve ser maior do que zero e não pode ser superior ao saldo e nem ao total de dinheiro disponível no caixa.
- **Pós-condições:** (i) a quantia é debitada do saldo da conta e do total disponível no caixa; (ii) a quantia é fornecida ao cliente.

# Exercício 2 (2)

Baseado em [Rubira2006]

- **Fluxo básico:**

1. O cliente escolhe no menu principal do terminal a opção Efetuar Saque.
2. O sistema verifica se o login foi efetuado.
3. O sistema verifica se a conta está ativa, através do Cadastro de Contas.
4. O sistema solicita que o cliente informe a quantia desejada.
5. O cliente informa a quantia desejada.
6. O sistema verifica se o saldo da conta é suficiente para realizar a transação e, em caso afirmativo, se há dinheiro em quantidade suficiente no caixa.
7. O sistema subtrai o valor solicitado do saldo da conta do cliente e do valor disponível no caixa e libera a quantia solicitada, através do Dispensador de notas.

- **Fluxo alternativo 1:**

- No passo 2 do Fluxo Básico, se o login não tiver sido efetuado, o sistema informa isso ao cliente.

- **Fluxo alternativo 2:**

- No passo 3 do Fluxo Básico, se a conta não estiver ativa, o sistema avisa isso ao cliente e informa que o saque não pode ser realizado.