
ânima
EDUCAÇÃO
UNISUL
MODELAGEM DE SOFTWARE
Prof. Richard Henrique de Souza
Prof. Ricardo Ribeiro Assink
ricardo.assink@unisul.br
Prof. Rafael Lessa
rafael.lessa@unisul.br

Agenda

Modelagem UML e a Orientação a Objetos

- *Unified Modeling Language* – UML
- Digrama de atividades

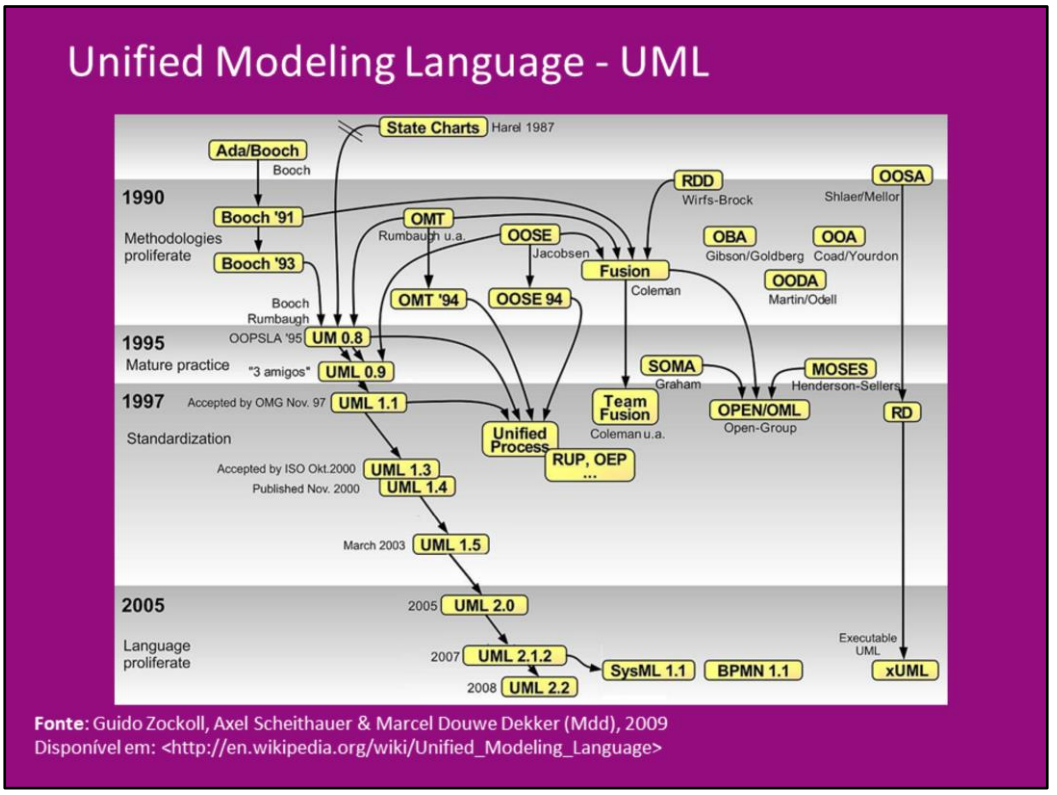
Unified Modeling Language - UML

Diversas notação, processos, e ferramentas para descrição do projeto orientado a objetos foram proposta nos anos 80 e 90.

A UML era uma tentativa de padronizar a modelagem para que qualquer sistema possa ser descrito corretamente de forma consistente e clara. Existiam várias metodologias de modelagem orientada a objetos que causavam uma guerra entre a comunidade de desenvolvedores orientado a objetos. A UML acabou com esta guerra trazendo as melhores ideias de cada uma destas metodologias, e mostrando como deveria ser a migração de cada uma para a UML.

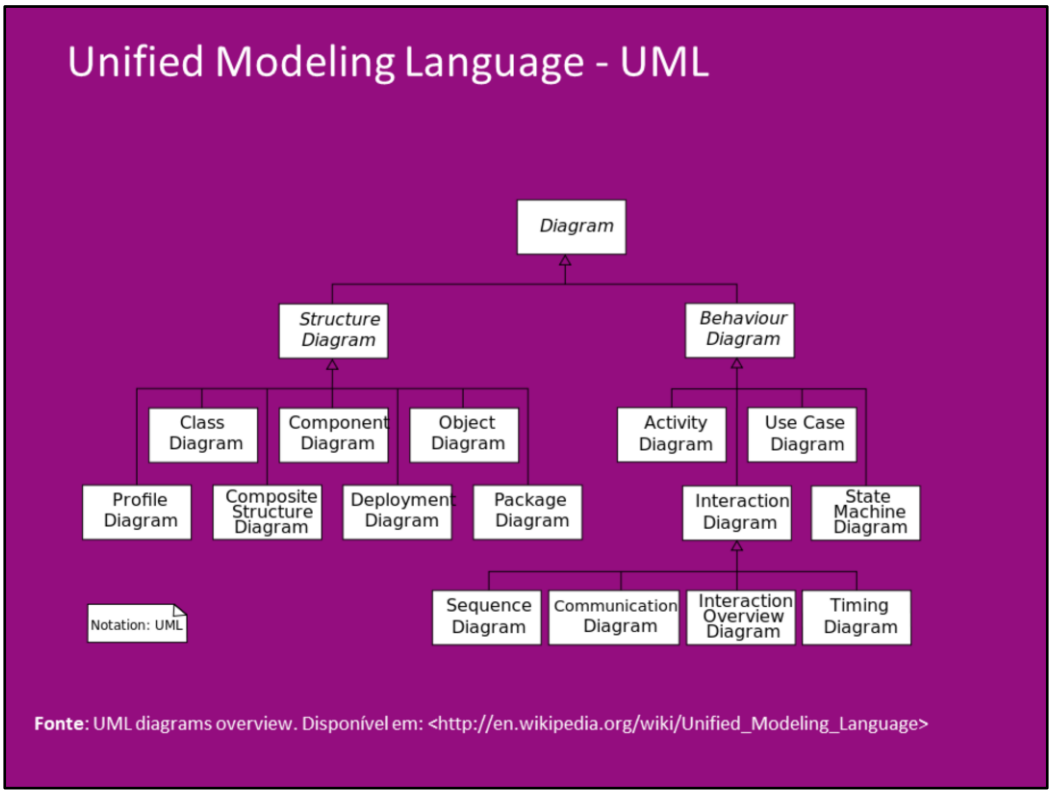
A UML foi desenvolvida por Grady Booch, James Rumbaugh, e Ivan Jacobson. Sendo que a OMG (*Object Management Group*), em 1997, unificou na UML diferentes notações existentes na época.

Versão atual da UML 2.0 <http://www.uml.org>

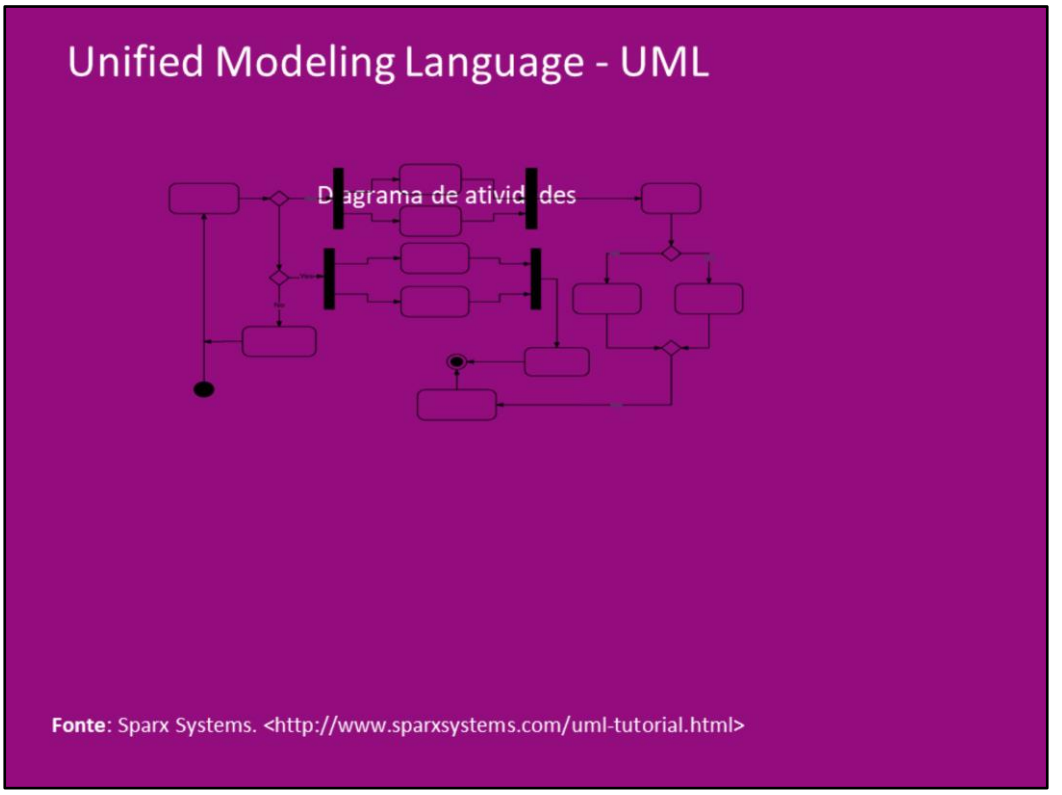


Fonte: Guido Zockoll, Axel Scheithauer & Marcel Douwe Dekker (Mdd), 2009
Disponível em: <http://en.wikipedia.org/wiki/Unified_Modeling_Language>

Descrição da imagem:
Em 1987 temos State Charts [Harel 1987]
Também na década de 80 temos Ada/Booch [Booch]
Entre 1990 até 1994 temos as methodohies proliferate: Booch ’91 , Booch ’93, OMT [Rumabaugh u.a], OMT ‘94, OOSE [Jacobsen], OOSE 94, RDD [wirfs-Brock], Fusion [Coleman], OBA [Gibson/Goldberg], OOSA [Shlaer/Mellor], OOA [Coad/Yourdon], OODA [Martin/ Odell]
Entre 1995 e 1996 temos a Mature practice: UM0.8 [Booch, Rumbaugh, OOPSLA ’95], UML 0.9 [“3 amigos”], SOMA [Graham], MODES [Henderson-Sellers.
Entre 1997 até 2003 temos a Standartization: UML 1.1 [Accepted by OMG nov. 97], Unified Process, Team Fusion [Coleman u.a.] OPEN/OML [Open-Group], RD, UML 1.3 [Accepted by ISO Okt. 2000], RUP, OPE,..., UML 1.4 [Published Nov. 2000], UML 1.5 [March 2003]
Após 2005, Language proliferate: UML 2.0 [2005], UML 2.1.2 [2007], SysML 1.1, BPMN 1.1, xUML [Executable UML], UML 2.2 [2008]



Descrição textual do DIAGRAMA
Notação UML
Primeiro temos o elemento “Diagram” no TOPO do diagrama, ligados a outros dois elementos: “Structure Diagram” e “Behaviour Diagram”, essa ligação é a de herença, onde “Diagram” é o elemento genérico.
O Elemento “Structue Diagram” está ligado aos elementos: “ Profle Diagram”, “Class Diagram”, “composite Structure Diagram”, “Component Diagram”, “Deployment Diagram”, “Object Diagram” e “Package Diagram”, também a ligação indica o relacionamento de herança, onde “Structue Diagram” é o elemento genérico.
O elemento “Behaviour Diagram” está ligado aos elementos “Activity Diagram”, ”Use case Diagram”, ”State Machine Diagram”, ”Interaction Diagram” , também a ligação indica o relacionamento de herança, onde “Behaviour Diagram” é o elemento genérico.
O elemento ”Interaction Diagram” está ligado aos elementos: “Sequence Diagram”, ”Communication Diagram”, ”Interaction Oberview Diagram”, ”Timing Diagram” , também a ligação indica o relacionamento de herança, onde “Interaction Diagram” é o elemento genérico.



#PraCegoVer
Na figura é uma representação do diagrama de atividades, onde temos os retângulos que representam as atividades, as linhas que indicam a sequencia a ser seguida e losangos para decisões e linhas grossas representam inicio de paralelismo e o fim do paralelismo.

Em dotUML
A // é inicio e comentário
Assim como / * inicia comentário em bloco (várias linhas)
E */ termina um comentário em bloco (várias linhas)
Usei os comentários para explicar o que acontece em cada comando,
Segue a figura em dotUML:

```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{

   state s1   // cria a atividade s1
choice p1  // cria a decisao p1
fork f1     // cria um paralelismo f1
choice p2  // cria a decisao p2
fork f2  // cria um paralelismo f2
state s2  // cria a atividade s2
state s3  // cria a atividade s3
state s4  // cria a atividade s4
```

```
join j1 // cria o sincronismo j1 para as atividades que
   estavam em paralelo
state s5 // cria a atividade s5
state s6  // cria a atividade s6
join j2  // cria o sincronismo j2 para as atividades que
   estavam em paralelo
state s7 // cria a atividade s7
choice p3 // cria a decisao p3
state s8 // cria a atividade s8
state s9  // cria a atividade s9
choice p4 // cria a decisao p4
state s10  // cria a atividade s10
state s11  // cria a atividade s11

   initialState -> s1  // Indica que a atividade inicial é
   a s1
s1 -> p1 // Indica que após a atividade s1 o próximo passo
   é p1
p1 -> f1 "Yes"  // Se na decisão a resposta for YEs então
   vai para f1
p1 -> p2 // Se na decisão a resposta for diferente YEs
   então vai para p2
p2 -> f2 "Yes"  // Se na decisão a resposta for YEs então
   vai para f2
p2 -> s2 "No"  // Se na decisão a resposta for No então vai
   para s2
s2 -> finalstate // Indica que depois da atividade s2
   termina o processo
 f1 -> s3 // Indica que s3 irá ocorrer em paralelo
 f1 -> s4 // Indica que s4 irá ocorrer em paralelo a s3, ou
   seja as atividades podem ocorrer simultaneamente ou
   independentemente
 f2 -> s5 // Indica que s5 irá ocorrer em paralelo
 f2 -> s6 // Indica que s6 irá ocorrer em paralelo a s5, ou
   seja as atividades podem ocorrer simultaneamente ou
   independentemente
 s3 -> j1 // Finaliza paralelismo em j1
 s4 -> j1 // Finaliza paralelismo em j1
 s5 -> j2 // Finaliza paralelismo em j2
 s6 -> j2 // Finaliza paralelismo em j2
 j1 -> s7 // Após sincronizar a próxima atividade é s7
 s7 -> p3 // Indica que após a atividade s7 o próximo passo
   é p3
 p3 -> s8 "X"  // Se a resposta é "X" então a próxima
   atividade é s8
 p3 -> s9 "Y"  // Se a resposta é "Y" então a próxima
```

```
  atividade é s9
 s8 -> p4  // Indica que após a atividade s8 o próximo
  passo é p4
 s9 -> p4  // Indica que após a atividade s9 o próximo
  passo é p4
 p4 -> s10 "YES"  // Se a resposta é "YES" então a próxima
  atividade é s10
 s10 -> finalstate   // Indica que depois da atividade s10
  termina o processo
 j2 -> s11
 s11 -> finalstate // Indica que depois da atividade s11
  termina o processo
}
```

Diagrama de atividades

Descrição da Figura:
Imagem com código de linguagens de programação.

Diagrama de Atividades

– Diagrama de atividades permite modelar o comportamento do sistema, denotando os caminhos lógicos que um processo pode seguir. Ele é um dos diagramas que compõem a visão dinâmica da UML.

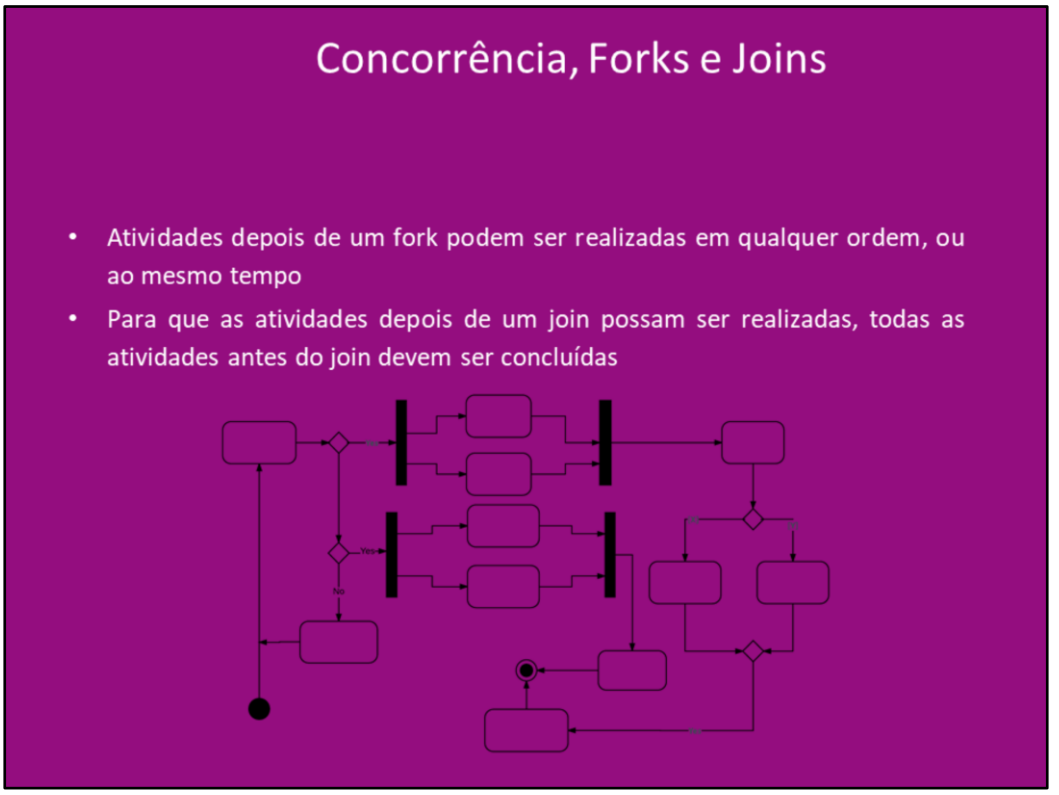
É necessário entender um processo para poder escrever ou gerar o código correto para o comportamento.

Diagramas de Atividades

- Os Diagramas de Atividades mostram o fluxo entre atividades (ações não-atômicas)
- São semelhantes aos antigos fluxogramas
- São muito usados para modelar atividades concorrentes

Concorrência, Forks e Joins

- Barras de sincronização são usadas para especificar forks e joins
- Um fork representa um único fluxo de controle em vários fluxos de controle concorrentes
- Um join representa a sincronização de dois ou mais fluxos de controle concorrentes



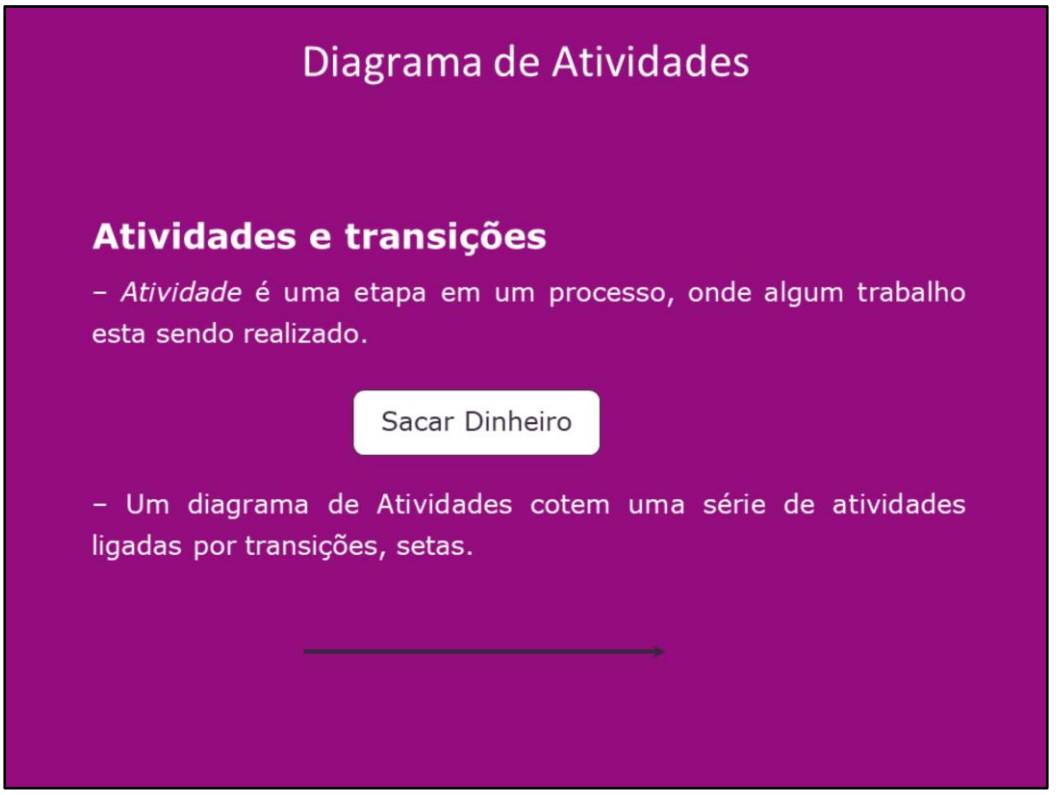
#PraCegoVer
Na figura é uma representação do diagrama de atividades, onde temos os retângulos que representam as atividades, as linhas que indicam a sequencia a ser seguida e losangos para decisões e linhas grossas representam inicio de paralelismo e o fim do paralelismo.

Em dotUML
A // é inicio e comentário
Assim como / * inicia comentário em bloco (várias linhas)
E */ termina um comentário em bloco (várias linhas)
Usei os comentários para explicar o que acontece em cada comando,
Segue a figura em dotUML:

```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{

   state s1   // cria a atividade s1
choice p1  // cria a decisao p1
fork f1     // cria um paralelismo f1
choice p2  // cria a decisao p2
fork f2  // cria um paralelismo f2
state s2  // cria a atividade s2
state s3  // cria a atividade s3
state s4  // cria a atividade s4
```

```
join j1 // cria o sincronismo j1 para as atividades que
   estavam em paralelo
state s5 // cria a atividade s5
state s6  // cria a atividade s6
join j2  // cria o sincronismo j2 para as atividades que
   estavam em paralelo
state s7 // cria a atividade s7
choice p3 // cria a decisao p3
state s8 // cria a atividade s8
state s9  // cria a atividade s9
choice p4 // cria a decisao p4
state s10  // cria a atividade s10
state s11  // cria a atividade s11

   initialState -> s1  // Indica que a atividade inicial é
   a s1
s1 -> p1 // Indica que após a atividade s1 o próximo passo
   é p1
p1 -> f1 "Yes"  // Se na decisão a resposta for YEs então
   vai para f1
p1 -> p2 // Se na decisão a resposta for diferente YEs
   então vai para p2
p2 -> f2 "Yes"  // Se na decisão a resposta for YEs então
   vai para f2
p2 -> s2 "No"  // Se na decisão a resposta for No então vai
   para s2
s2 -> finalstate // Indica que depois da atividade s2
   termina o processo
 f1 -> s3 // Indica que s3 irá ocorrer em paralelo
 f1 -> s4 // Indica que s4 irá ocorrer em paralelo a s3, ou
   seja as atividades podem ocorrer simultaneamente ou
   independentemente
 f2 -> s5 // Indica que s5 irá ocorrer em paralelo
 f2 -> s6 // Indica que s6 irá ocorrer em paralelo a s5, ou
   seja as atividades podem ocorrer simultaneamente ou
   independentemente
 s3 -> j1 // Finaliza paralelismo em j1
 s4 -> j1 // Finaliza paralelismo em j1
 s5 -> j2 // Finaliza paralelismo em j2
 s6 -> j2 // Finaliza paralelismo em j2
 j1 -> s7 // Após sincronizar a próxima atividade é s7
 s7 -> p3 // Indica que após a atividade s7 o próximo passo
   é p3
 p3 -> s8 "X"  // Se a resposta é "X" então a próxima
   atividade é s8
 p3 -> s9 "Y"  // Se a resposta é "Y" então a próxima
```

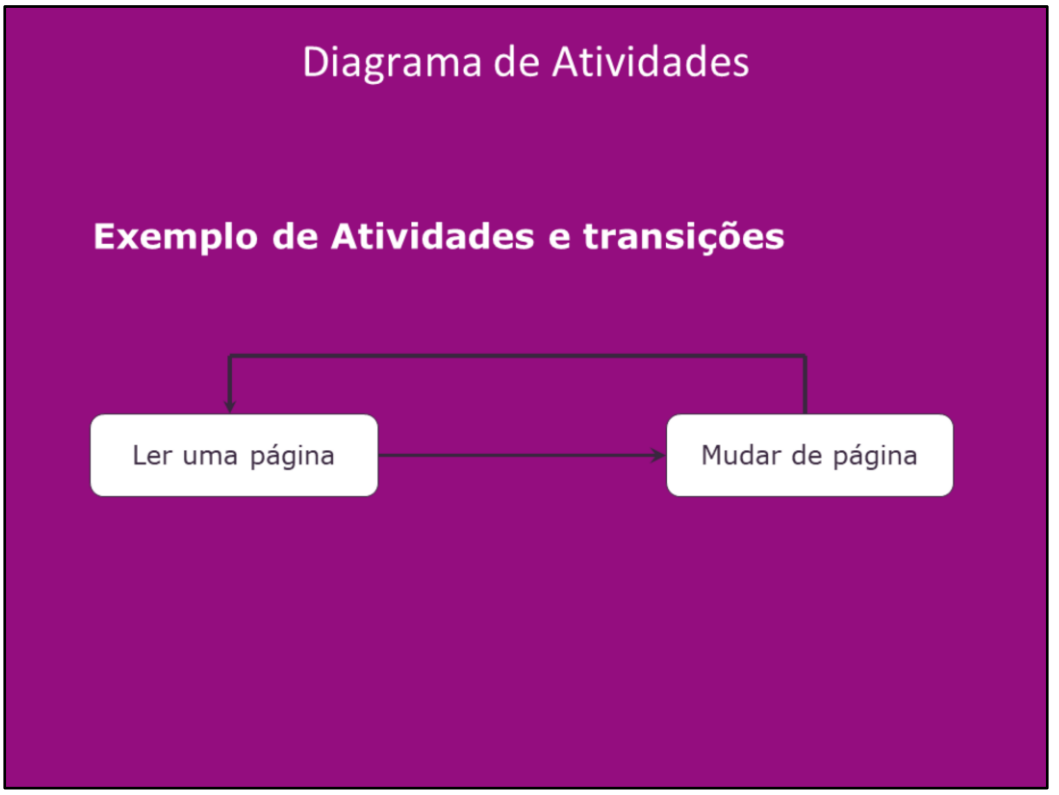
```
  atividade é s9
 s8 -> p4  // Indica que após a atividade s8 o próximo
  passo é p4
 s9 -> p4  // Indica que após a atividade s9 o próximo
  passo é p4
 p4 -> s10 "YES"  // Se a resposta é "YES" então a próxima
  atividade é s10
 s10 -> finalstate   // Indica que depois da atividade s10
  termina o processo
 j2 -> s11
 s11 -> finalstate // Indica que depois da atividade s11
  termina o processo
}
```



#PraCegoVer
A primeira figura é um retângulo com as pontas arredondadas e dentro está escrito Sacar Dinheiro. A figura indica que Sacar Dinheiro é um exemplo de atividade.
A segunda figura é uma linha com uma flexa na ponta. As atividades são ligadas por meio de uma linha com uma seta indicando a direção (Sequencia).

EmdotUML

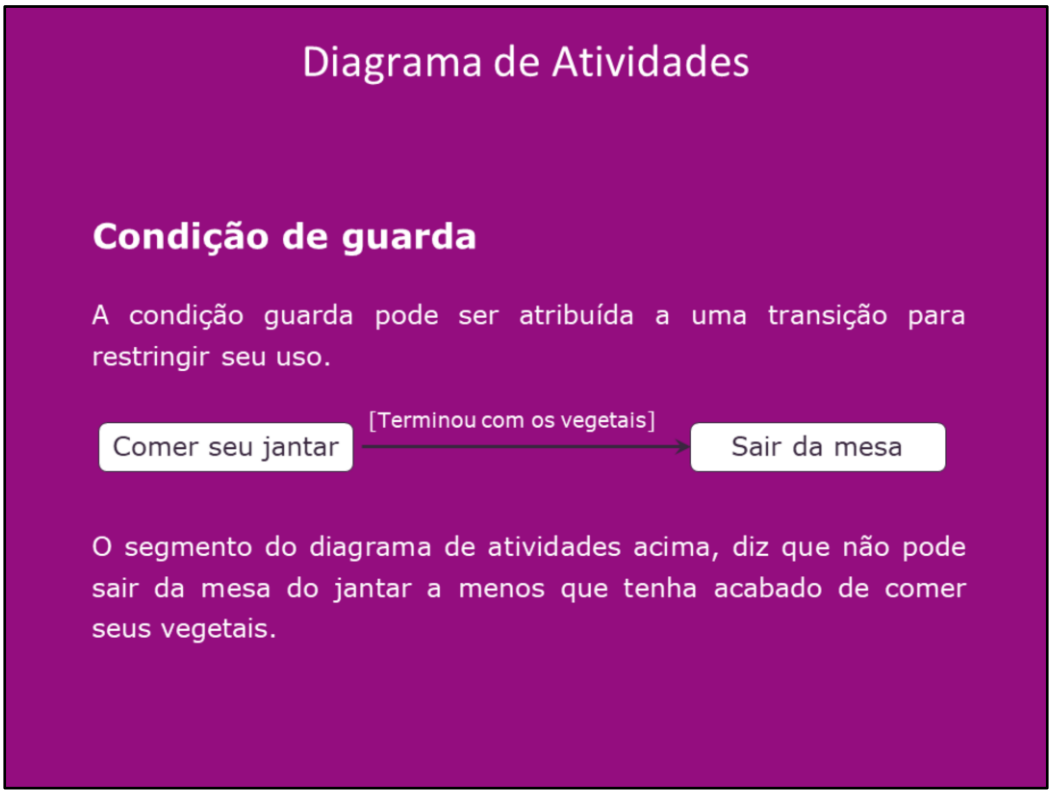
```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{
 state s1 as “Sacar Dinheiro"
}
```



#PraCegoVer
Na figura é apresentado duas atividades: Ler uma página e Mudar de página, por meio das setas pode-se perceber que é possível passar da atividade Ler uma página para a atividade Mudar de página. E a segunda seta indica que é possível passar da atividade Mudar de página para a atividade Ler uma página.

EmdotUML

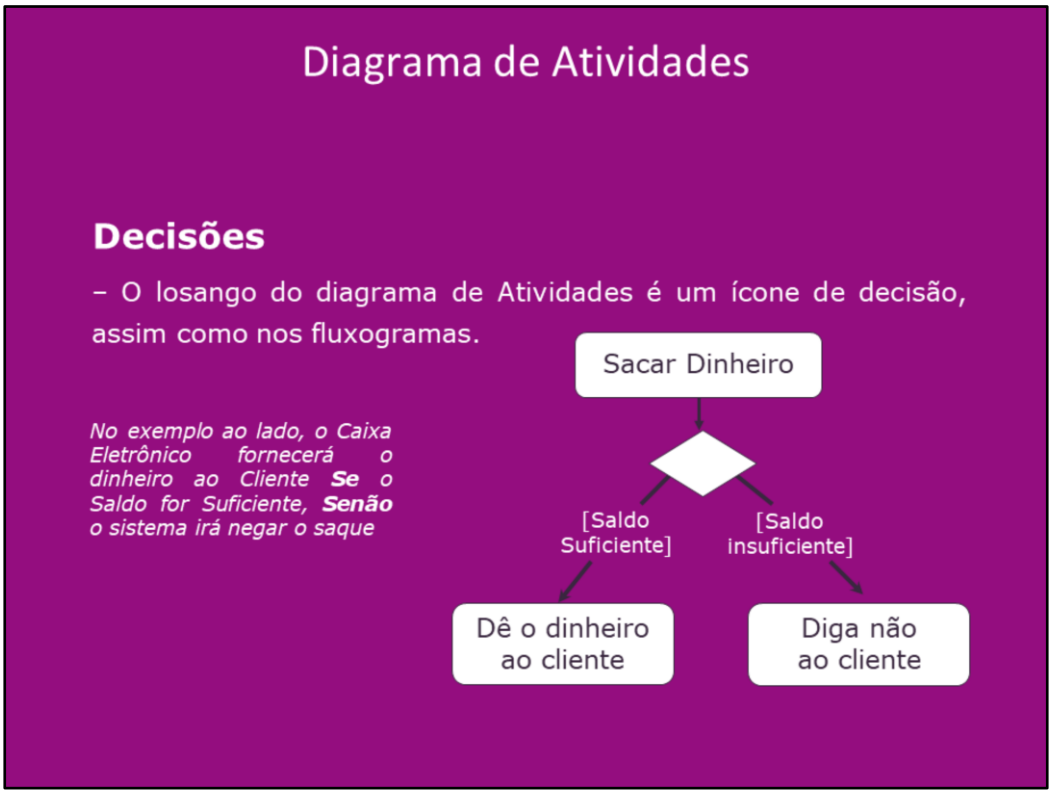
```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{

   state s1 as "Ler uma página"
state s2 as "Mudar de página"

   s1 -> s2
s2 -> s1
}
```



#PraCegoVer
Na figura é descrito duas atividades: Comer seu jantar e sair da mesa, sabe-se que a primeira atividade é comer seu jantar pois a flexa está apontada para a atividade Sair da mesa, e em cima da linha que liga as atividades tem a seguinte condição de guarda [Terminou com os vegetais].

Em dotUML

```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{

   state s1 as "Comer seu jantar"
state s2 as "Sair da mesa"

   s1 -> s2 "[Terminou com os vegetais]"
}
```



#PraCegoVer
Na figura tem três atividades: Sacar Dinheiro, Dê o dinheiro ao cliente e Diga não ao Cliente.
A primeira atividade é Sacar Dinheiro, logo depois pela seta é uma Decisão representada por um losango,
Onde se a decisão for de acordo com a guarda [Saldo Suficiente] então a próxima atividade é Dê o dinheiro ao Cliente.
Mas se a decisão for de acordo com a guarda [Saldo insuficiente] então a próxima atividade será Diga não ao Cliente.

Em dotUML

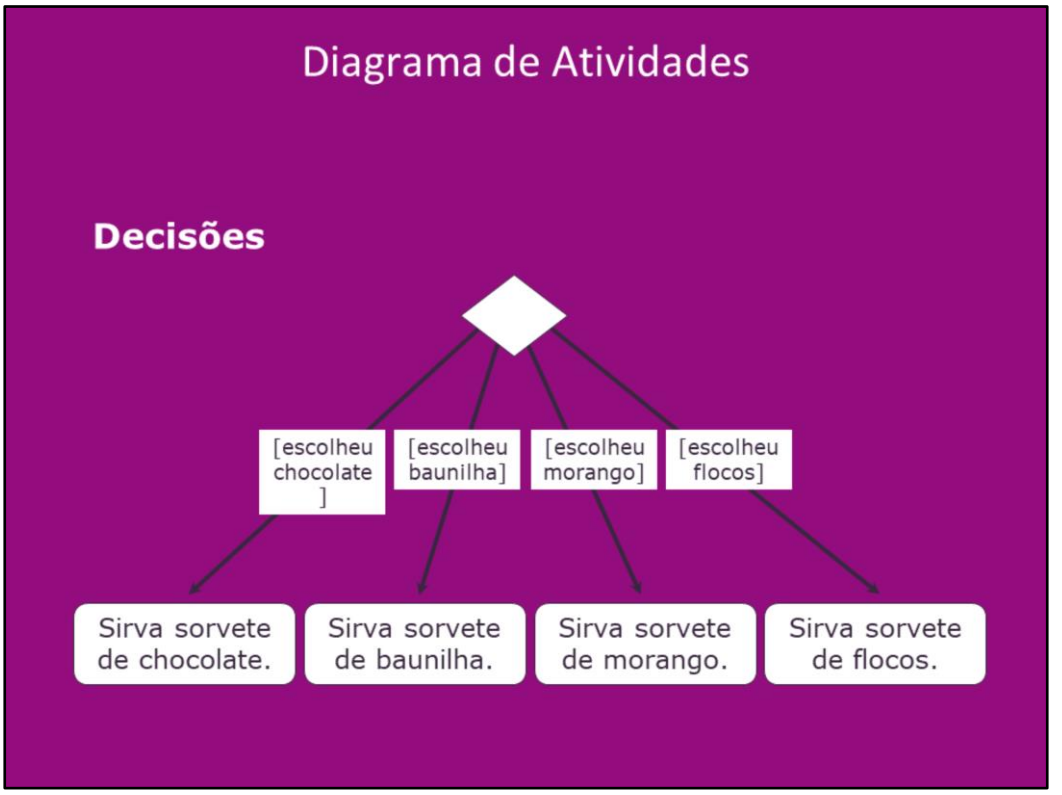
```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{

   state s1 as "Sacar Dinheiro"
choice p1
state s2 as "Dê o dinheiro ao cliente"
state s3 as "Diga não ao cliente"

s1 -> p1
p1 -> s2 "[saldo Suficiente]"
p1 -> s3 "[saldo insuficiente]"
}
```

Diagrama de Atividades

Decisões

Cada opção é identificada por meio de uma condição guarda. Cada condição deve ser mutuamente exclusiva, de modo que somente uma opção seja possível em qualquer ponto de decisão.

Essa construção está relacionada a instruções **case** ou estruturas **if-else**

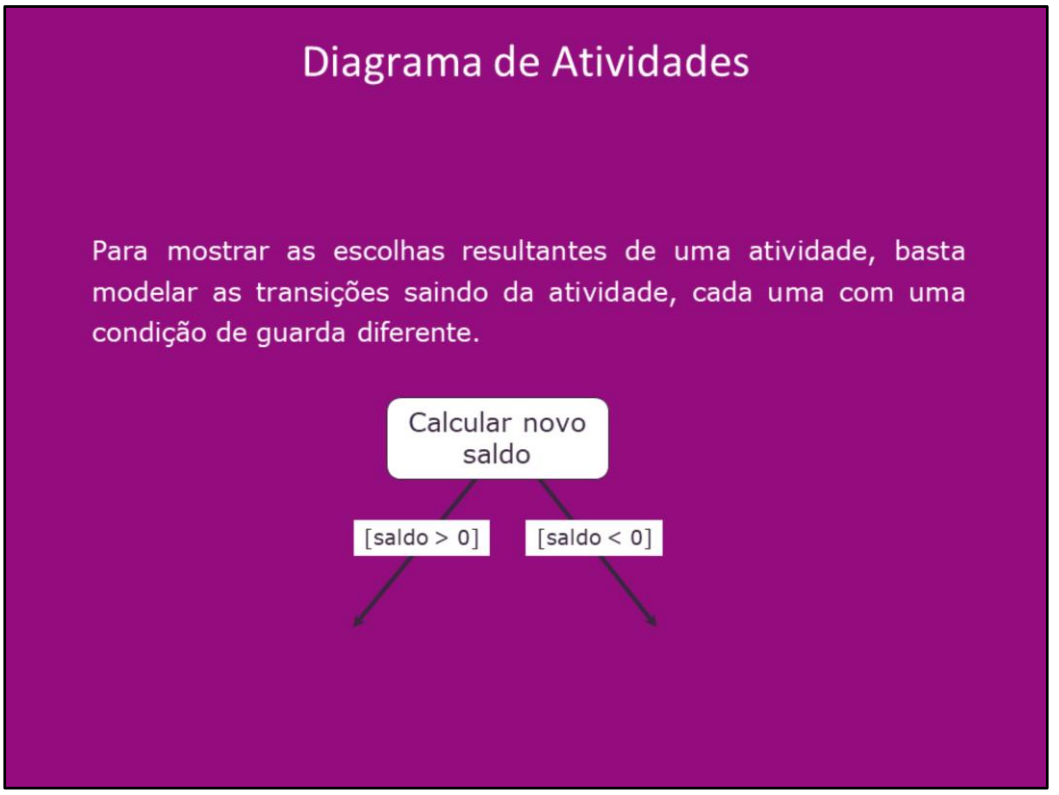


#PraCegoVer
A Figura tem quatro atividades: Sirva sorvete de chocolate, Sirva sorvete de baunilha, Sirva sorvete de morango, Sirva sorvete de flocos.
O inicio é uma decisão onde:
se a decisão for de acordo com a guarda [escolheu chocolate] então a atividade ser realizada é a de Sirva sorvete de chocolate.
se a decisão for de acordo com a guarda [escolheu baunilha] então a atividade ser realizada é a de Sirva sorvete de baunilha.
se a decisão for de acordo com a guarda [escolheu morango] então a atividade ser realizada é a de Sirva sorvete de morango.
se a decisão for de acordo com a guarda [escolheu flocos] então a atividade ser realizada é a de Sirva sorvete de flocos.

EmDotUML

```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{

   choice p1 /* Colocar em comentário o que significa ou a
   pergunta */

   state s1 as "Sirva sorvete de chocolate"
state s2 as "Sirva sorvete de baunilha"
state s3 as "Sirva sorvete de morango"
```

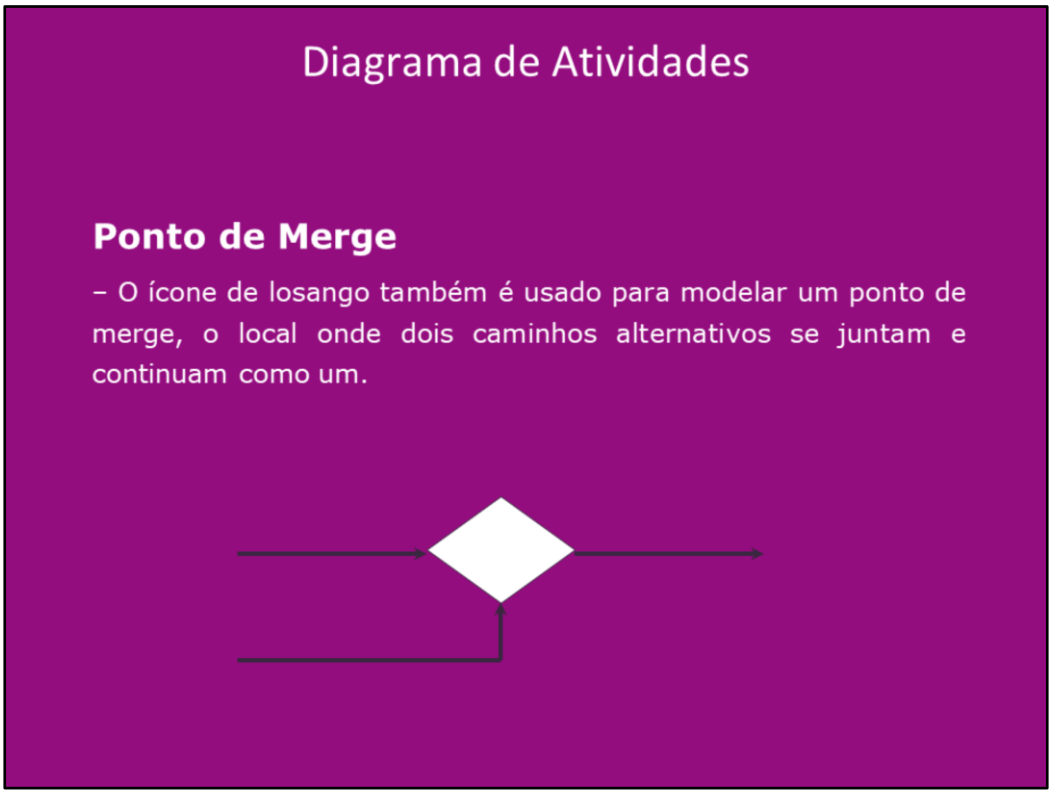
```
state s4 as "Sirva sorvete de flocos"

   p1 -> s1 "escolheu chocolate"
p1 -> s2 "escolheu baunilha"
p1 -> s3 "escolheu morango"
p1 -> s4 "escolheu flocos"
}
```



#PraCegoVer
Na Figura tem uma atividade: Calcular novo saldo e tem duas transições possíveis : uma com a guarda [saldo > 0] e outra com a guarda [saldo < 0]

Em dotUML

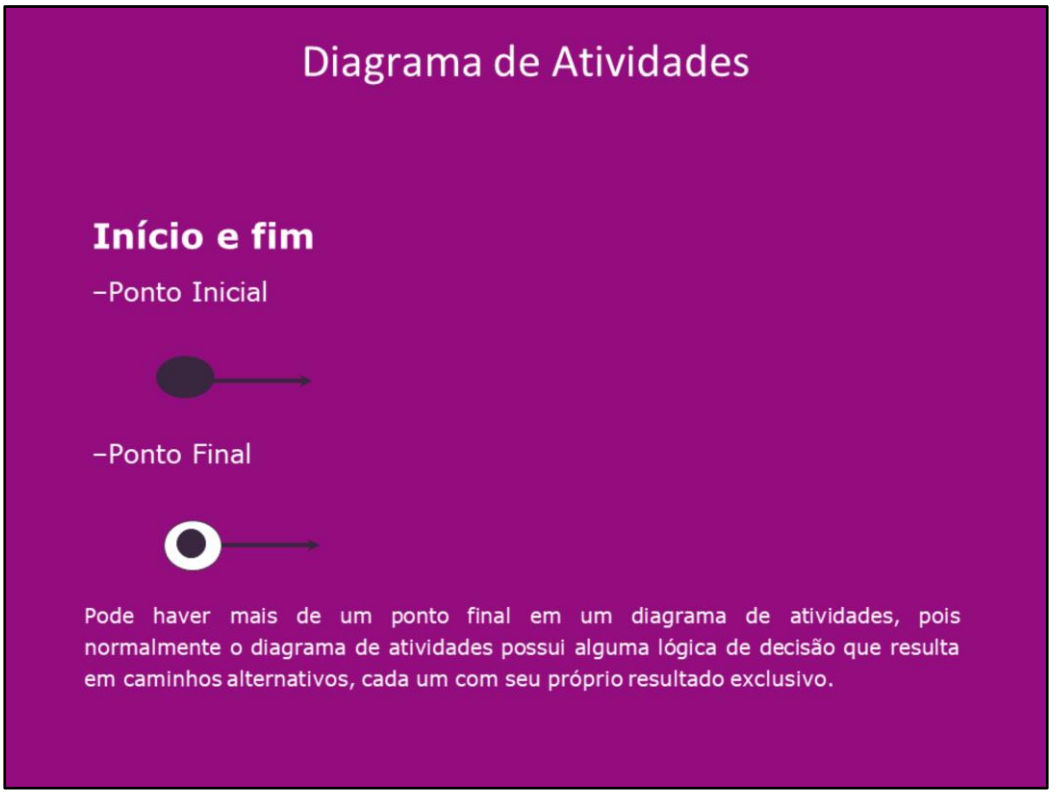
```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{

   state s1 as "Calcular novo Saldo"
state s2
state s3

   s1 -> s2 "[saldo > 0]"
s1 -> s3 "[saldo < 0]"
}
```



#PraCegoVer
Um losango é usado para realizar o merge , onde duas setas chegam e uma seta sai do losango.

Em dotUML

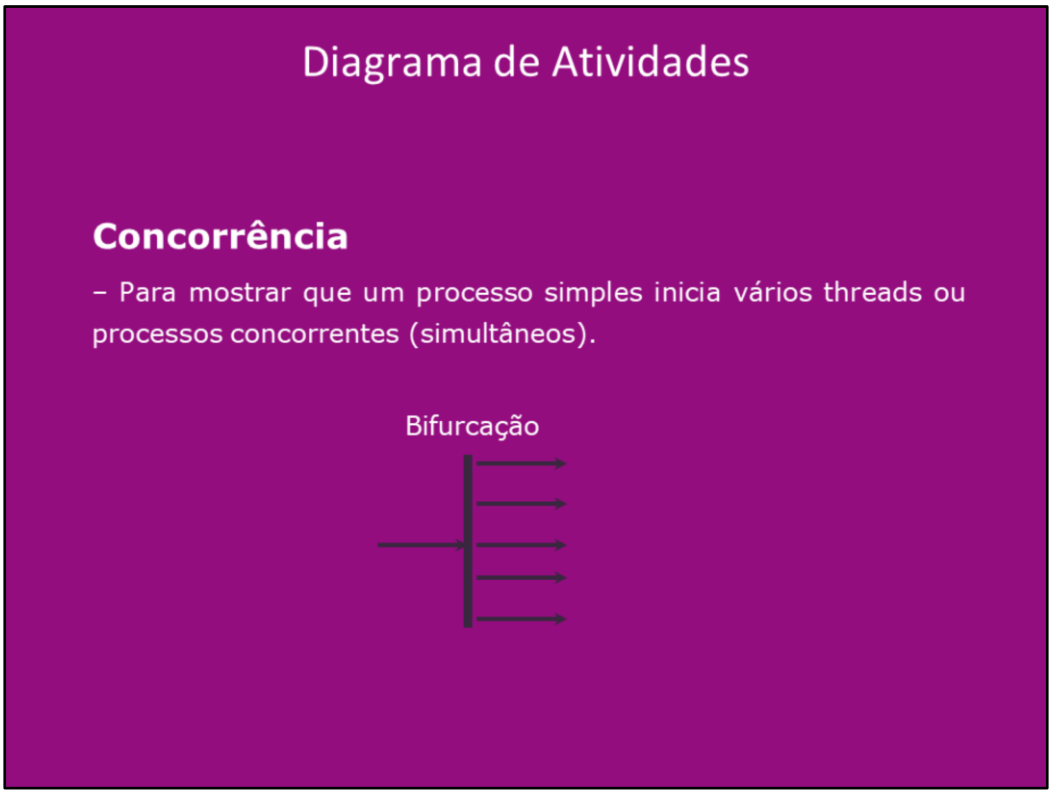
```
choice p1 /* Colocar em comentário o que significa ou a pergunta */
```



#PraCegoVer
O ponto inicial é indicado por um circulo cheio e uma seta que indicará qual é a primeira atividade.
O ponto final é indicado por um circulo com um ponto dentro.

Em dotUML

```
initialState   //
finalstate    //
```



#PraCegoVer
A bifurcação é uma seta apontando para uma linha mais grossa e várias setas saindo do lado oposto da linha grossa.

Em dotUML
fork f
E para juntar
join j

Diagrama de Atividades

Exemplo/Exercício

Nome do caso de uso: Pedido
Diálogo do caso de uso:

- O usuário Consulta Cliente

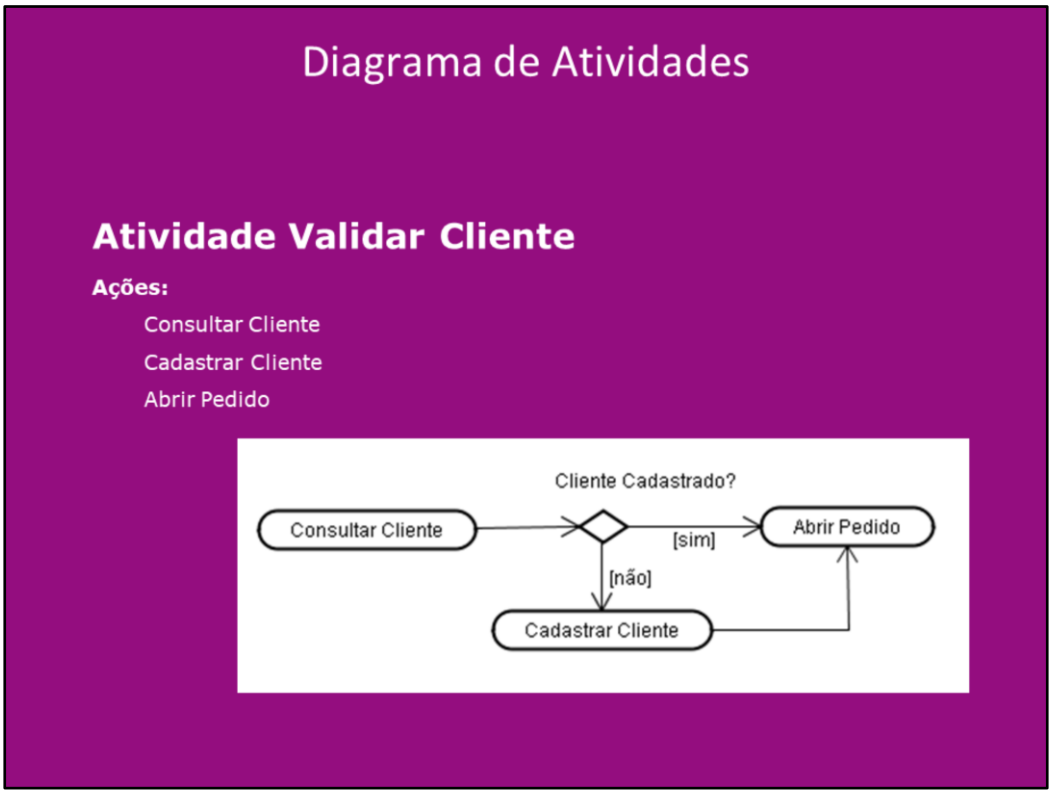
Se não for encontrado o Cliente,

- Cadastrar Cliente

Senão (localizado)

- prossegue

- Abrir Pedido



#PraCegoVer
A figura tem 3 atividades: Consultar Cliente, Cadastrar Cliente, Abrir Pedido.
Inicia com a Tarefa Consultar Cliente, depois vai para a uma decisão (Losango) com a seguinte pergunta: Cliente cadastrado ?
Caso a resposta seja não vai para a tarefa Cadastrar Cliente
Caso seja Sim vai para a tarefa Abrir Pedido.
Após executar a tarefa Cadastrar cliente a próxima atividade é de abrir pedido.

Em dotUML

```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{

   state s1 as "Consultar Cliente"
choice p1 /* Cliente cadastrado?*/
state s2 as "Abrir Pedido"
state s3 as "Cadastrar Cliente"

   s1 -> p1 "Cliente cadastrado?"
p1 -> s2 "[SIM]"
p1 -> s3 "[NÃO]"
s3 -> s2
}
```

Diagrama de Atividades

Nome do caso de uso: Pedido

Diálogo do caso de uso:

- O usuário Consulta o Produto
- O usuário Consulta o Estoque
- Adiciona o Produto

Se houver mais produtos

- Consultar Produto

Senão

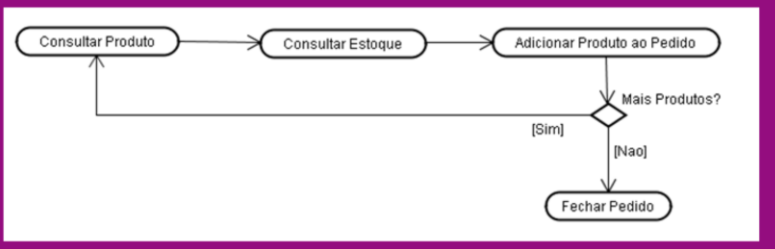
- Fechar Pedido

Diagrama de Atividades

Atividade Fazer Pedido

Ações:

- Consultar Produto
- Consultar Estoque
- Adicionar Produto ao Pedido
- Fechar Pedido

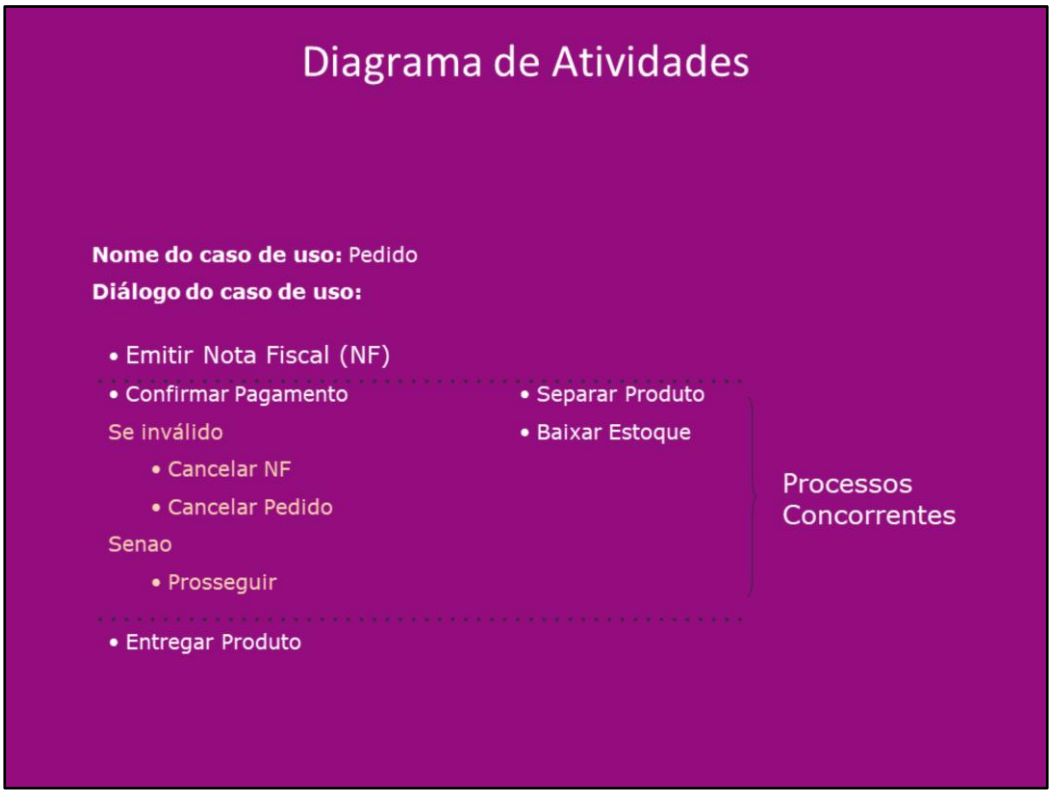


#PraCegoVer

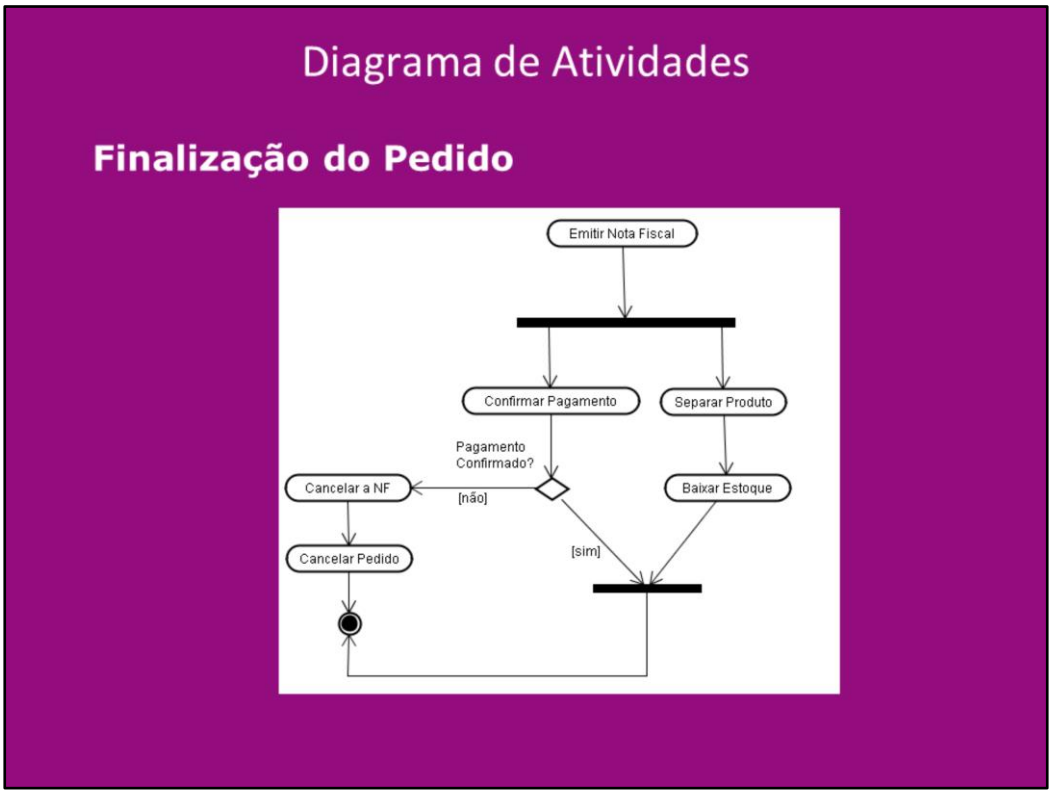
Em dotUML

```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{

   state s4 as "Consultar Produto"
state s5 as "Consultar Estoque"
state s6 as "Adicionar Produto ao Pedido"
choice p2 /* Mais Produtos ?*/
state s7 as "Fechar Pedido"

   s4 -> s5
```

```
s5 -> s6
s6 -> p2 "Mais Produtos?"
p2 -> s4 "[SIM]"
p2 -> s7 "[Não]"
}
```



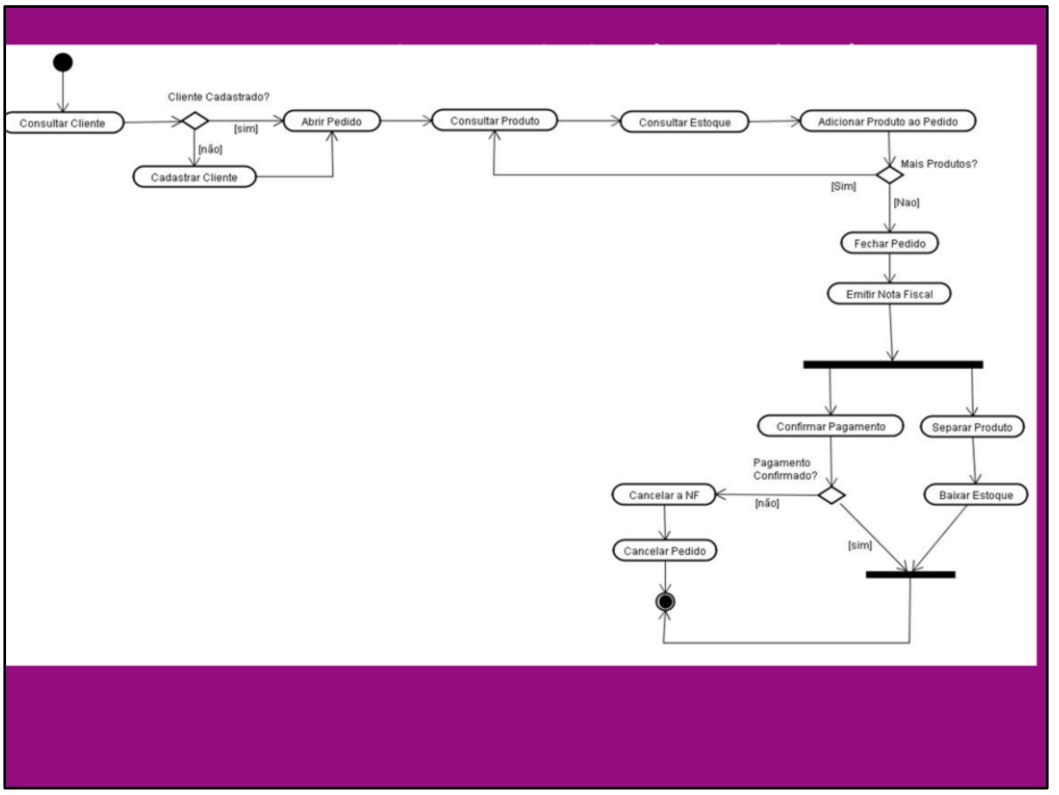
#PraCegoVer
Temos dois processos concorrentes, ou seja, em paralelo
O primeiro é a sequencia: Confirmar Pagamento, Se inválido (Cancelar NF , Cancelar Pedido) Senao Prosseguir
O segundo é Separar produto e baixar estoque.
No tem que ambos iniciam com Emitir Nota fiscal (NF)
E ambos terminam com entregar Produto.



#PraCegoVer
O diagrama de atividade inicia com a atividade Emitir nota fiscal, depois inicia um paralelismo.
Um caminho em paralelo inicia com a atividade Separar Produto e depois para a atividade Baixar estoque e então finaliza o paralelismo.
O segundo caminho em paralelo inicia com a atividade Confirmar Pagamento que vai para uma decisão: Pagamento Confirmado? Em caso [Sim] finaliza o paralelismo.
Em caso [não] inicia a atividade Cancelar a NF e depois a atividade Cancelar Pedido.
Termina as atividades de duas formas:
Após o cancelar Pedido
Ou após os dois caminhos terminarem.
Em dotUML

```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{

   state s8 as "emitir nota fiscal"
fork f /* inicia paralelismo*/
state s9  as "Separar Produto"
state s10 as "baixar estoque"
state s11 as "confirmar pagamento"
choice p3 /* Pagamento confirmado ?*/
join j /*Finaliza paralelismo*/
```

```
state s12 as "cancelar a NF"
state s13 as "cancelar Pedido"

   s8 -> f
f -> s9
f -> s11
s9 -> s10
s11 -> p3
s10 -> j
p3 -> j "[Sim]"
j -> finalstate
p3 -> s12 "[Não]"
s12 -> s13
s13 -> finalstate
}
```



#PraCegoVer
A figura tem 13 atividades: Consultar Cliente, Cadastrar Cliente, Abrir Pedido,

Inicia com a Tarefa Consultar Cliente, depois vai para a uma decisão (Losango) com a seguinte pergunta: Cliente cadastrado ?
Caso a resposta seja não vai para a tarefa Cadastrar Cliente
Caso seja Sim vai para a tarefa Abrir Pedido.
Após executar a tarefa Cadastrar cliente a próxima atividade é de abrir pedido.
abrir pedido pode ir para atividade

Emitir nota fiscal.
Emitir nota fiscal, inicia um paralelismo.

Um caminho em paralelo inicia com a atividade Separar Produto e depois para a atividade Baixar estoque e então finaliza o paralelismo.
O segundo caminho em paralelo inicia com a atividade Confirmar Pagamento que vai para uma decisão: Pagamento Confirmado? Em caso [Sim] finaliza o paralelismo.
Em caso [não] inicia a atividade Cancelar a NF e depois a atividade Cancelar Pedido.
Termina as atividades de duas formas:
Após o cancelar Pedido
Ou após os dois caminhos terminarem.

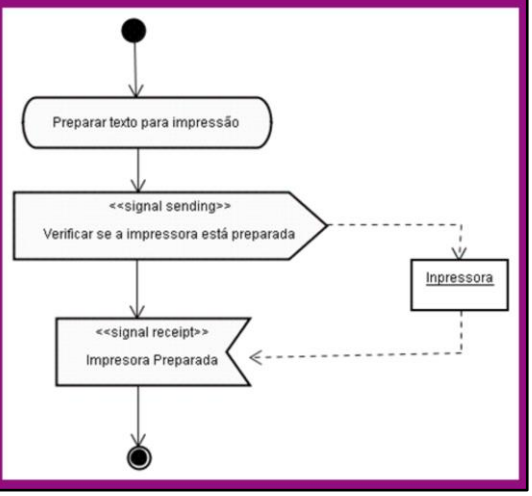
Em dotUML:

```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{

   state s1 as "Consultar Cliente"
choice p1 /* Cliente cadastrado?*/
state s2 as "Abrir Pedido"
state s3 as "Cadastrar Cliente"
state s4 as "Consultar Produto"
state s5 as "Consultar Estoque"
state s6 as "Adicionar Produto ao Pedido"
choice p2 /* Mais Produtos ?*/
state s7 as "Fechar Pedido"
state s8 as "emitir nota fiscal"
fork f /* inicia paralelismo*/
state s9  as "Separar Produto"
state s10 as "baixar estoque"
state s11 as "confirmar pagamento"
choice p3 /* Pagamento confirmado ?*/
join j /*Finaliza paralelismo*/
state s12 as "cancelar a NF"
state s13 as "cancelar Pedido"

   initialState -> s1
s1 -> p1 "Cliente cadastrado?"
p1 -> s2 "[SIM]"
p1 -> s3 "[NÃO]"
s3 -> s2
align {
s3
p1
}  // Esse align { s3 p1}  serve apenas para alinhar esses
   elementos para quem for ver a figura  deixar um do lado
   do outro
s2 -> s4
s4 -> s5
```

```
    s5 -> s6
    s6 -> p2 "Mais Produtos?"
    p2 -> s4 "[SIM]"
    p2 -> s7 "[Não]"
    s7 -> s8
    s8 -> f
    f -> s9
    f -> s11
    s9 -> s10
    s11 -> p3
    s10 -> j
    p3 -> j "[Sim]"
    j -> finalstate
    p3 -> s12 "[Não]"
    s12 -> s13
    s13 -> finalstate
}
```

Recebimento de Sinal

- Representa o recebimento de um sinal de um dispositivo externo, normalmente um item de hardware.



#PraCegoVer
Exemplo de Diagrama de atividade com Recebimento de sinal
Inicia com a atividade Preparar texto para impressão a seguir
Inicia a atividade <<sinal sending>> Verificar se a impressora está preparada
Uma linha pontilhada com uma seta sai da atividade <<sinal sending>> Verificar se a impressora está preparada
E vai ate o dispositivo impressora.
Depois da atividade <<sinal sending>> Verificar se a impressora está preparada vai para a atividade que recebe o sinal da impressoa: <<signal receipt>> Impressora Preparada,
E tem uma linha pontilhada com uma seta do dispositivo impressora até a atividade <<signal receipt>> Impressora Preparada
Após a atividade <<signal receipt>> Impressora Preparada o diagrama finaliza

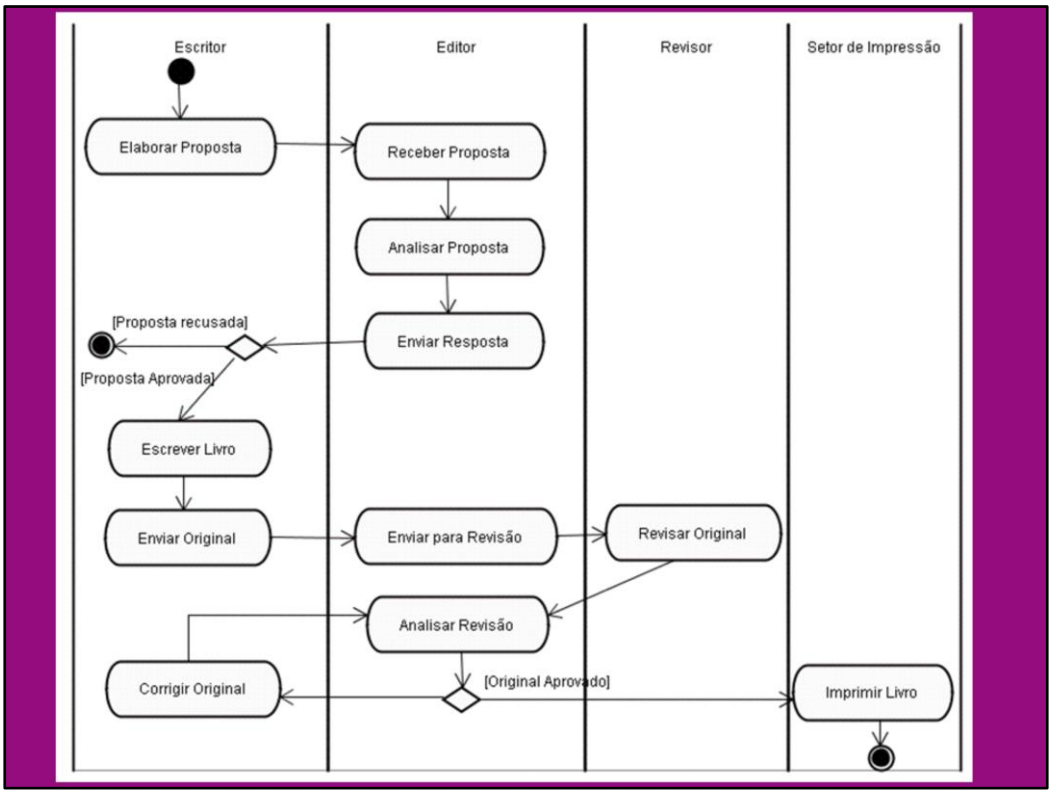
Em dotUML

```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{

   state s1 as "Preparar texto para impressão"
state s2 as "sinal sending - \n Verificar se a impressora
   está preparada "
state s3 as "signal receipt - \n Impressora Preparada"
state s4 as "impressora" [style=dashed]
```

```
   initialState -> s1
s1 -> s2
s2 ->s3
s3 -> finalstate
s2 -> s4 [arrowhead=vee style=dashed]
s4 -> s3 [arrowhead=vee style=dashed]

   }
```

Raias de Natação

- São uma extensão do Diagrama de Atividades, onde procura-se identificar os diversos setores, departamentos ou mesmo os atores que interagem com um processo.



#PraCegoVer

Existe 11 atividades: Elaborar proposta, receber Proposta, Analisar Proposta, Enviar Resposta, Escrever Livro, Enviar Original, Enviar para Revisão, Revisar Original, analisar Revisão, Corrigir Original e Imprimir Livro.

Existe 4 raias (Divisões) : Escritor, Editor, Revisor e Setor de Impressão

Inicia com a Atividade Elaborar Proposta na raia do Escritor.

Após realizar a atividade Elaborar Proposta, a próxima atividade é a de Receber Proposta que está na raia do Editor.

Após realizar a atividade Receber Proposta, a próxima atividade é a de Analisar Proposta que está na raia do Editor.

Após realizar a atividade Analisar Proposta , a próxima atividade é a de Enviar Resposta que está na raia do Editor.

Após realizar a atividade Enviar Resposta, vai para uma decisão, se a opção for [Proposta recusada] finaliza o processo,

Se a opção for [proposta Aprovada], vai para a próxima atividade que é a de Escrever Livro que está na raia do Escritor.

Após realizar a atividade Escrever Livro, a próxima atividade é a de Enviar Original que está na raia do Escritor.

Após realizar a atividade Enviar Original , a próxima atividade é a de Enviar para Revisão que está na raia do Editor.

Após realizar a atividade Enviar para Revisão, a próxima atividade é a de revisar Original que está na raia do Revisão.

Após realizar a atividade Enviar para Revisão, a próxima atividade é a de Analisar

Revisão que está na raia do Editor.

Após realizar a atividade Analisar Revisão, vai para uma decisão: Se opção for [original Aprovado] vai para a atividade Imprimir Livro na raia Setor de impressão.

Se a opção for [Não aprovado] vai para a atividade Corrigir Original na raia Escritor.

Após realizar a atividade Corrigir Original, a próxima atividade é a de Analisar Revisão que está na raia do Editor.

Após realizar a atividade Imprimir Livro, termina o processo.

Em dotUML:

```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho]{
// Escritor
state s1 as "Elaborar proposta"
choice p1 /* Proposta*/
state s2 as "Escrever Livro"
state s3 as "Enviar Original"
state s4 as "Corrigir Original"
//Editor,
state s5 as "receber Proposta"
state s6 as "Analisar Proposta"
state s7 as "Enviar Resposta"
state s8 as "Enviar para Revisão"
state s10 as "analisar Revisão"
choice p2 /* Aprovar revisao*/
//Revisor
state s9 as "Revisar Original"
// Setor de Impressão
state s11 as "Imprimir Livro"

   initialState -> s1
s1 -> s5
s5 -> s6
s6 -> s7
s7 -> p1
p1 -> finalstate "[Proposta recusada]"
p1 -> s2 "[Poposta Aprovada]"
s2 -> s3
s3 -> s8
s8 -> s9
s9 -> s10
s10 -> p2
p2 <- s4
p2 -> s11 "[Original Aprovado]"
```

}

Após realizar a atividade Elaborar Proposta, a próxima atividade é a de Receber Proposta que está na raia do Editor.

Vamos trabalhar?

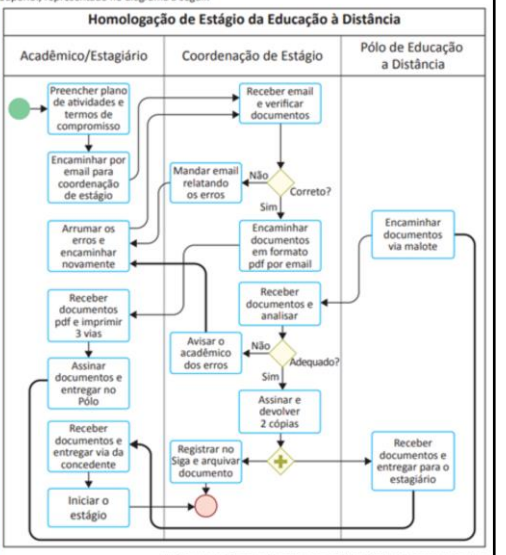
Exercício at1

1. Considerando o diagrama apresentado, faça o que se pede nos itens a seguir.

a) Cite o nome de duas tarefas, dois gateways (Decisões) e duas raias que podem ser identificados.

b) Se o estagiário entregar os documentos errados, cite as tarefas que deverão ser realizadas e por quem.

A notação *Business Process Model Notation* (BPMN) adota diversos elementos padronizados para representação de processos. Analise o processo de homologação de estágio em uma Instituição de Ensino Superior, representado no diagrama a seguir.



UFJF. Escritório de Processos. Disponível em: www2.ufjf.br/escritoriodeprocessos/portifolio/ Acesso em: 26 Ago. 2021 (adaptado).

Figura no link:
https://drive.google.com/drive/folders/1HoaXb2H-kuQ1HC37ljvOpDJwp1m6BNvW?usp=sharing

#PraCegoVer

1. Diagrama de atividades do Processo de Homologação de Estágio da Educação à Distância
2. Tem três raias: Acadêmico/Estagiário, Coordenação de Estágio, Pólo de Educação a Distância
3. Tem 16 atividades: Preencher plano de atividades e termos de compromisso, Encaminhar por email para coordenação de estágio, Receber email e verificar documentos, mandar email relatando os erros, arrumar os erros e encaminhar novamente, encaminhar documentos em formato pdf por email, encaminhar documentos via malote, receber documentos e analisar, receber documentos pdf e imprimir 3 vias, avisar o acadêmico, Assinar e devolver 2 cópias, Assinar documentos e entregar no Pólo, Receber documentos e entregar via da concedente, Iniciar estágio, Registrar no Siga e arquivar documento, Receber documentos e entregar para o estagiário.

4. Inicia com a atividade de Preencher plano de atividades e termos de compromisso na raia do Acadêmico/Estagiário.
5. Após realizar a atividade Preencher plano de atividades e termos de compromisso, a próxima atividade é a de Encaminhar por email para coordenação de estágio que está na raia do Acadêmico/Estagiário.
6. Após realizar a atividade Encaminhar por email para coordenação de estágio e termos de compromisso, a próxima atividade é a de Receber email e verificar documentos que está na raia do Coordenação de Estágio.

7.Após realizar a atividade Receber email e verificar documentos tem a decisão conforme a pergunta " correto?"
8.Caso a resposta seja SIM: a próxima atividade é Encaminhar documentos em formato pdf por email na raia do Coordenação de Estágio.
9.Caso a resposta seja Não: a próxima atividade é Mandar email relatando os erros na raia do Coordenação de Estágio.
10.Após realizar a atividade Encaminhar documentos em formato pdf por email, a próxima atividade é a de Receber documentos pdf e imprimir 3 vias que está na raia do Acadêmico/Estagiário.
11.Após realizar a atividade Receber documentos pdf e imprimir 3 vias a próxima atividade é a de Assinar documentos e entregar no Pólo que está na raia do Acadêmico/Estagiário.
12.Após realizar a atividade Assinar documentos e entregar no Pólo a próxima atividade é a de Encaminhar documentos via malote que está na raia do Pólo de Educação a Distância.
13.Após realizar a atividade Encaminhar documentos via malote a próxima atividade é a de Receber documentos e analisar que está na raia Coordenação de Estágio.
14.Após realizar a atividade Encaminhar documentos via malote a próxima atividade é a de Receber documentos e analisar que está na raia Coordenação de Estágio.
15.Após realizar a atividade Receber documentos e analisar tem a decisão conforme a pergunta "Adequado?"
16.Caso a resposta seja SIM: a próxima atividade é Assinar e devolver 2 cópias na raia do Coordenação de Estágio.
17.Caso a resposta seja Não: a próxima atividade é Avisar acadêmico dos erros na raia do Coordenação de Estágio.
18.Após realizar a atividade Assinar e devolver 2 cópias o processo vai para uma bifurcação para continuar com duas atividades em paralelo: Registrar no Siga e arquivar documentos na raia Coordenação de Estágio e a atividade Receber documentos e entregar para o estagiário na raia Pólo de Educação a Distância.
19.Após realizar a atividade Registrar no Siga e arquivar documentos é o processo espera a outra atividade em paralelo finaliza para terminar o processo.
20.Após realizar a atividade Receber documentos e entregar para o estagiário a próxima atividade é a de Receber documentos e entregar via da concedente que está na raia Acadêmico/Estagiário.
21.Após realizar a atividade Receber documentos e entregar via da concedente a próxima atividade é a de iniciar estágio que está na raia Acadêmico/Estagiário.
22.Após realizar a atividade iniciar estágio, o paralelismo finaliza e o processo termina.

23.Voltando para o caminho Adequado = não (passo 17):
24.Após realizar a atividade Avisar o acadêmico dos erros a próxima atividade é

a de Arrumar os erros e encaminhar novamente que está na raia Acadêmico/Estagiário.

25.Após realizar a atividade Arrumar os erros e encaminhar novamente atividade é a de Receber email e verificar documentos que está na raia Coordenação de Estágio.

26. Volta ao passo 7

27.Voltando para o caminho correto = não (passo 9):

28.Após realizar a atividade Mandar email relatando os erros a próxima atividade é a de Arrumar os erros e encaminhar novamente que está na raia Coordenação de Estágio.

29. Ir ao passo 25

Em dotUML:

```
StateDiagram [frame=true framecolor=steelblue
    label="State Diagram" splines=ortho]{

/* Atividades do Acadêmico/Estagiário  */

state s1 as "Preencher plano de atividades e termos
    de compromisso"

state s2 as "Encaminhar por email para coordenação
    de estágio"

state s3 as "arrumar os erros e encaminhar
    novamente"

state s4 as "receber documentos pdf e imprimir 3
    vias"

state s5 as "Assinar documentos e entregar no Pólo"

state s6 as "Receber documentos e entregar via da
    concedente"

state s7 as "Iniciar estágio"

/* Atividades do Cordenação de Estágio  */
```

```
state s8 as "Receber email e verificar documentos"
choice p1 /* Correto ?*/
state s9 as "mandar email relatando os erros"
state s10 as "encaminhar documentos em formato
   pdf por email"
state s12 as "receber documentos e analisar"
choice p2 /*Adequado*/
state s13 as "Assinar e devolver 2 cópias"
state s14 as "avisar o acadêmico dos Erros"
 fork f
state s15 as "Registrar no Siga e arquivar documento"
/* Atividades do Pólo de Educação a Distância  */
state s11 as "encaminhar documentos via malote"
state s16 as "Receber documentos e entregar para o
   estagiário"
initialState-> s1
s1 -> s2
s2 -> s8
s8 -> p1 "Correto?"
p1 -> s9 "[Não]"
p1 -> s10 "[Sim]"
```

```
s9 -> s3
s3 -> s8
s10 -> s4
s4 -> s5
s5 -> s11
s11 -> s12
s12-> p2 "Adequado?"
p2 -> s13 "Sim"
p2 -> s14 "Não"
s14 -> s3
s13 -> f
f -> s15
f -> s16
s15 -> finalstate
s16 -> s6
s6 -> s7
s7 -> finalstate
}
```

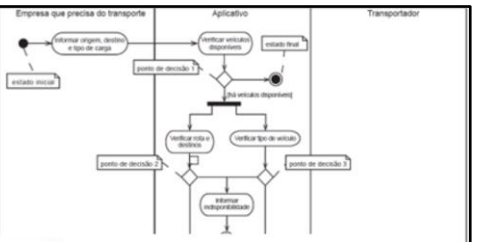
Resposta

?

Gabarito

- Coordenador de estagio:
  - Mandar email relatando os erros.
  - Avisar o acadêmico dos erros
  - Receber email e verificar documentos
  - Encaminhar documentos em formato pdf por email
- Acadêmico/Estagiário
  - Arrumar os erros e encaminhar novamente.
  - Receber documentos pdf e imprimir 3 vias
  - Assinar documentos e entregar no polo
- Polo de educação a distancia:
  - Encaminhar documentos via malote

Exercício at2



Com base no diagrama apresentado, avalie as afirmações a seguir.

I. Os serviços oferecidos por outros aplicativos, como mapas para o cálculo de rotas, por exemplo, poderão ser usados no desenvolvimento do aplicativo, mesmo que não tenham sido representados no diagrama.

II. Os fluxos paralelos indicados nos pontos de decisão 2 e 3 levam a erros no funcionamento do aplicativo, uma vez que permitem a continuidade do processo mesmo que não haja rota, destino ou tipo de veículo adequado.

III. O diagrama apresenta atividades desempenhadas pela "Empresa que precisa de transporte" e pelo "Transportador", no entanto, as atividades que precisarão ser modeladas no sistema restringem-se às desempenhadas pelo "Aplicativo".

IV. As ações "Informar indisponibilidade" e "Confirmar disponibilidade" correspondem a funcionalidades distintas no sistema, visto que a primeira se relaciona à existência de tipo de veículo, rota e destino e a segunda se relaciona à agenda do transportador.

É correto apenas o que se afirma em

A I e II. B I e IV. C II e III. D I, III e IV. E II, III e IV.

Figura no link:
https://drive.google.com/drive/folders/1HoaXb2H-kuQ1HC37ljvOpDJwp1m6BNvW?usp=sharing

#PraCegoVer

1.Diagrama de atividades do Transporte

2.Tem três raias: Empresa que precisa do transporte, Aplicativo, Transportador

3.Tem 9 atividades: Informar origem, destino e tipo de carga, Verificar veículos disponíveis, Verificar rota e destinos, informar indisponibilidade, confirmar disponibilidade, calcular valor e tempo de espera, informar dados do caminhão, valor e tempo de espera, Preparar a carga para transporte .

4.Inicia com a atividade Informar origem, destino e tipo de carga na raia do Empresa que precisa do transporte.

5.Após realizar a atividade Informar origem, destino e tipo de carga, a próxima atividade é a de Verificar veículos disponíveis que está na raia do Aplicativo.

6.Após realizar a atividade Verificar veículos disponíveis, a próxima etapa é uma decisão, chamada de ponto de decisão 1.

7. Se [não há veículos disponíveis] finaliza o processo.

8. Se [há veículos disponíveis] a próximo passo é uma bifurcação par iniciar duas atividades em paralelo: Verificar rota e destinos e Verificar tipo de veiculo.

9.Após realizar a atividade Verificar rota e destinos a próxima etapa é uma decisão chamada de ponto de decisão 2.

10: Se [não há rota ou destino atende ao pedido inicial] , a próxima atividade é a de informar indisponibilidade que está na raia do Aplicativo.

11.Após realizar a atividade informar indisponibilidade a próxima etapa é encerrar o processo.

12. Se [há rota ou destino atende ao pedido inicial] , a próxima etapa é esperar as atividades em paralelo serem executadas.
13. Após realizar a atividade Verificar tipo de veiculo a próxima etapa é uma decisão chamada de ponto de decisão 3.
14. Se [não há tipo de veículo adequado ao tipo de carga] , a próxima atividade é a de informar indisponibilidade.
15. Ir no passo 11.
16 . Se [há tipo de veículo adequado ao tipo de carga] , a próxima etapa é esperar as atividades em paralelo serem executadas.
17. Após as atividades Verificar rota e destinos e Verificar tipo de veiculo serem finalizadas a próxima atividade é a de confirmar disponibilidade que está na raia do Transportador.
18. Após realizar a atividade confirmar disponibilidade a próxima etapa é uma decisão.
19. Se [o motorista não atende a solicitação positivamente] o processo é encerrado.
19. Se [o motorista atende a solicitação positivamente] a próxima atividade é a de calcular valor e tempo de espera que está na raia do Aplicativo.
20. Após realizar a calcular valor e tempo de espera , a próxima atividade é a de informar dados do caminhão, valor e tempo de espera que está na raia do Aplicativo.
21. Após realizar a de informar dados do caminhão, valor e tempo de espera , a próxima atividade é a de Preparar a carga para transporte que está na raia do Empresa que precisa do transporte.
22. Após realizar a de Preparar a carga para transporte , finaliza o processo.

Em dotUml :

```
StateDiagram [frame=true framecolor=steelblue label="State Diagram"
    splines=ortho]{
/* Atividades da Empresa que precisa do transporte */

 state s2 as "Informar origem, destino e tipo de carga"
 state s3 as "Preparar a carga para transporte"

/* Atividades do Aplicativo */

 state s4  as "Verificar veículos disponíveis"
 state s5 as "Verificar rota e destinos"
 state s6 as "Verificar tipo de veiculo"
 state s7 as "informar indisponibilidade"

 state s8 as "calcular valor e tempo de espera"
 state s9 as "informar dados do caminhão"
 state s10 as "calcular valor e tempo de espera"
 state s11 as "Preparar a carga para transporte"
```

```
 /* Atividades do Transportador */
state s12 as  "confirmar disponibilidade"
 choice p1
 choice p2
 choice p3
 choice p4
 fork f
 join j

s2->s4
s4 -> p1 "Ponto de decisão 1"
p1 -> finalstate
p1 -> f "[Há veículos disponíveis]"
f -> s5
f -> s6
s5 -> p2  "Ponto de decisão 2"
s6 -> p3  "Ponto de decisão 3"
p2 -> s7
p3 -> s7
s7 -> finalstate
p2 -> j "[rota ou destino atende ao pedido]"
p3 -> j "[tipo de veículo é adequado ao tipo de carga]"
j -> s12
s12 -> p4
p4 -> finalstate
p4 -> s10 "[O motorista atende a solicitação positivamente]"
s10 -> s11
s11 -> s12
s3 -> finalstate

}
```

Com base no diagrama apresentado, avalie as afirmações a seguir

I. (item 1) Os serviços oferecidos por outros aplicativos, como mapas para cálculo de rotas, por exemplo, poderão ser usados no desenvolvimento do aplicativo, mesmo que não tenham sido representados no diagrama.

II. (item 2) Os fluxos paralelos indicados nos pontos de decisão 2 e 3 levam a erros no funcionamento do aplicativo, uma vez que permitam a continuidade do processo mesmo que não haja rota, destino ou tipo de veículo adequado.

III. (item 3) O diagrama apresenta atividades desempenhadas pela “Empresa que precisa de transporte” e pelo “Transportador”, no entanto, as atividades que precisarão ser modeladas no sistema restringem-se às desempenhadas pelo “Aplicativo”.

IV. (item 4) As ações “Informar indisponibilidade” e “Confirmar disponibilidade” correspondem a funcionalidades distintas no sistema, visto

que a primeira se relaciona à existência de tipo de veículo, rota e destino e a segunda se relaciona à agenda do transportador.

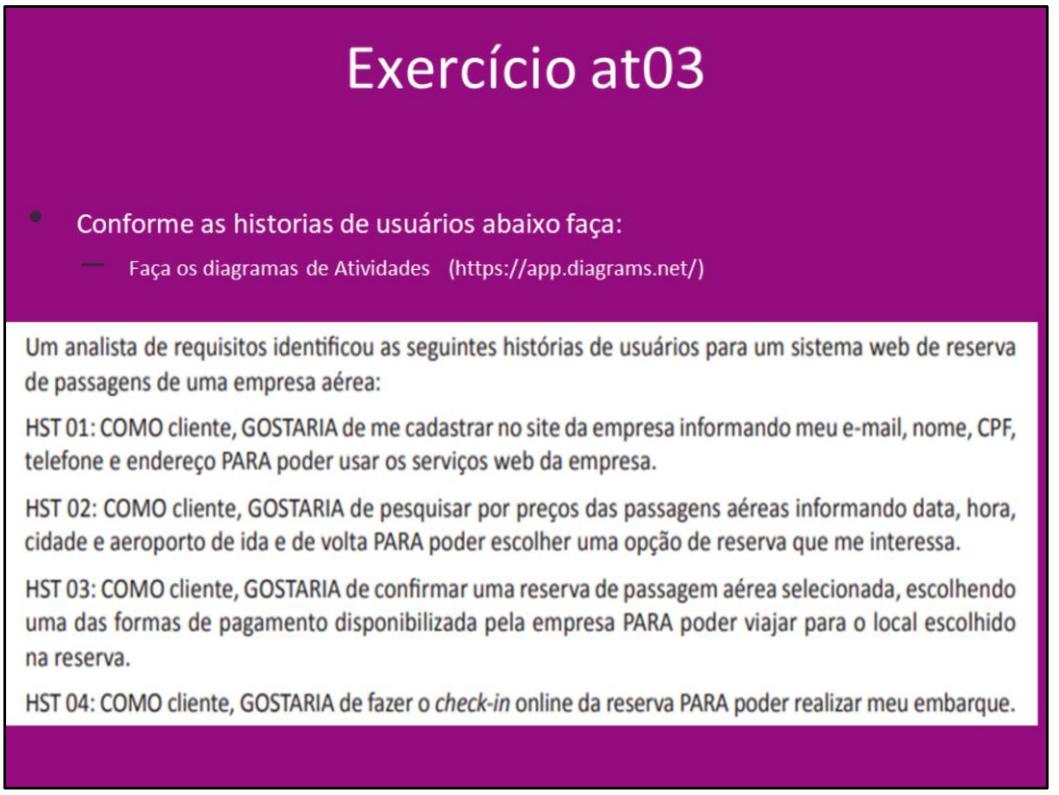
É correto apenas o se afirma em:
A – I e II (itens 1 e 2)
B – I e IV (itens 1 e 4)
C – II e III (itens 2 e 3)
D – I, III e IV (itens 1, 3 e 4)
E – II, III e IV (itens 2, 3 e 4)

Resposta

?

Gabarito

- b



#PraCegoVer

(QUESTAO ENADE) Um analista de requisitos identificou as seguintes histórias de usuários para um sistema web de reserva de passagens de uma empresa aérea:

HST 01: COMO cliente, GOSTARIA de me cadastrar no site da empresa informando meu e-mail, nome, CPF, telefone e endereço PARA poder usar os serviços web da empresa.

HST 02: COMO cliente, GOSTARIA de pesquisar por preços das passagens aéreas informando data, hora, cidade e aeroporto de ida e de volta PARA poder escolher uma opção de reserva que me

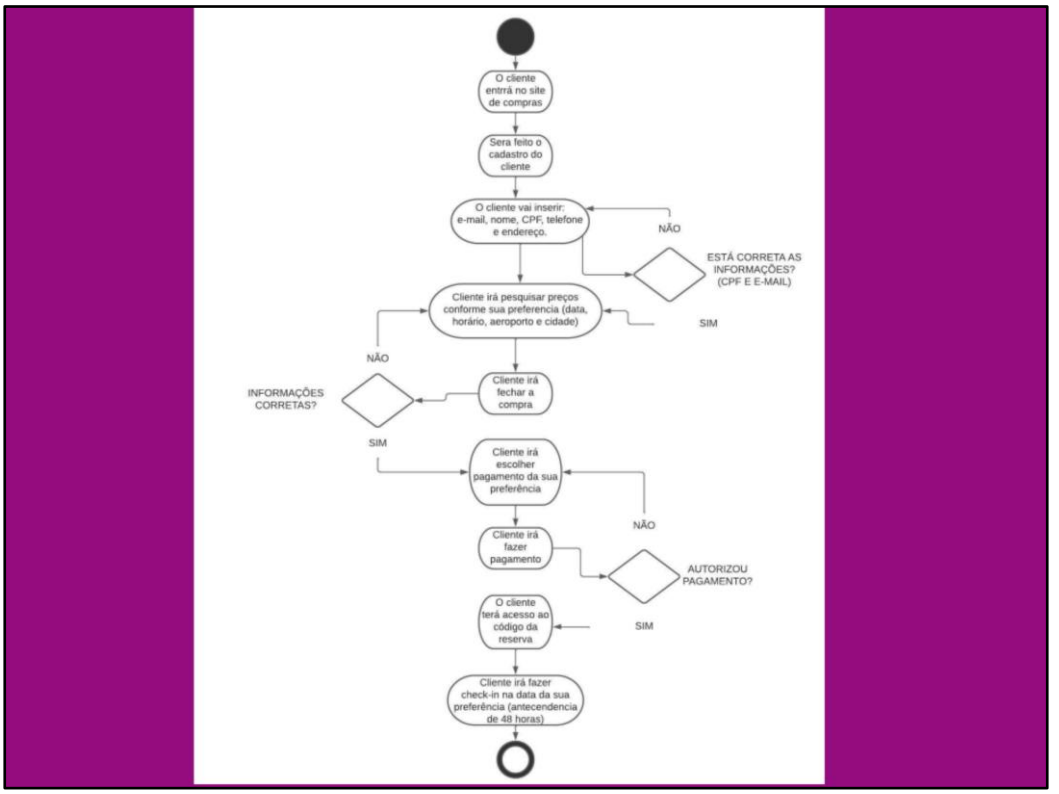
interessa.

HST 03: COMO cliente, GOSTARIA de confirmar uma reserva de passagem aérea selecionada, escolhendo uma das formas de pagamento disponibilizada pela empresa PARA poder viajar para o local escolhido na reserva.

HST 04: COMO cliente, GOSTARIA de fazer o check-in online da reserva PARA poder realizar meu embarque.

Resposta

?



#PraCegoVer

1.Diagrama de atividades da Venda de Passagens

2.Tem 9 atividades: O cliente entrará no site de compras, Será feito o cadastro do cliente, Inserir dados do cliente, Cliente pesquisa preços, Comprar passagem, Cliente escolhe forma de pagamento, Cliente faz pagamento, Cliente recebe código de reserva, Realizar Check-in.

3.Inicia com a atividade O cliente entrará no site de compras.

4.Após realizar a atividade de O cliente entrará no site de compras, a próxima atividade é a de Será feito o cadastro do cliente.

5. Após realizar a atividade de Será feito o cadastro do cliente, a próxima atividade é a de Inserir dados do cliente.

6. Após realizar a atividade de Inserir dados do cliente, a próxima etapa é uma decisão: Está correta as informações ? (CPF e E-mail?)

7. Se [não] volta para a atividade de Inserir dados do cliente (linha 6).

8. Se [sim], a próxima atividade é a de Cliente pesquisa preços.

9. Após realizar a atividade de Cliente pesquisa preços, a próxima atividade é a de Comprar passagem.

10. Após realizar a atividade de Comprar passagem, a próxima etapa é uma decisão: Informações corretas ?

11. Se [não] voltar para atividade Cliente pesquisa preços (linha 9)

12. Se [sim], a próxima atividade é Cliente escolhe forma de pagamento.

13. Após realizar a atividade de Cliente escolhe forma de pagamento, a próxima

atividade é a de Cliente faz pagamento.

Após realizar a atividade de Cliente faz pagamento, a próxima etapa é uma decisão: Autorizou pagamento ?

14. Se [não] volta para atividade Cliente escolhe forma de pagamento (linha 13)
15. Se [Sim] a próxima atividade é a de Cliente recebe código de reserva.
16. Após realizar a atividade de Cliente recebe código de reserva, a próxima atividade é a de Realizar Check-in.
17. Após realizar a atividade de Realizar Check-in, e finaliza o processo.

```
StateDiagram [frame=true framecolor=steelblue label="State
   Diagram" splines=ortho] {
  state O_cliente_entrar_no_site_de_compras
  state Sera_feito_o_cadastro_do_cliente
  state  Inserir_dados_do_cliente
  state  Cliente_pesquisa_preços
  state  Comprar_passagem
  state  Cliente_escolhe_forma_de_pagamento
  state  Cliente_faz_pagamento
  state  Cliente_recebe_código_de_reserva
  state  Realizar_Check_in
  choice esta_corretas_as_informacoes
  choice informacoes_corretas
  choice autorizou_pagamento


  initialState->O_cliente_entrar_no_site_de_compras
  O_cliente_entrar_no_site_de_compras ->
   Sera_feito_o_cadastro_do_cliente
  Sera_feito_o_cadastro_do_cliente ->
   Inserir_dados_do_cliente
 Inserir_dados_do_cliente -> esta_corretas_as_informacoes
 esta_corretas_as_informacoes -> Inserir_dados_do_cliente
   "[Não]"
 esta_corretas_as_informacoes -> Cliente_pesquisa_preços
   "[Sim]"
 Cliente_pesquisa_preços -> Comprar_passagem
 Comprar_passagem -> informacoes_corretas
 informacoes_corretas -> Cliente_pesquisa_preços "[Não]"
 informacoes_corretas -> Cliente_escolhe_forma_de_pagamento
   "[Sim]"
 Cliente_escolhe_forma_de_pagamento ->
   Cliente_faz_pagamento
 Cliente_faz_pagamento -> autorizou_pagamento
 autorizou_pagamento -> Cliente_escolhe_forma_de_pagamento
   "[Não]"
 autorizou_pagamento -> Cliente_recebe_código_de_reserva
```

```
  "[Não]"
Cliente_recebe_código_de_reserva -> Realizar_Check_in
Realizar_Check_in -> finalstate

  }
```

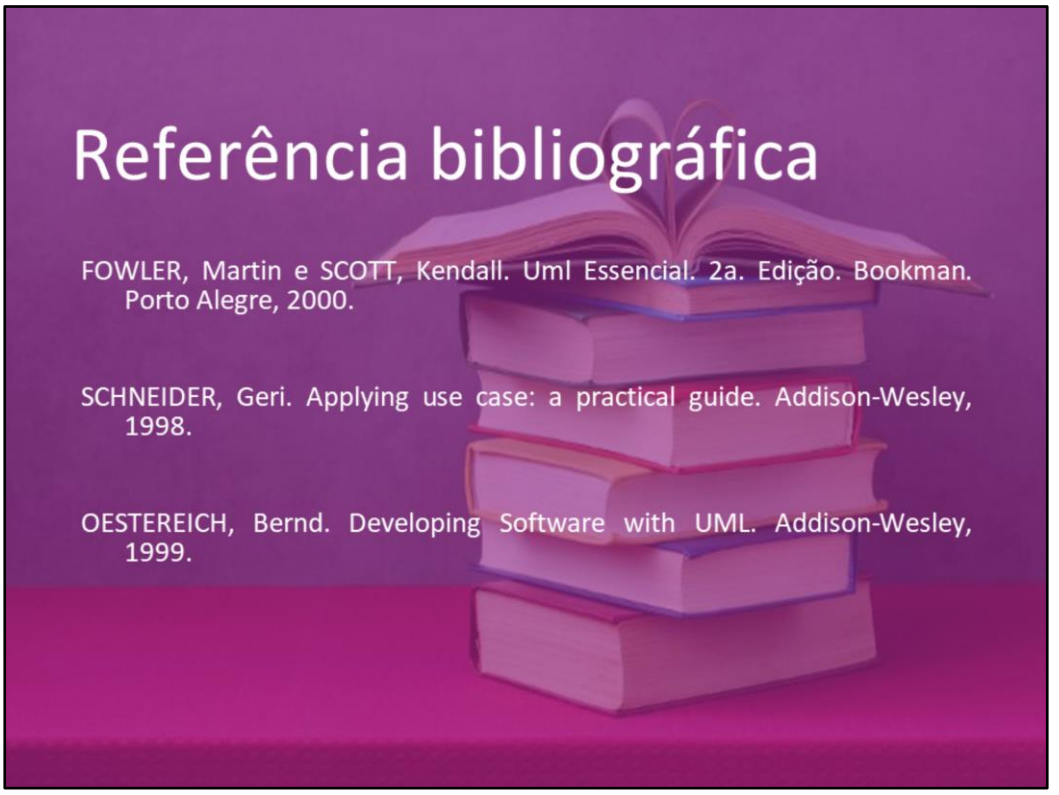
Exercício at04

Escopo do projeto

Professor de um curso acessa o sistema para cadastrar a prova. Sempre que uma prova é cadastrada, o sistema envia um email para o secretário e um email para o monitor. O secretário é responsável por imprimir a prova e informar, através do sistema, que realizou a tarefa. O monitor é responsável por fazer o gabarito da prova e informar ao sistema que realizou a tarefa. Sempre que o sistema é informado do término de uma das tarefas, ele informa ao professor. Enquanto o secretário imprime as provas, o monitor faz o gabarito. Em seguida, o professor aplica a prova. É responsabilidade do professor corrigir a prova e informar a nota ao aluno. Se o aluno desejar fazer uma revisão, o monitor é acionado para fazer a revisão da prova. Se a nota for alterada, o professor é informado pelo monitor. Ao final do processo, o professor lança a nota no sistema.


INTERVALO

Referência bibliográfica



FOWLER, Martin e SCOTT, Kendall. Uml Essencial. 2a. Edição. Bookman. Porto Alegre, 2000.

SCHNEIDER, Geri. Applying use case: a practical guide. Addison-Wesley, 1998.

OESTEREICH, Bernd. Developing Software with UML. Addison-Wesley, 1999.


ânima
EDUCAÇÃO
UNISUL
CRÉDITOS
COORDENAÇÃO
Vera Rejane Niedersberg Schuhmacher
PROFESSORES
Rafael Lessa
Daniella Vieira