

# Aula 18

## Professor:

Geraldo Xexéo  
DCC/IM/UFRJ  
PESC/COPPE/UFRJ

## Conteúdo:

**Diagramas de  
Atividade: UML 2.0**

# Diagramas de Atividade

➡ O Diagrama de Atividade é uma das formas que UML propõe para modelar os aspectos dinâmicos de um sistema,

— Comportamento

➡ Um tipo de "fluxograma" mostrando como o controle flui entre as ações que compõe uma atividade.

# Versões 1.5 e 2.0

➡ A versão mais moderna de Diagramas de Atividade é a 2.0

⇒ 2.1.1

➡ A versão 1.5 é suportada pela maioria das ferramentas CASE

⇒ Algumas ainda estão em UML 1.3

# Versão 1.5

➡ Na versão 1.5, um diagrama de atividades era o mesmo que uma máquina de estados.

➡ Essa semântica é menos interessante para nós do que a atual.



# Exemplo de DA (UML 2.0)

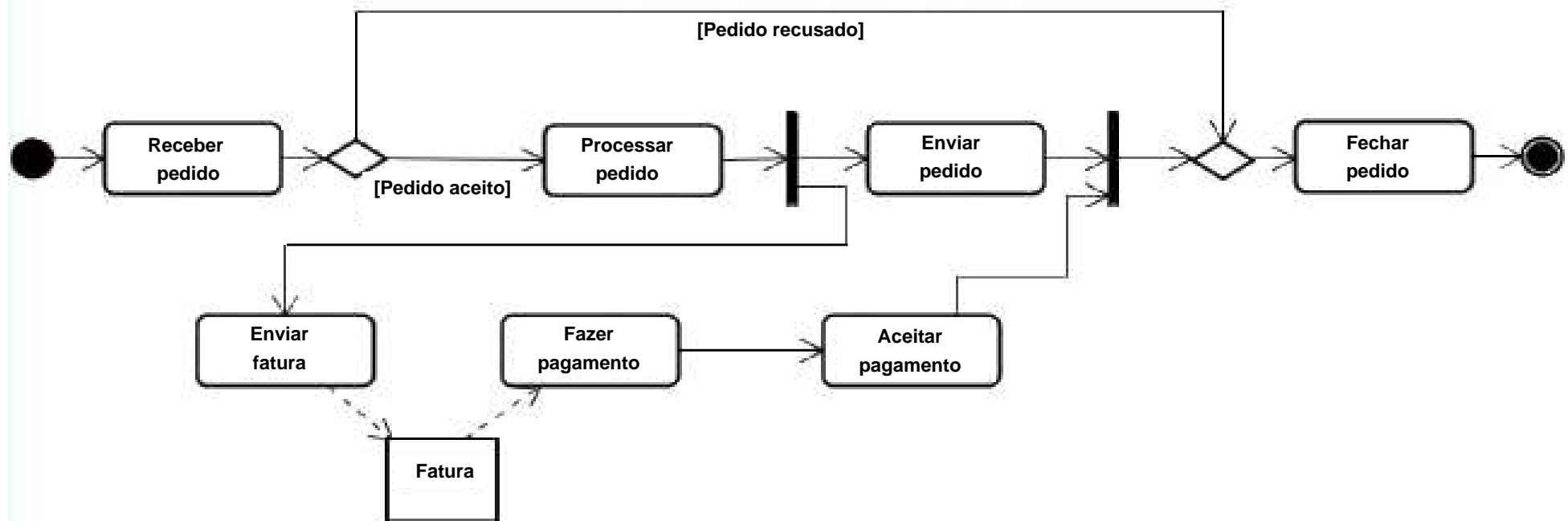
➡ O exemplo ao lado  
é exatamente igual  
a um exemplo da  
versão 1.5

➡ O significado  
"prático" é o  
mesmo

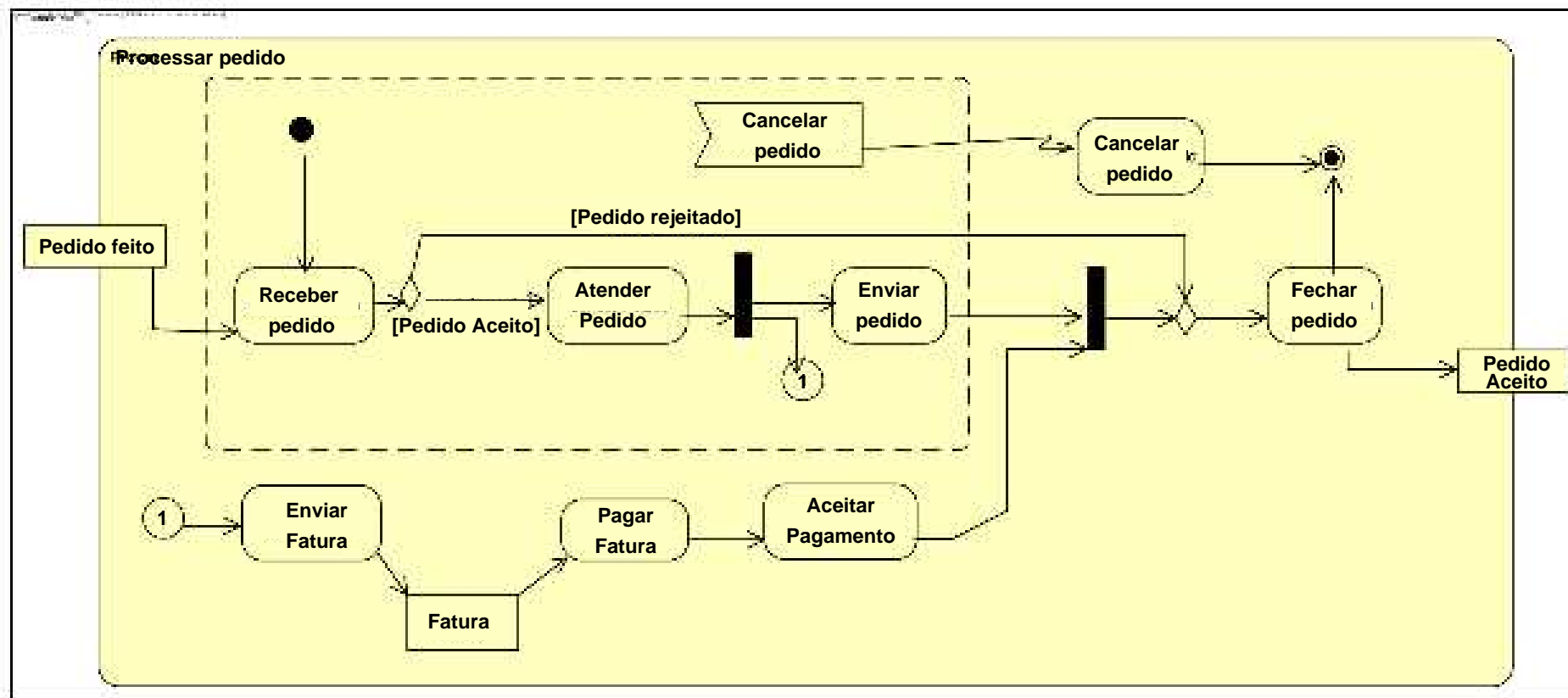
➡ Mas...



# Exemplo Complexo de DA

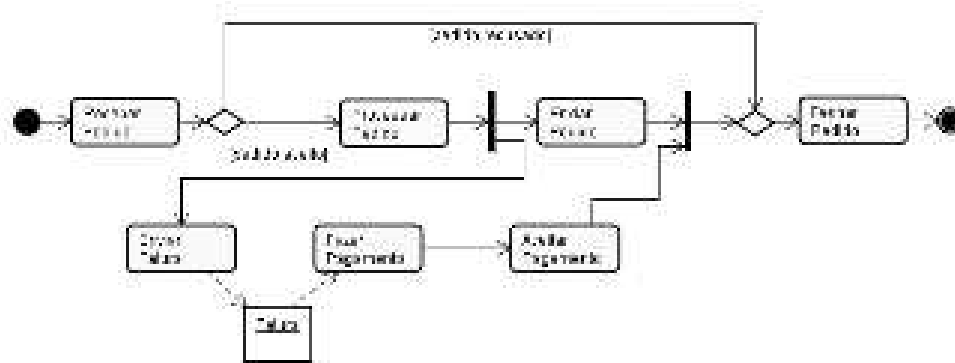


# Outro Exemplo + UML 2.0

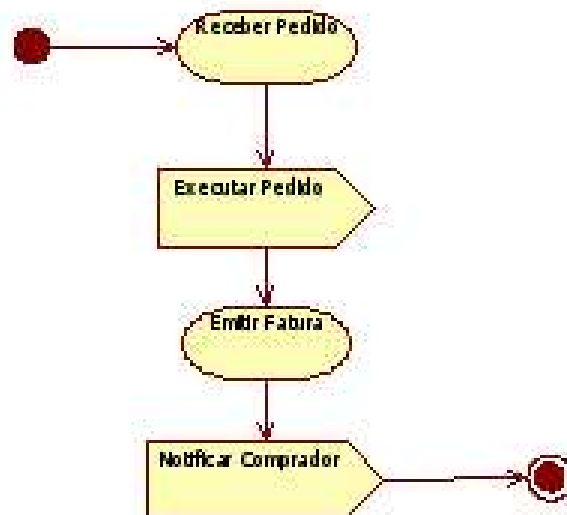


# Ferramentas CASE (1/2)

➡ Jude Community 3.1.1 - UML 1.5 (+)



➡ StarUML 5.0.2.1570 - Ainda não é 2.0

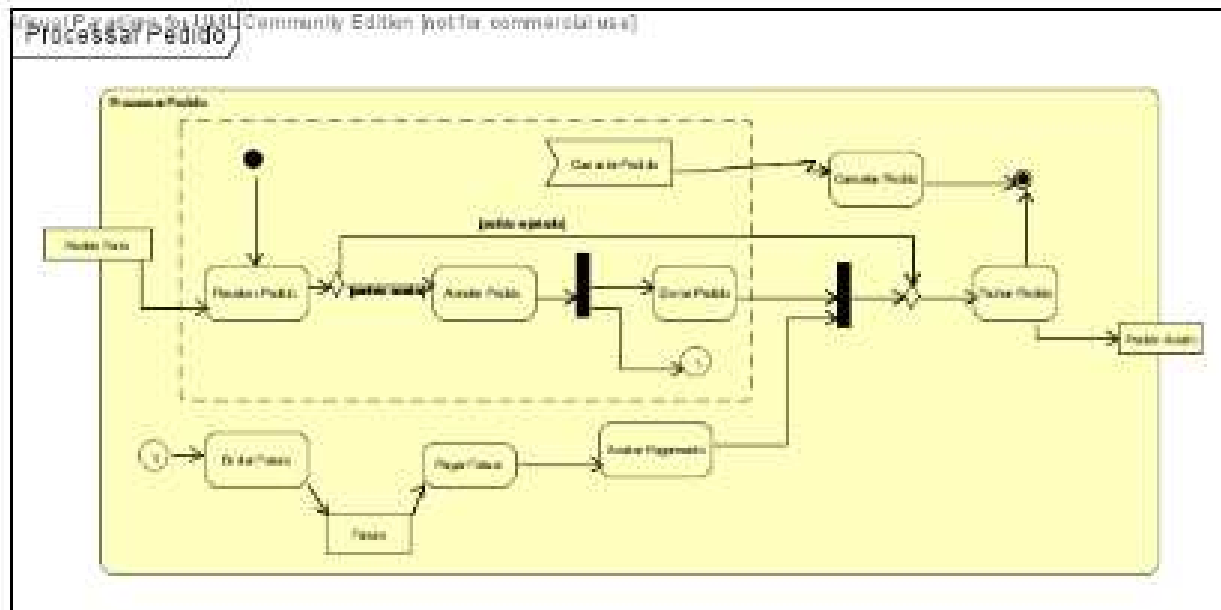




# Ferramentas CASE (2/2)

➡ Visual Paradigm CE 6.0 -

— Falta muito pouco para UML 2.0

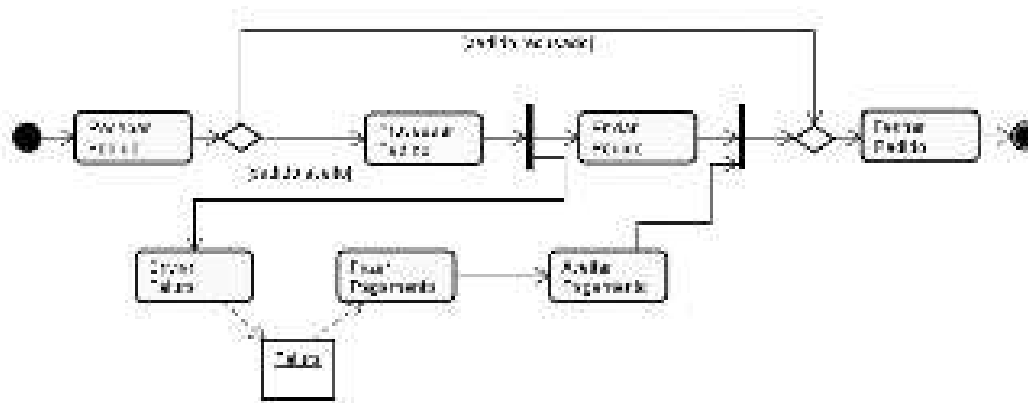


# Outras Ferramentas CASE UML

- ➡ IBM Rational Rose 7.0 (UML 1.5!)
- ➡ IBM Rational Software Modeller 7.0
- ➡ Borland Together
- ➡ Visual Paradigm
- ➡ Magic Draw
- ➡ Poseidon e Apollo for Eclipse
- ➡ OMondo for Eclipse (free)
- ➡ Dezenas de outras....

# DA seqüência Ações

➡ Em um diagrama de atividades, cada atividade é modelada como uma seqüência de ações simples.



➡ As ações são simples, não a seqüência

# Composição do Diagrama

## Nós

- ▬ Vários tipos de nós
- ▬ Indicam ação, controle e objetos

## Arestas

- ▬ Poucos tipos (2)
- ▬ Indicam o fluxo

# Nós do Diagrama de Atividade

➡ Ações

➡ Nós de Controle

- ➡ Nós abstratos

- ➡ Vários subtipos

➡ Objetos

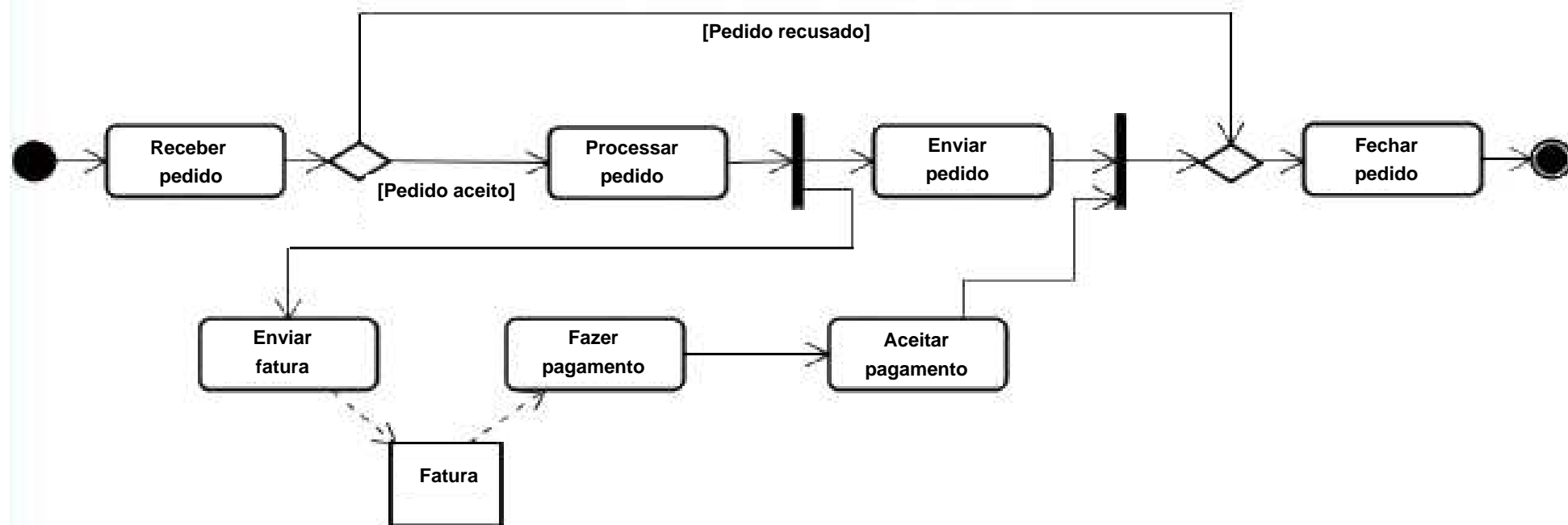
# Arestas do Diagrama de Atividade

- ➡ Fluxo de Controle
- ➡ Fluxo de Dados/Objetos
- ➡ Ambas modeladas por setas em UML 2.0
  - ➡ UML 1.5 usava setas tracejadas para fluxo de objetos/dados

# Semântica do Diagrama

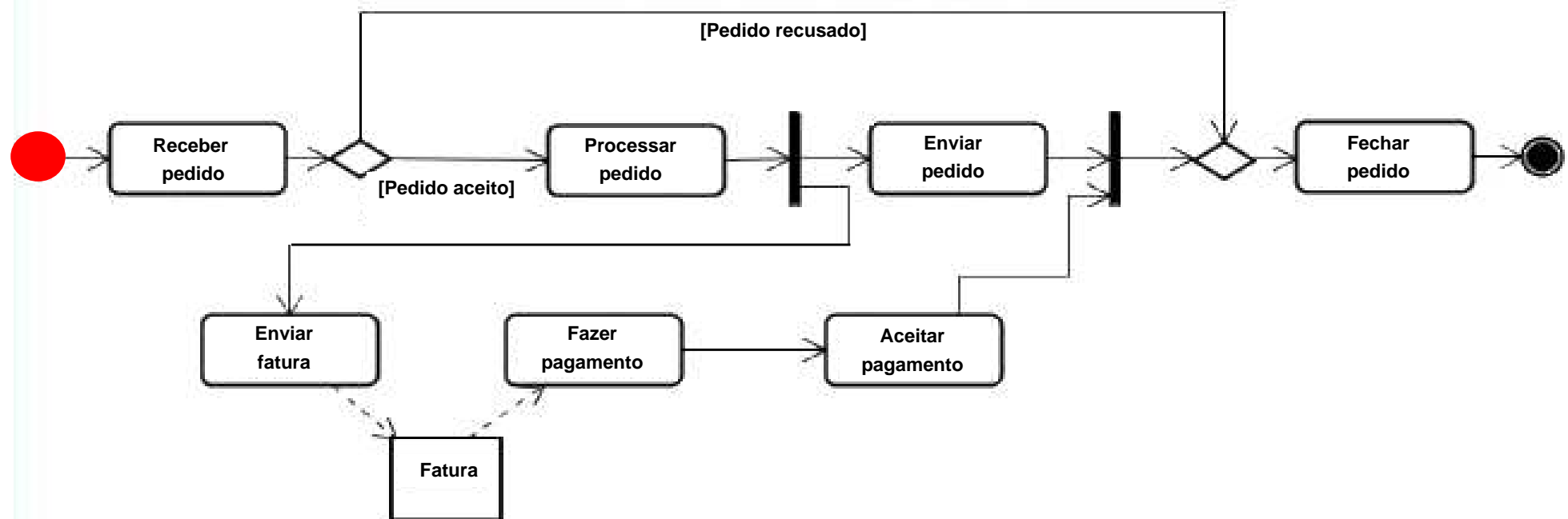
- ➡ "Semântica de fluxo de tokens"
- ➡ Tokens são marcações que andam pelos nós e arestas, indicando quais nós estão ativos
- ➡ Existem tokens de controle e tokens de dados

# Tokens andando

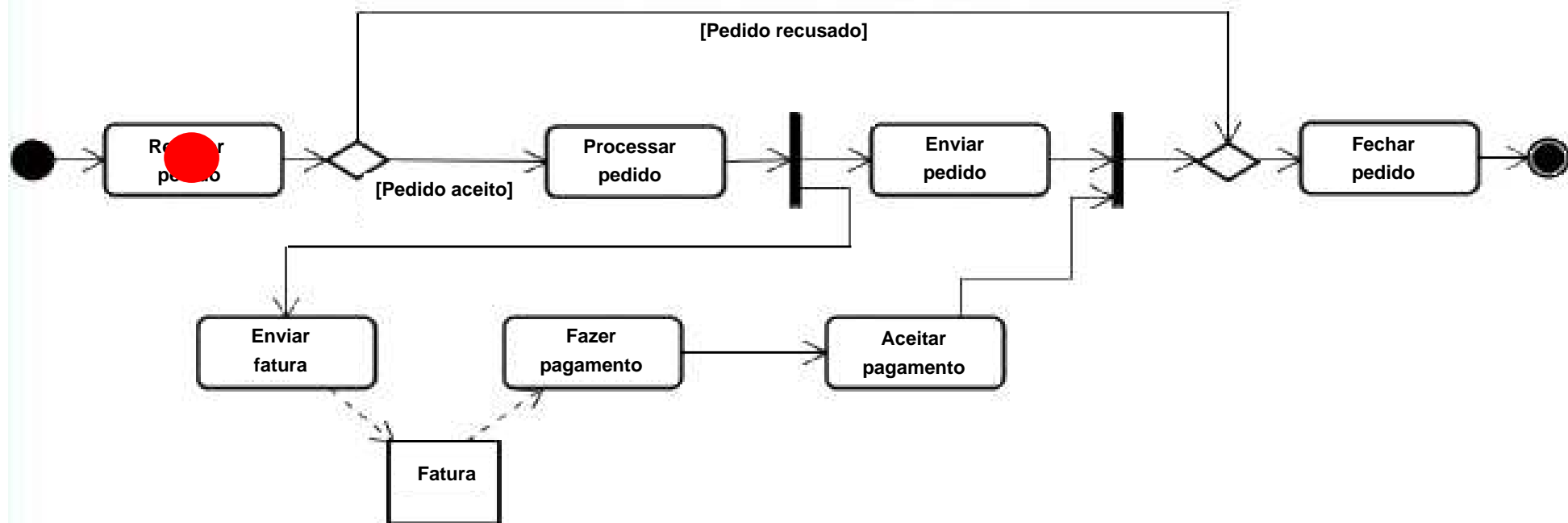




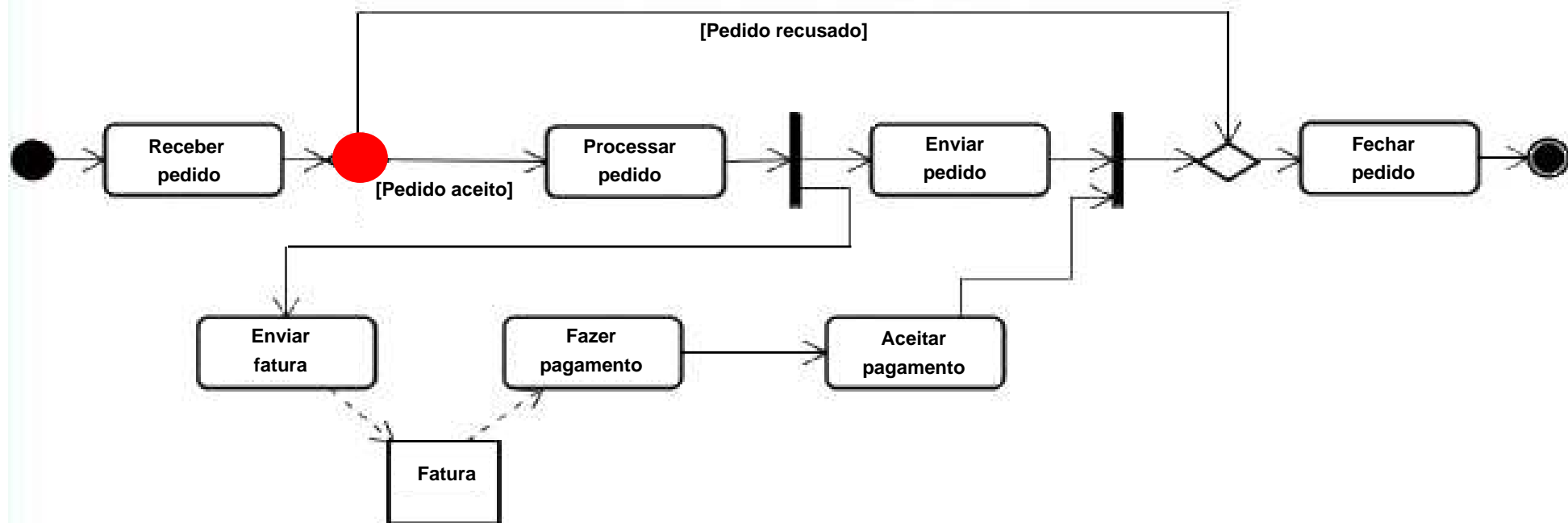
# Tokens andando



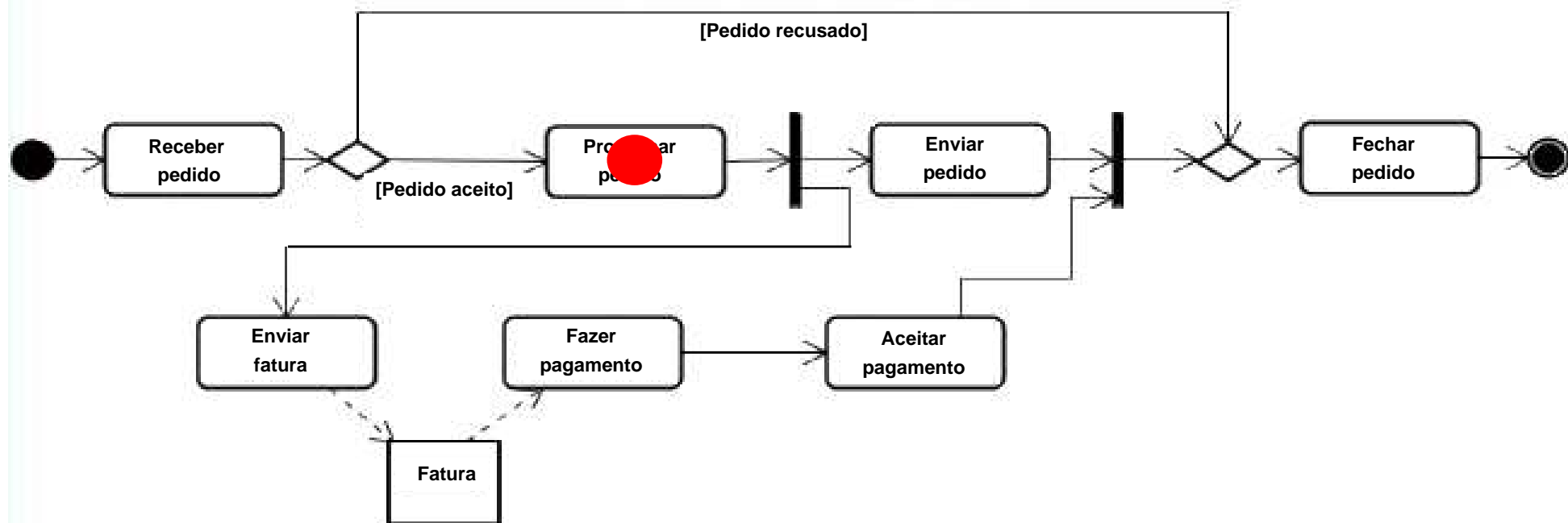
# Tokens andando



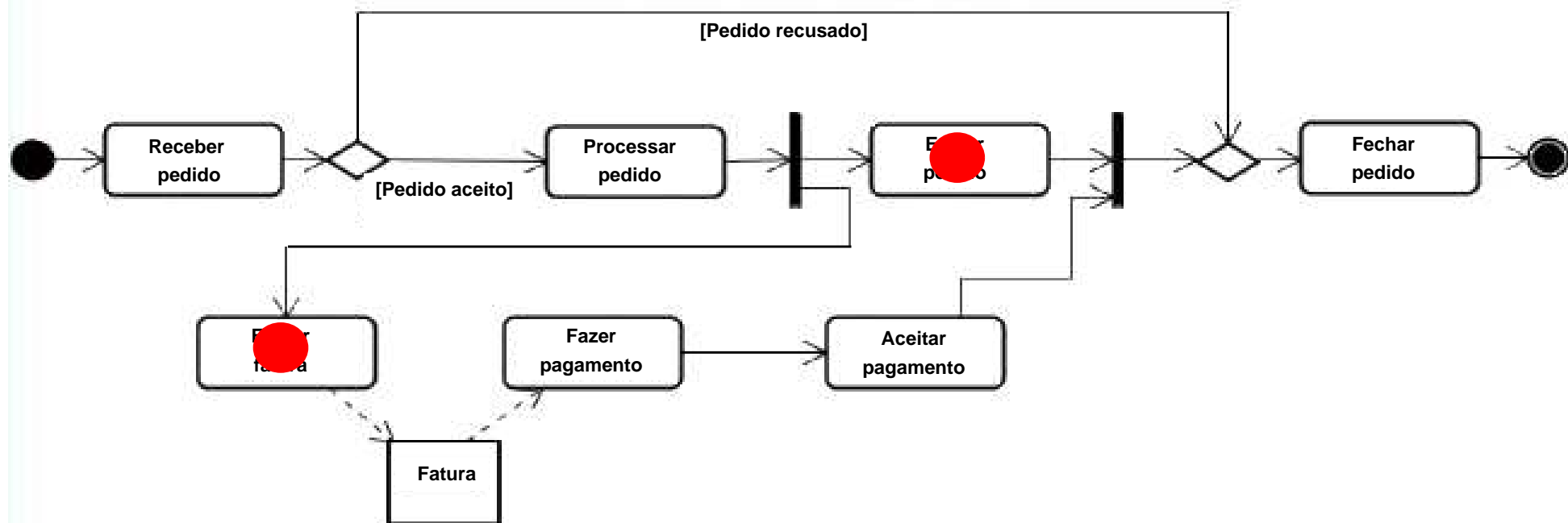
# Tokens andando



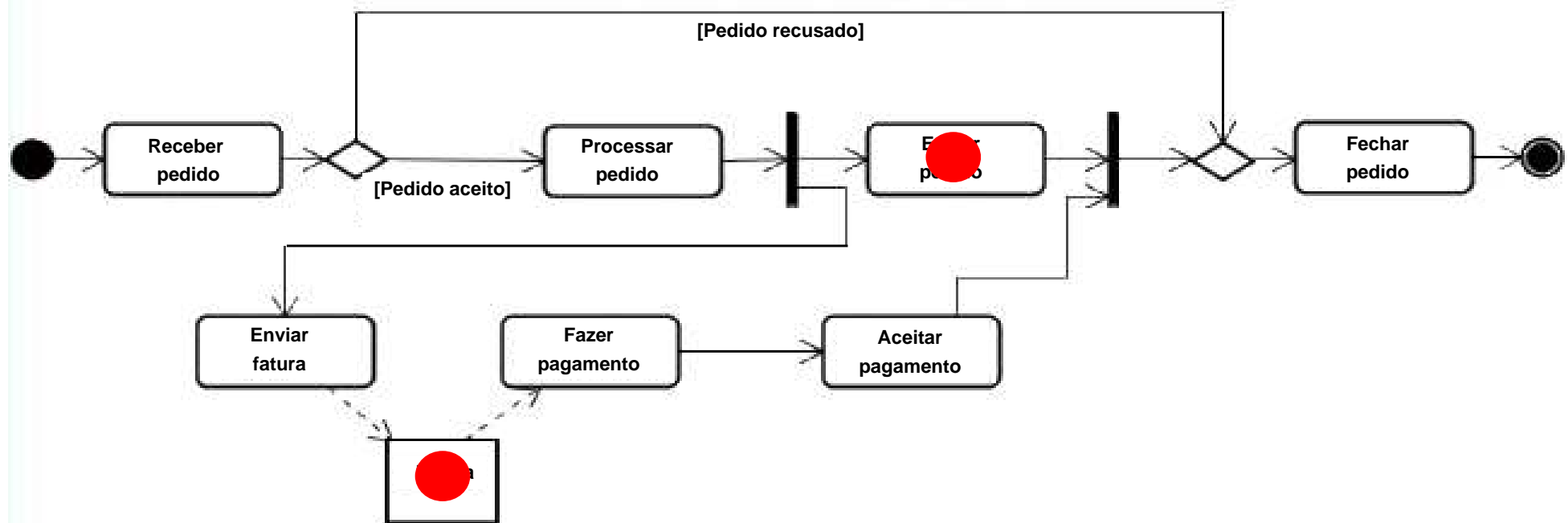
# Tokens andando



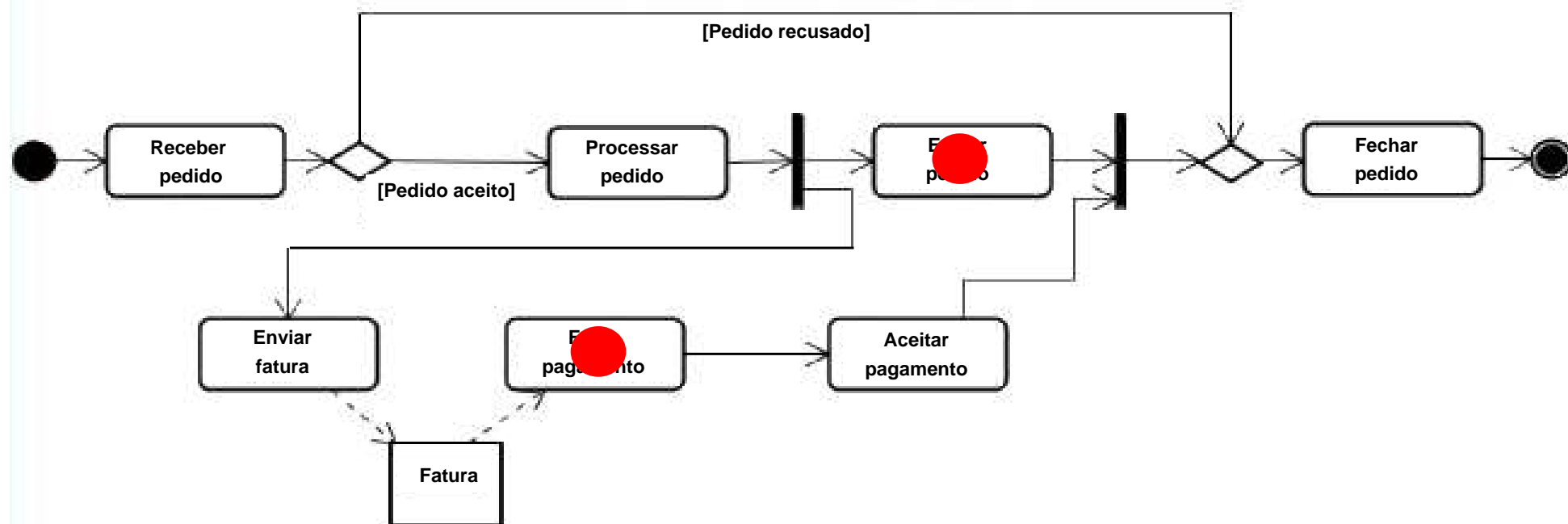
# Tokens andando



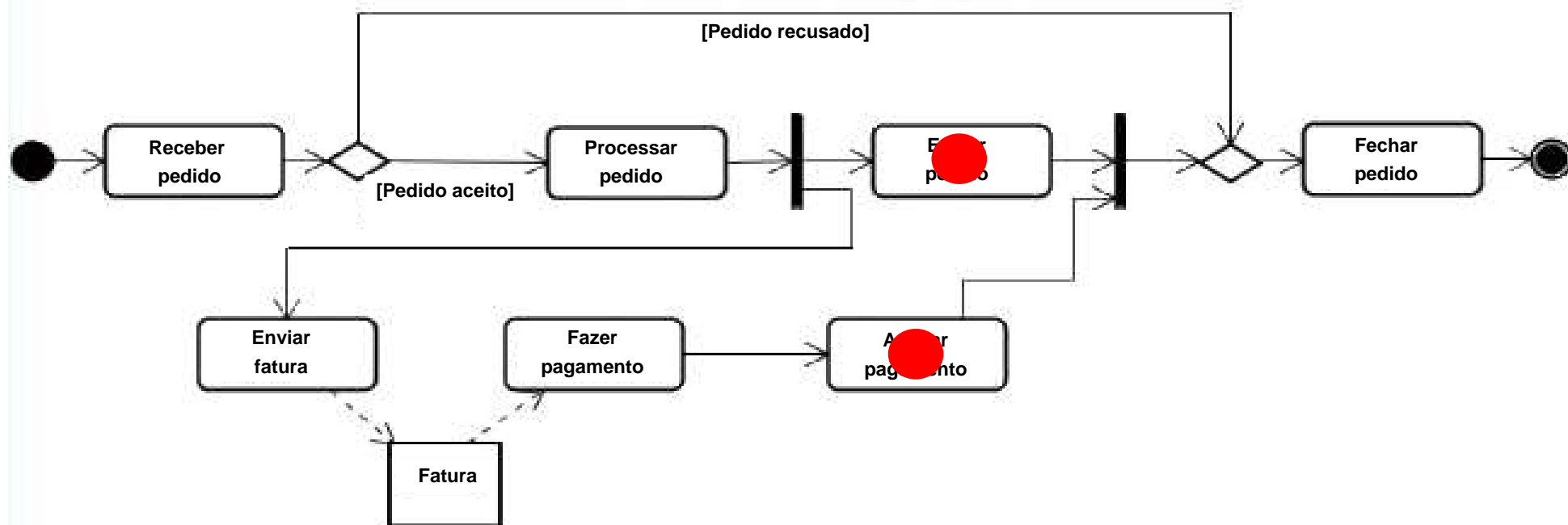
# Tokens andando



# Tokens andando

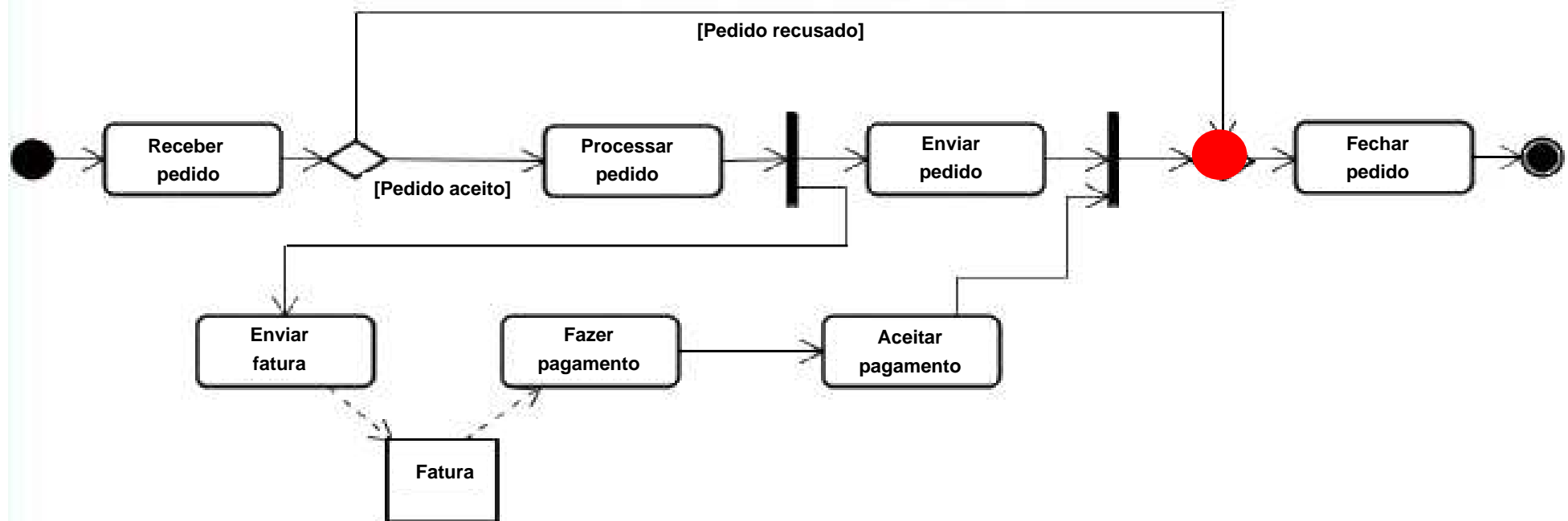


# Tokens andando

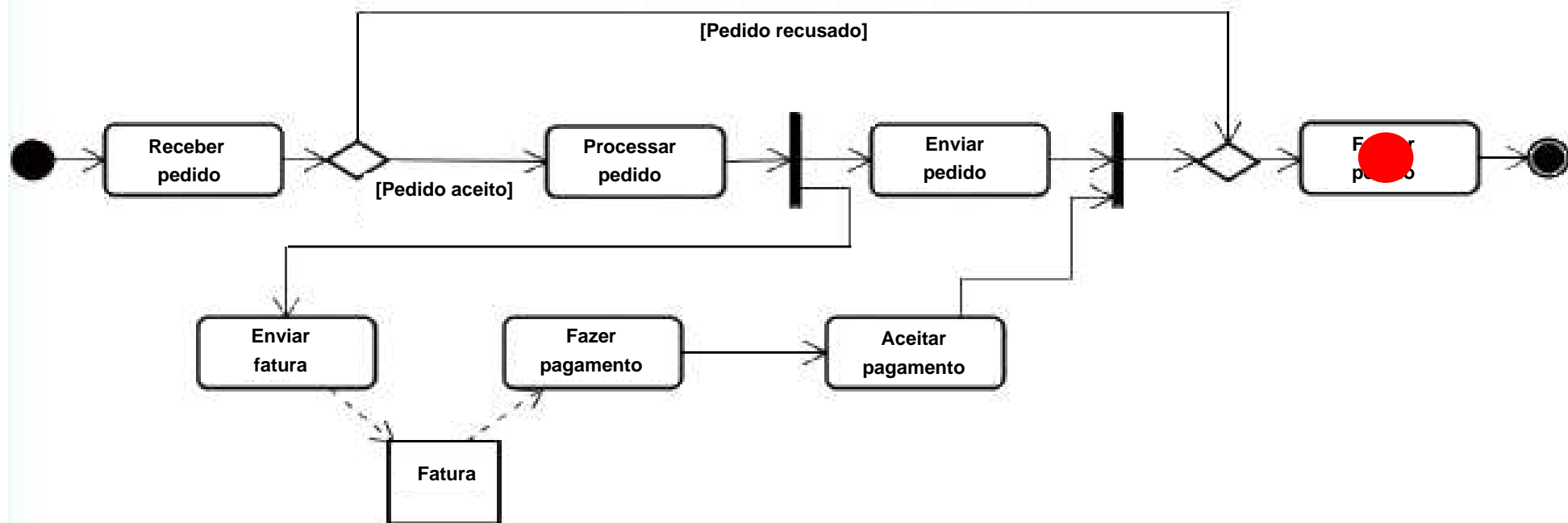




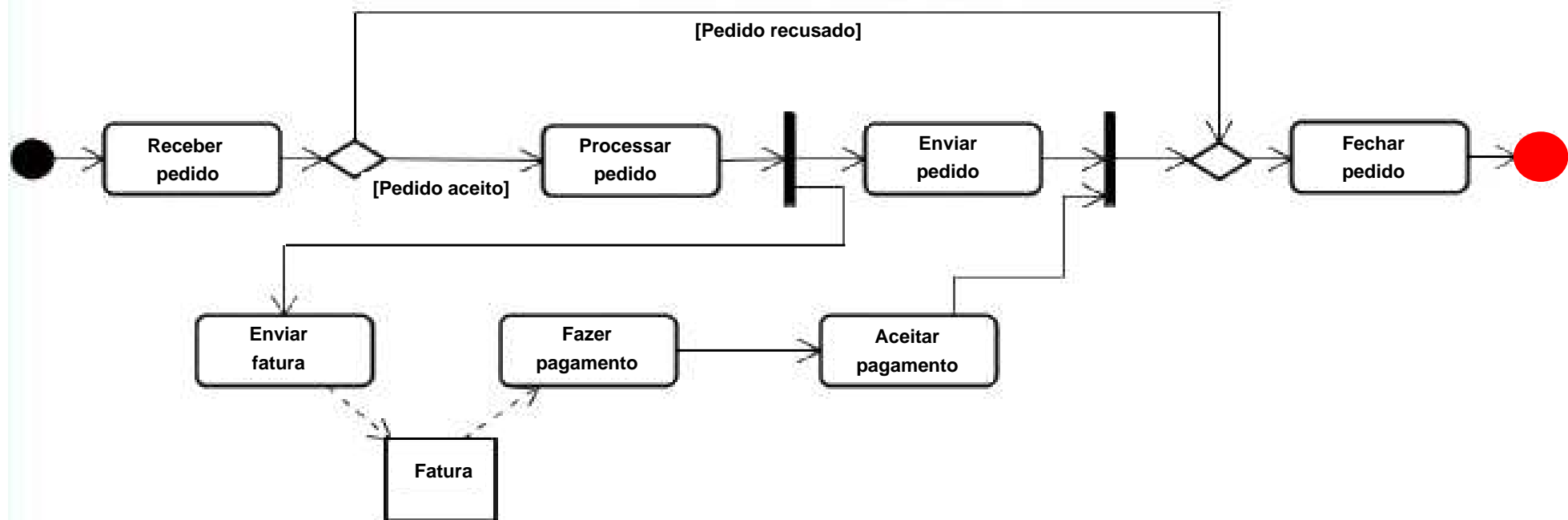
# Tokens andando



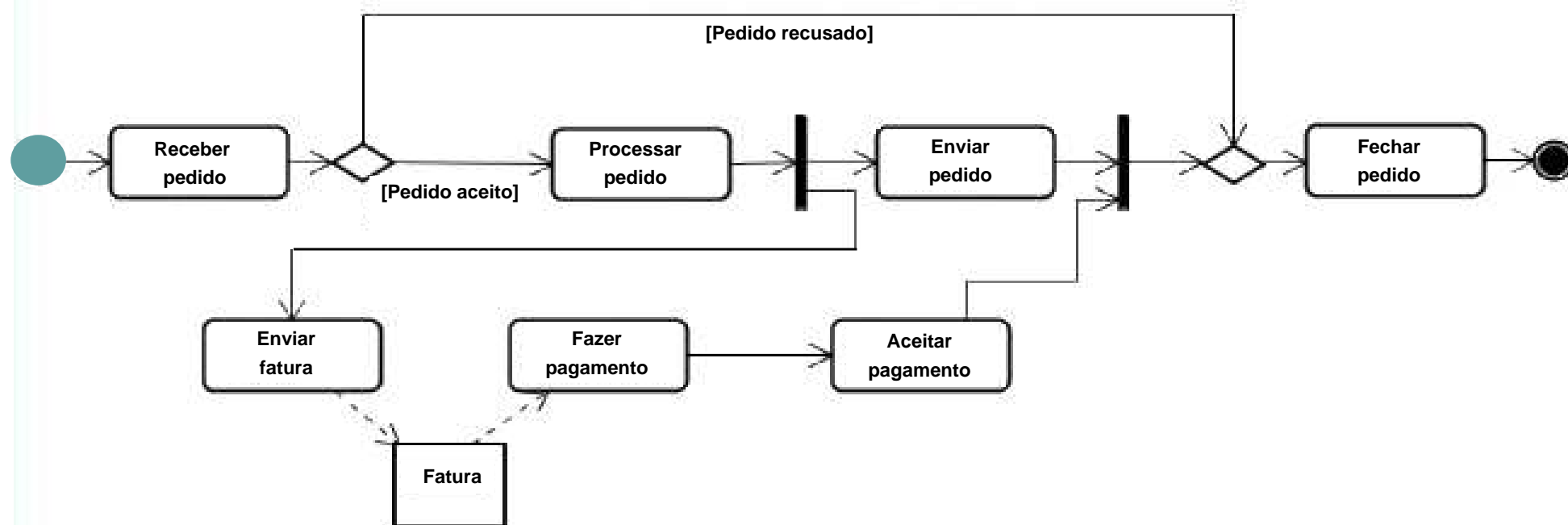
# Tokens andando



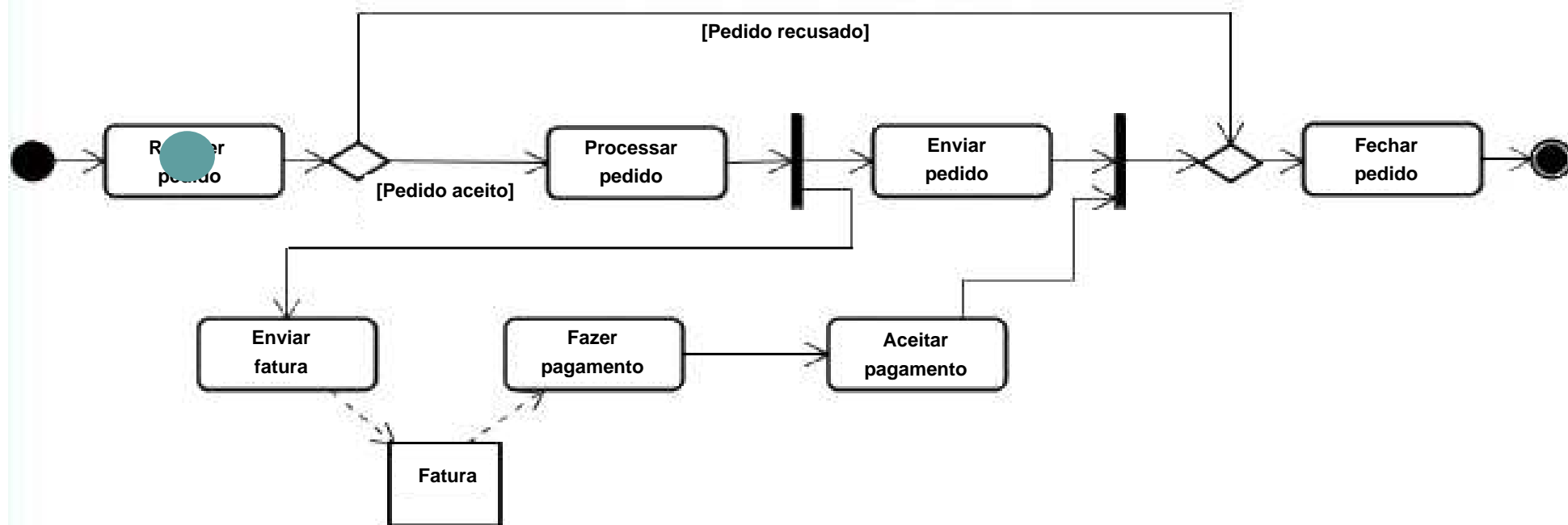
# Tokens andando



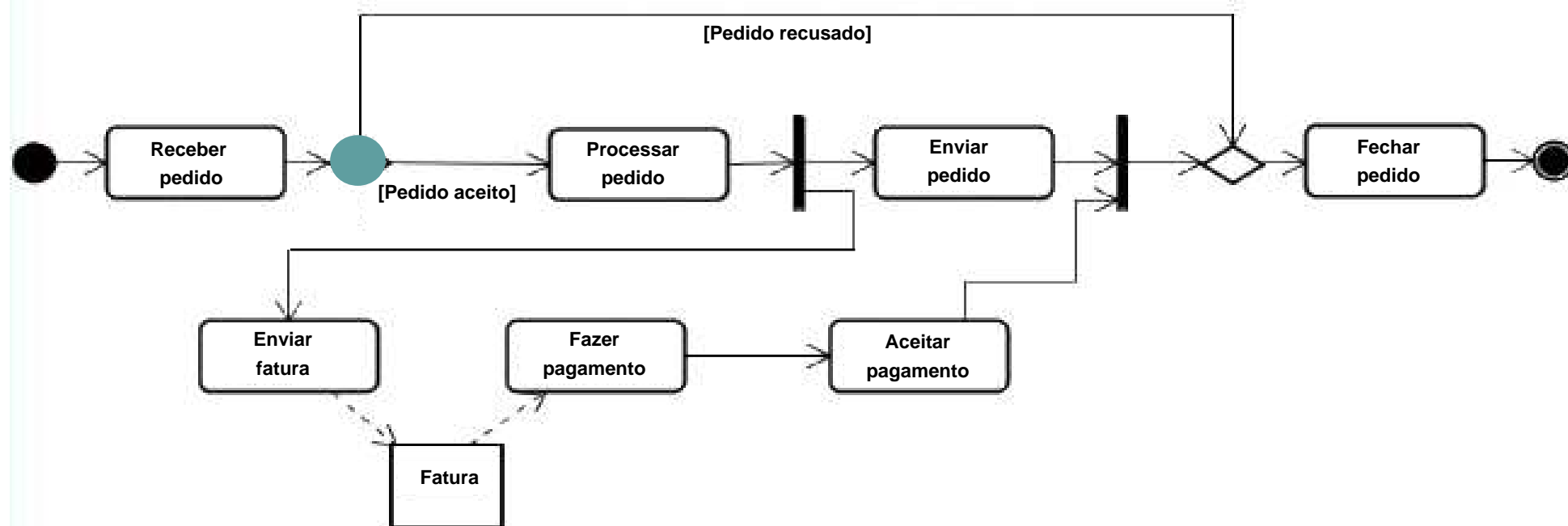
# Tokens andando



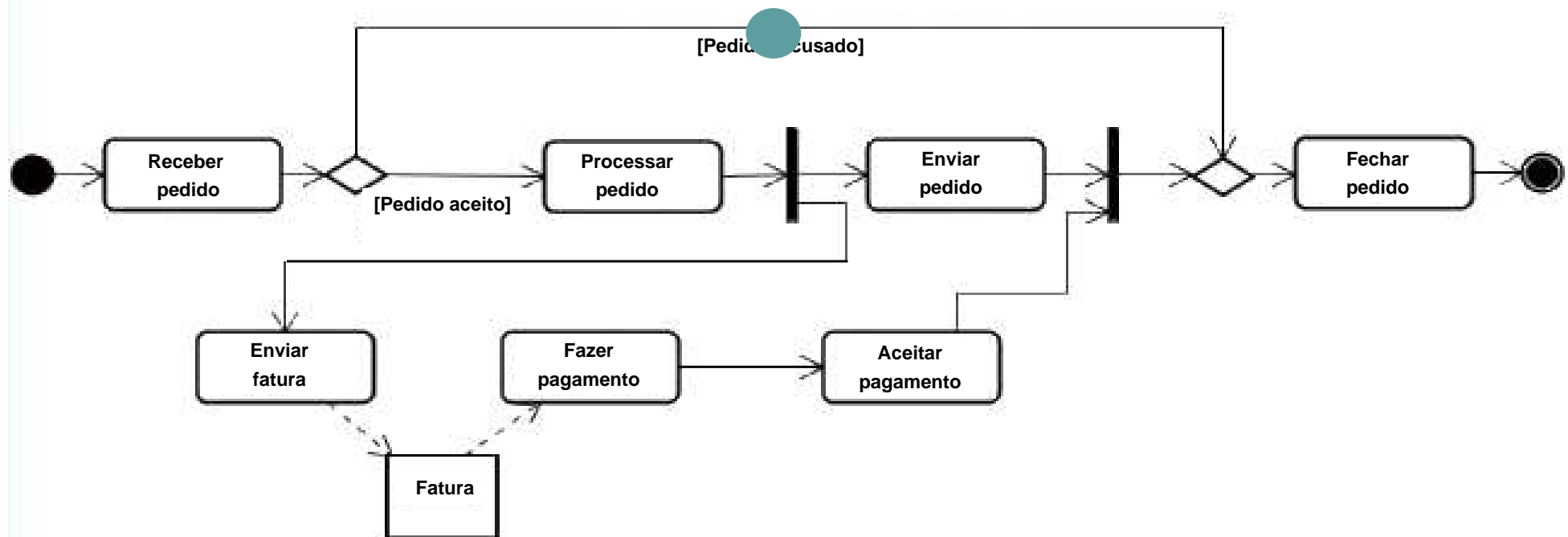
# Tokens andando



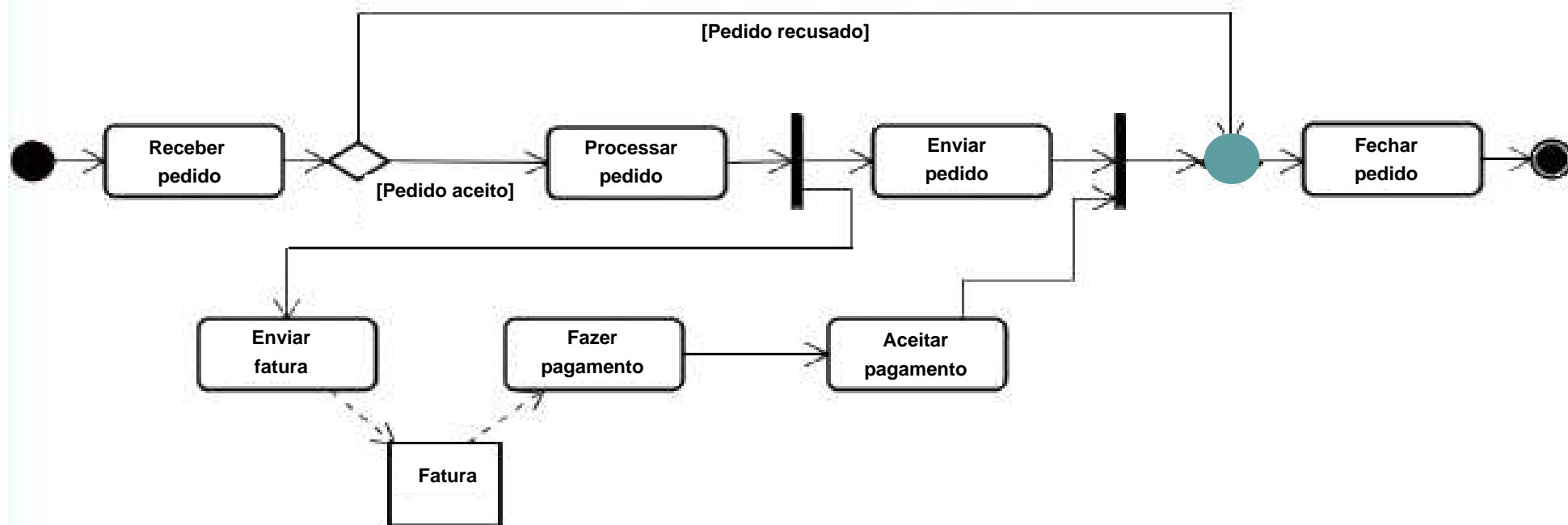
# Tokens andando



# Tokens andando

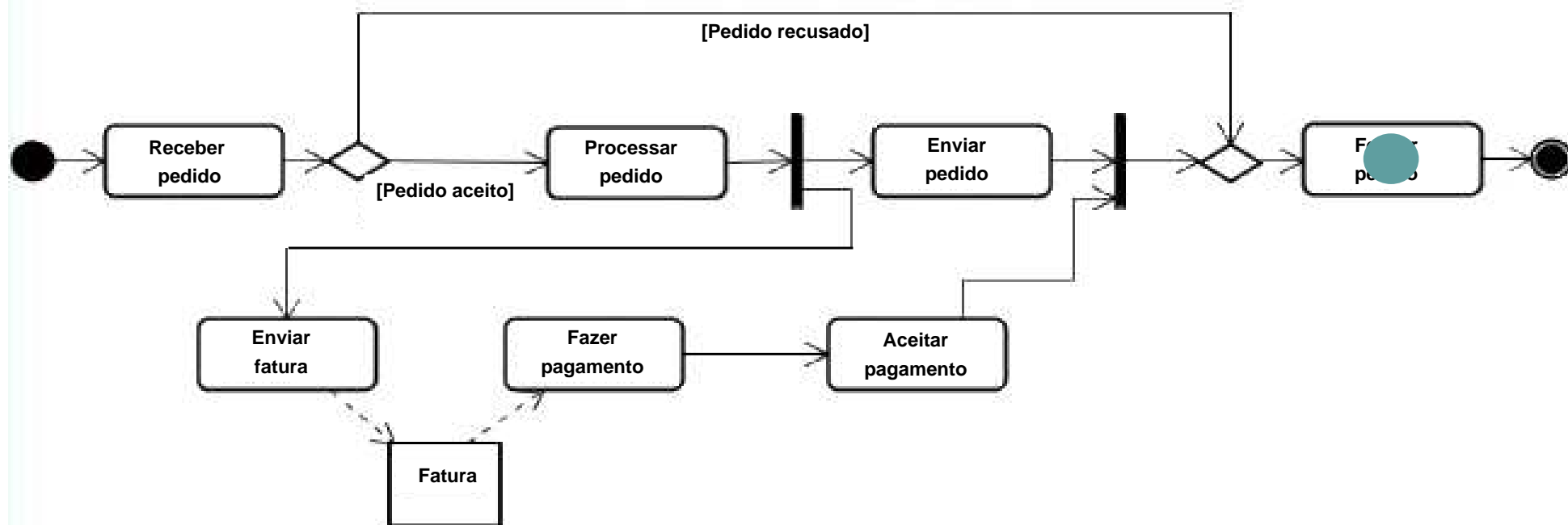


# Tokens andando

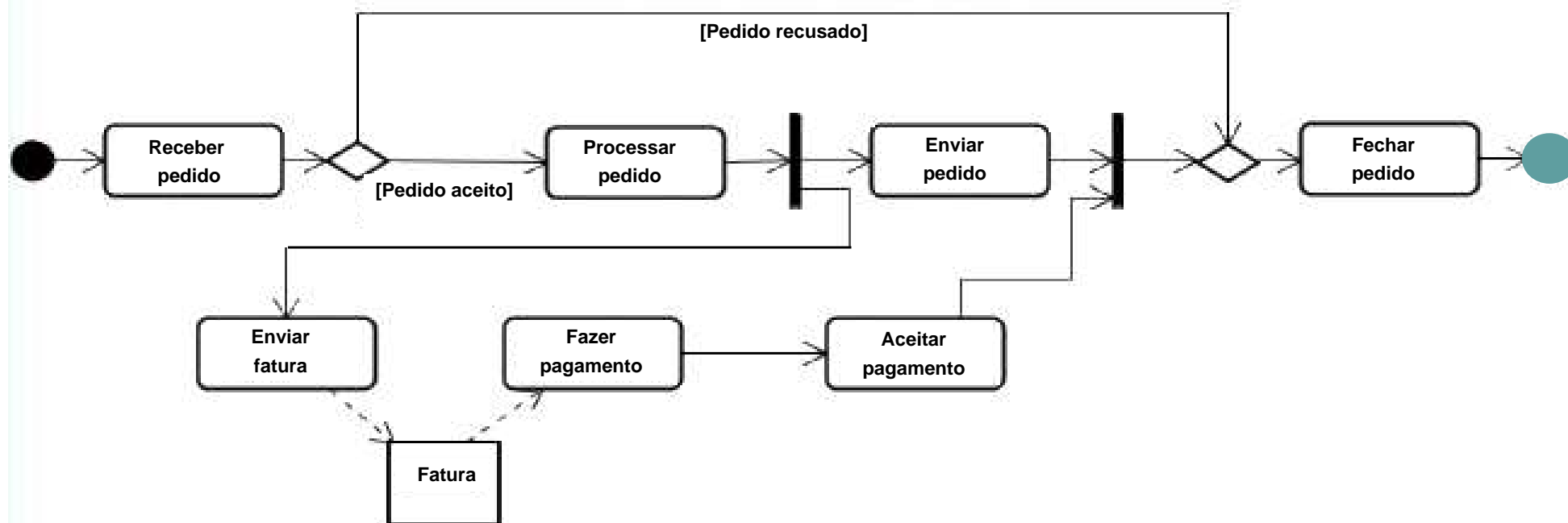




# Tokens andando

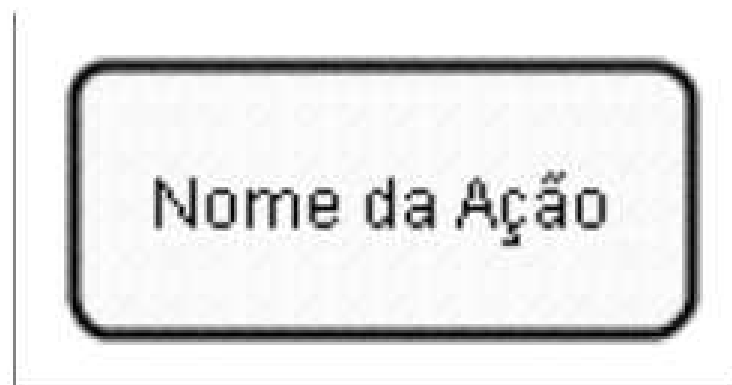


# Tokens andando



# Ação (1/3)

- ➡ Uma ação é a unidade básica de descrição de comportamento
- ➡ Uma ação converte um conjunto de entradas em um conjunto de saídas
  - ▢ Quaisquer dos conjuntos pode ser vazio



## Ações (2/3)

➡ Existem em UML vários tipos especializados e ações

- ▢ Detalhe da UML que não se aplica ao curso de forma geral
  - ▢ Como SendSignalAction, que veremos ainda
  - ▢ Como CallAction, que invoca um comportamento e retorna valores, e não veremos mais nesse curso


# Ações (3/3)

➡ Ações estão sempre contidas em comportamentos  
(DAs, por exemplo)

- ▬ Comportamentos fornecem contexto às ações
  - ▬ Quando executam
  - ▬ Que entradas exigem

# Tipos Básicos de Ações

## Chamada de Operações

-  As operações são executadas dinamicamente, devido ao polimorfismo existem no modelo OO





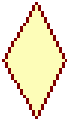
## Envio de Sinais

## Invocação direta de comportamentos

# Nós de Controle

- ➡ São Abstratos
- ➡ Coordenam o fluxo da atividade

# Tipos de Nós de Controle

- ⇒ Nó Inicial 
- ⇒ Nó Final
  - ⇒ Fim de Atividade 
  - ⇒ Fim de Fluxo 
- ⇒ Nó de "fork" (divisão de fluxo)
- ⇒ Nó de "join" (união e sincronização de fluxos) 
- ⇒ Nó de decisão (escolha entre fluxos)
- ⇒ Nó de "merge" (união de fluxos disjuntos) 



# Nó Inicial



Um DA pode ter um ou mais nós iniciais

- ⇒ Não são necessários
- ⇒ Cada nó inicial indica um fluxo (thread)
- ⇒ Todos os nós iniciais de uma atividade são invocados no seu início
- ⇒ Indicado por um círculo negro

# Nó de Fim de Atividade



- ➡ O nó final determina o fim da atividade
- ➡ O nó final termina todos os caminhos paralelos que estão sendo seguidos
- ➡ Indica por um círculo negro dentro de um círculo vazado

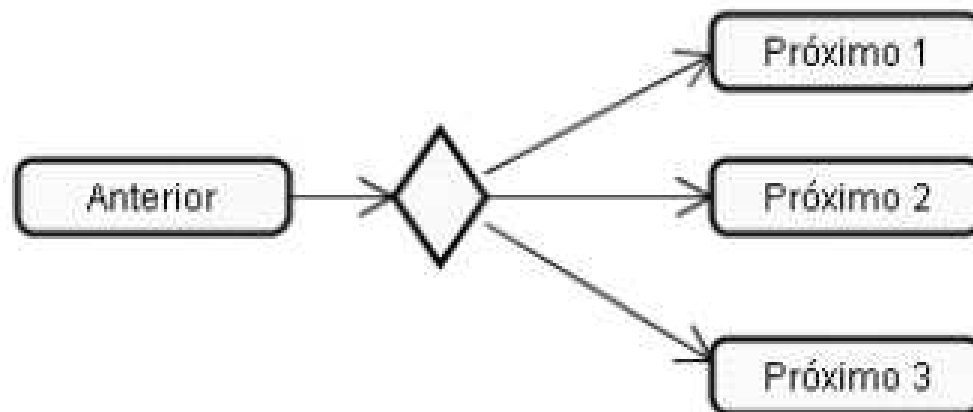
# Nó de fim de fluxo



- ➡ O nó de fim de caminho determina que o caminho sendo seguido se esgotou
  - ▮ **Não determina o fim da atividade** como um todo se outros caminhos estiverem sendo seguidos

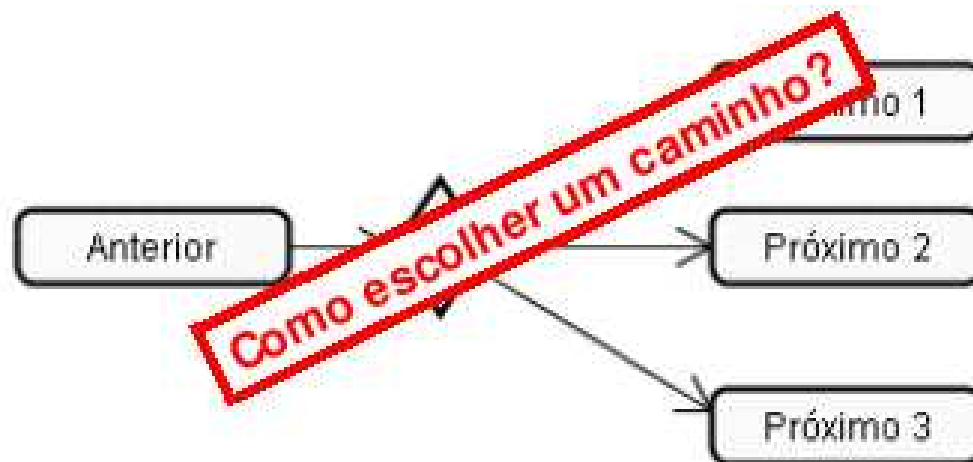
# Nó de Decisão

- ➡ Permite escolher um entre vários caminhos possíveis, de acordo com uma regra de escolha
- ➡ Semelhante a um "CASE"



# Nó de Decisão

- ➡ Permite escolher um entre vários caminhos possíveis, de acordo com uma regra de escolha
- ➡ Semelhante a um "CASE"

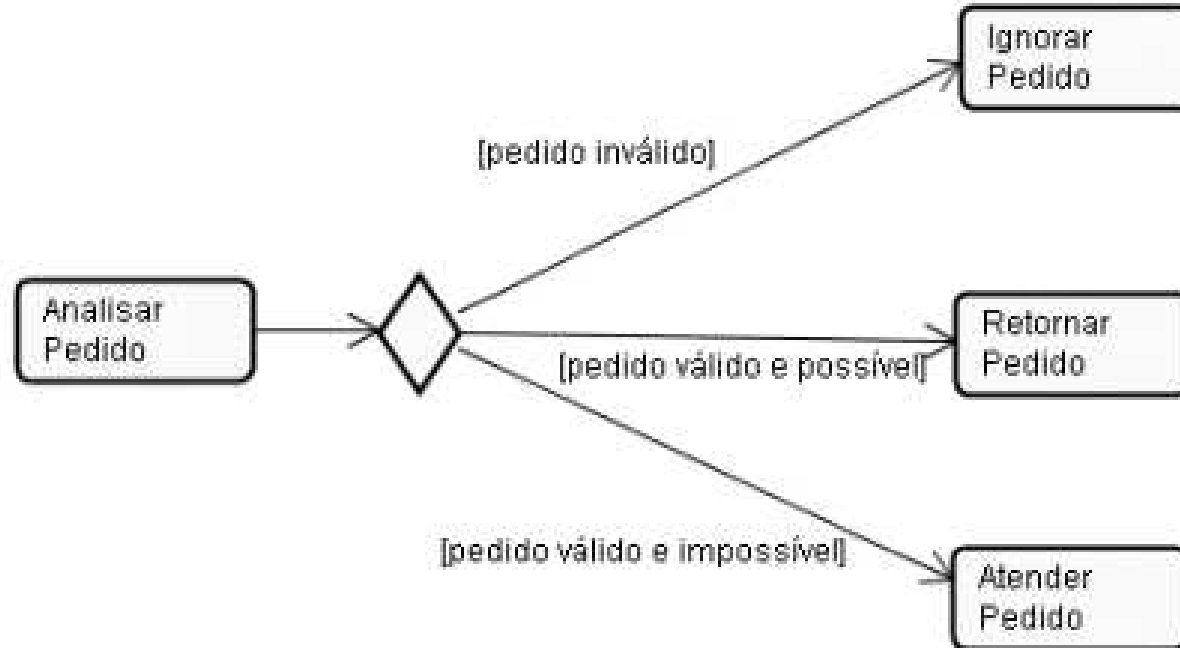


# "Guardas"

- ➡ São expressões lógicas que ajudam a decidir que caminho tomar em um nó de decisão
  - ▬ "guardam o caminho"
- ➡ Apenas um caminho pode ser verdade
- ➡ Aplicam-se aos caminhos (arestas)
- ➡ Estão dentro de colchetes

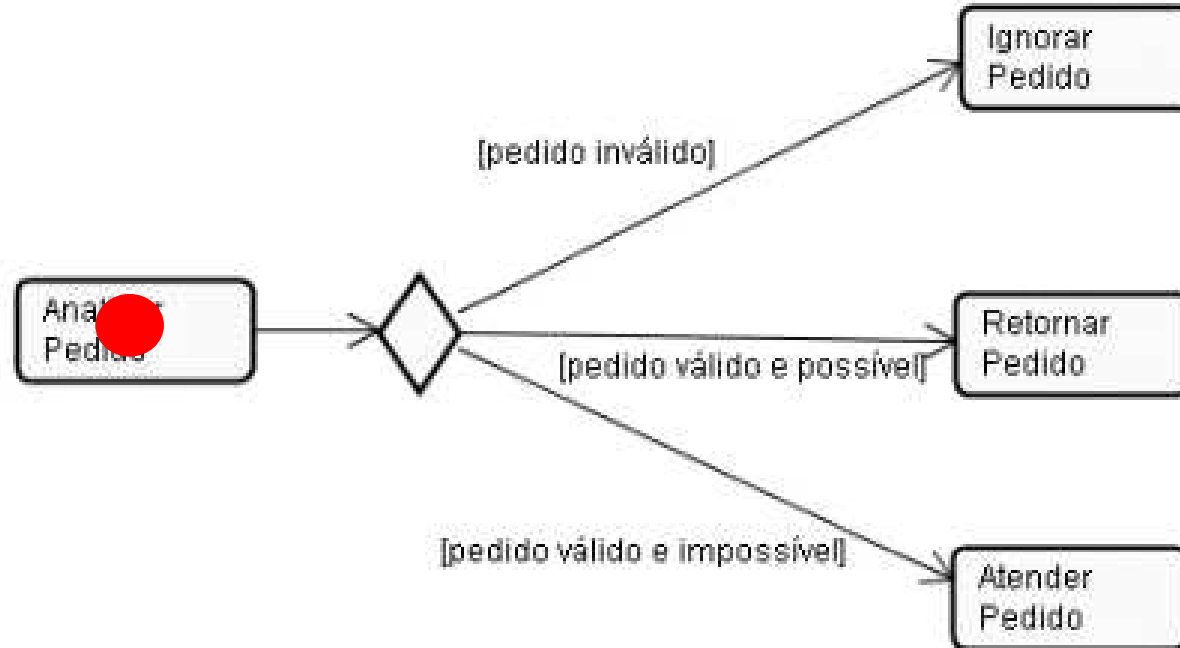
# Decisão e Guardas

➡ Atenção para o formato específico



# Decisão e Guardas

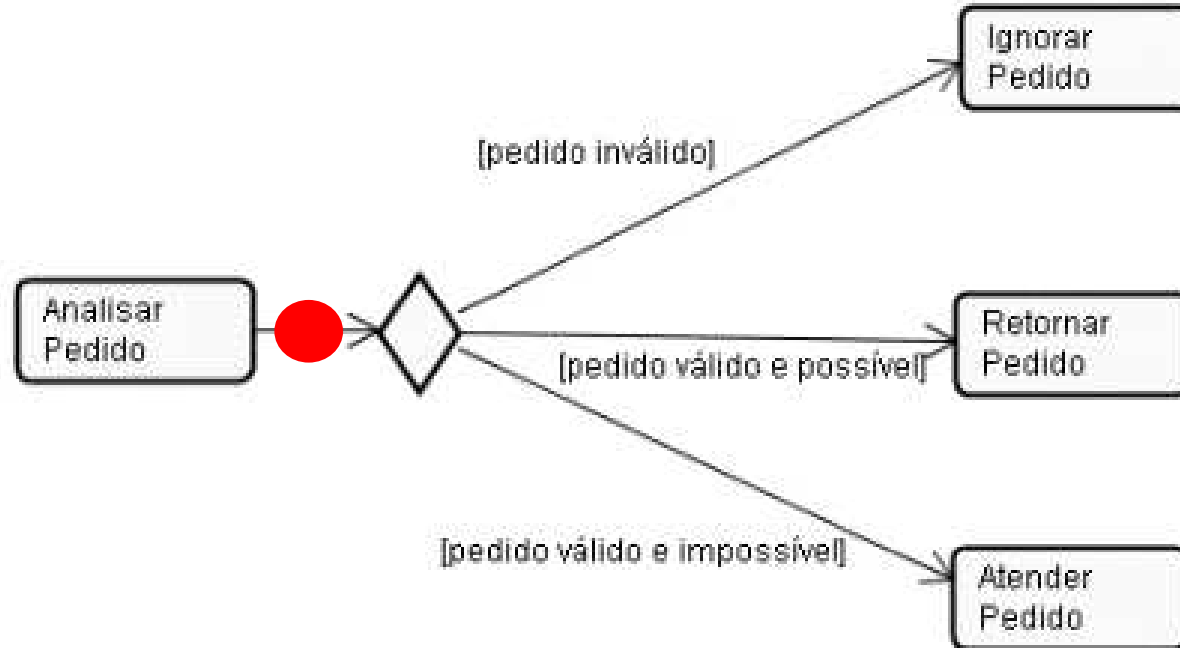
➡ Atenção para o formato específico





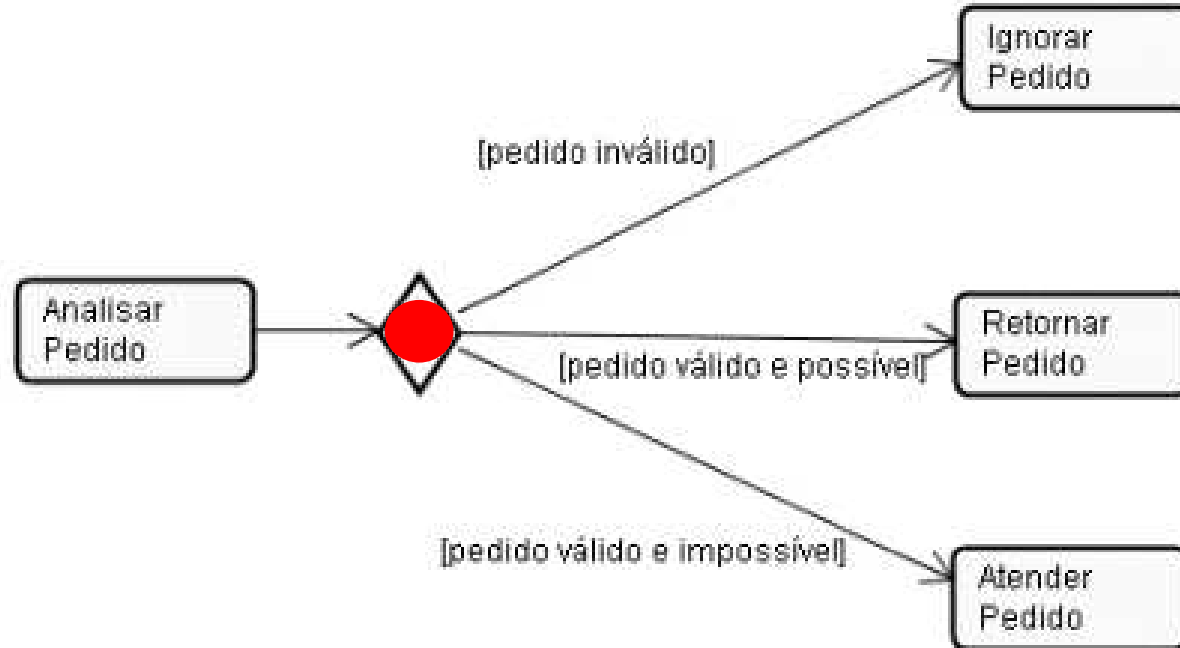
# Decisão e Guardas

➡ Atenção para o formato específico



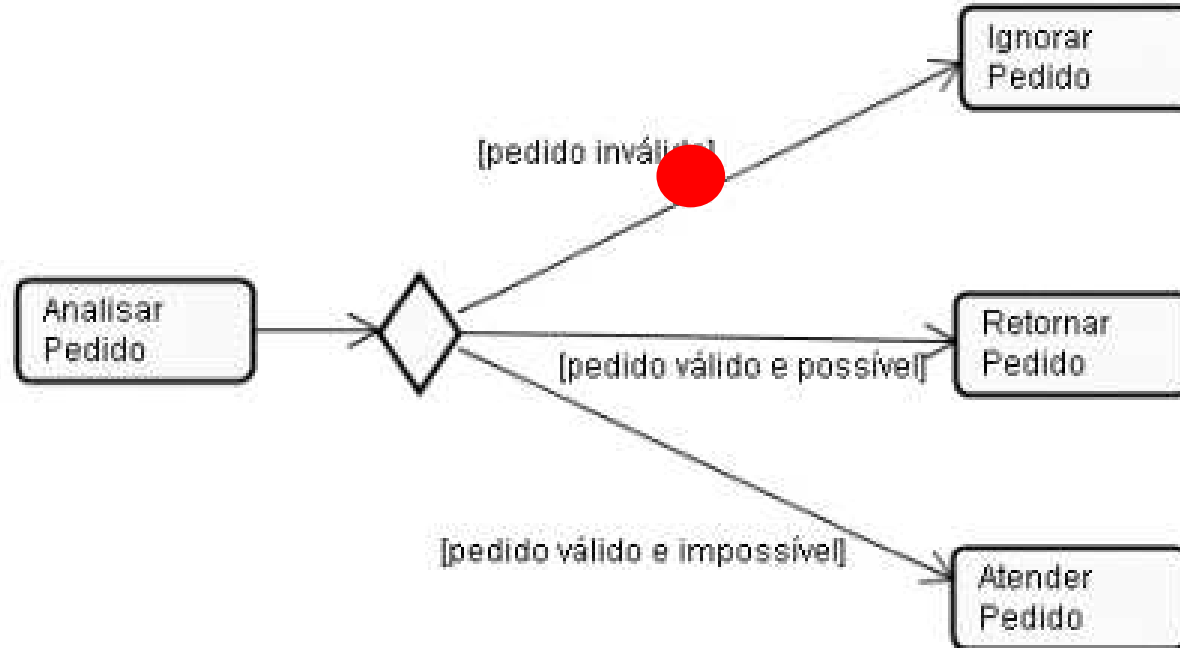
# Decisão e Guardas

➡ Atenção para o formato específico



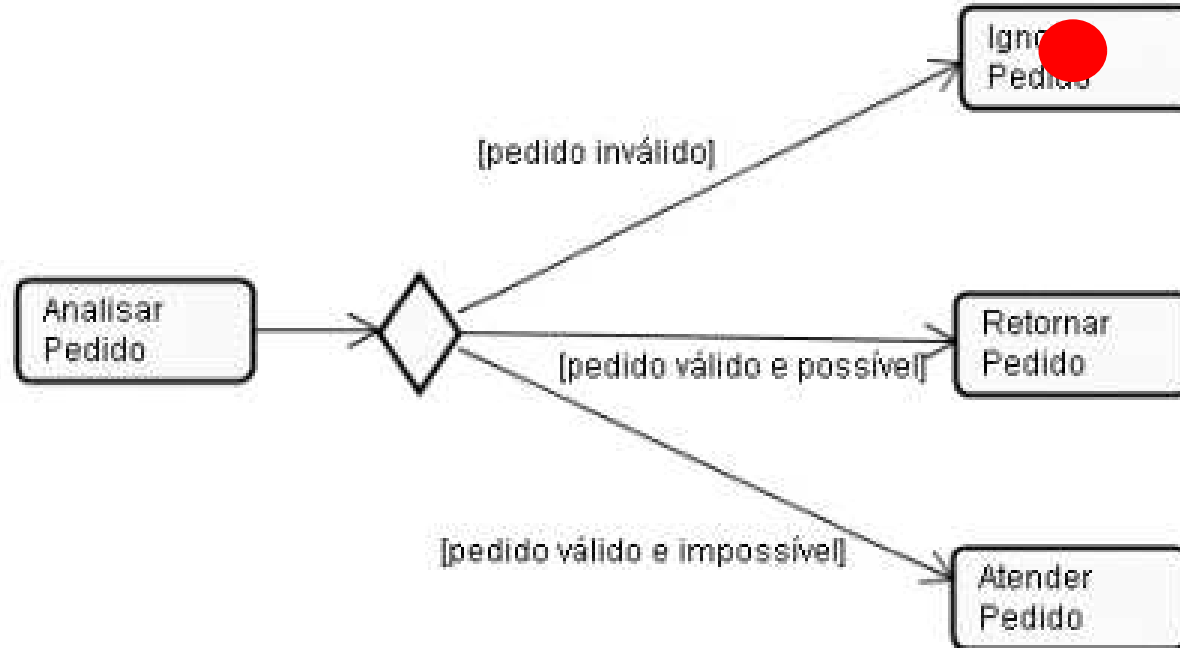
# Decisão e Guardas

➡ Atenção para o formato específico



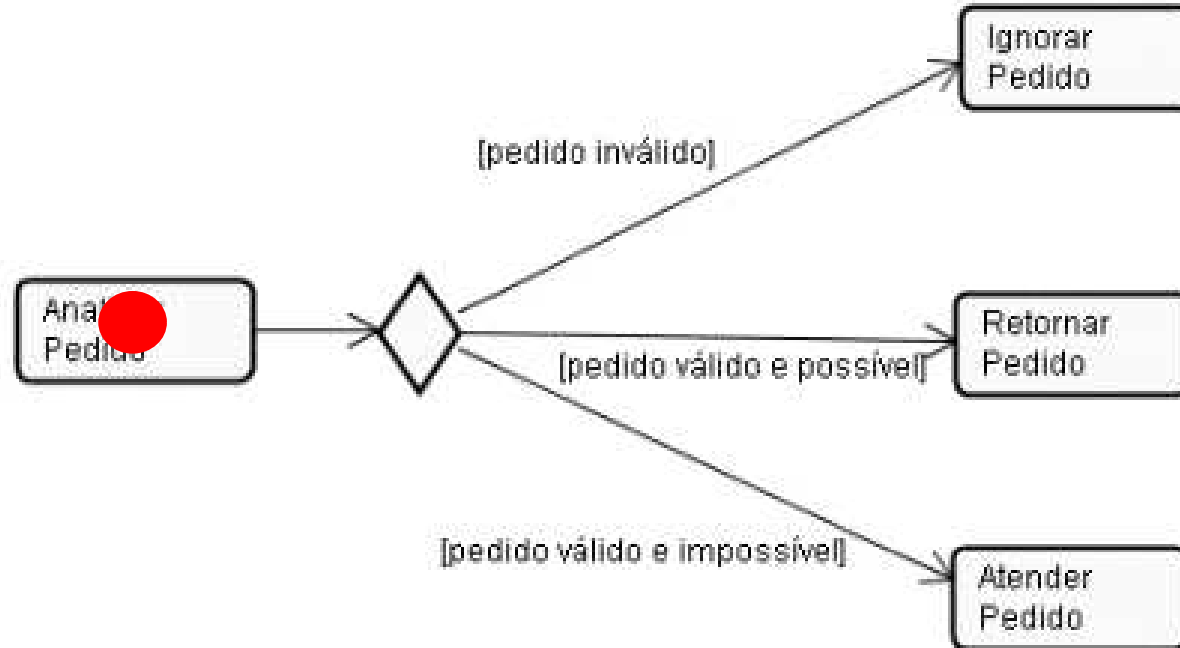
# Decisão e Guardas

➡ Atenção para o formato específico



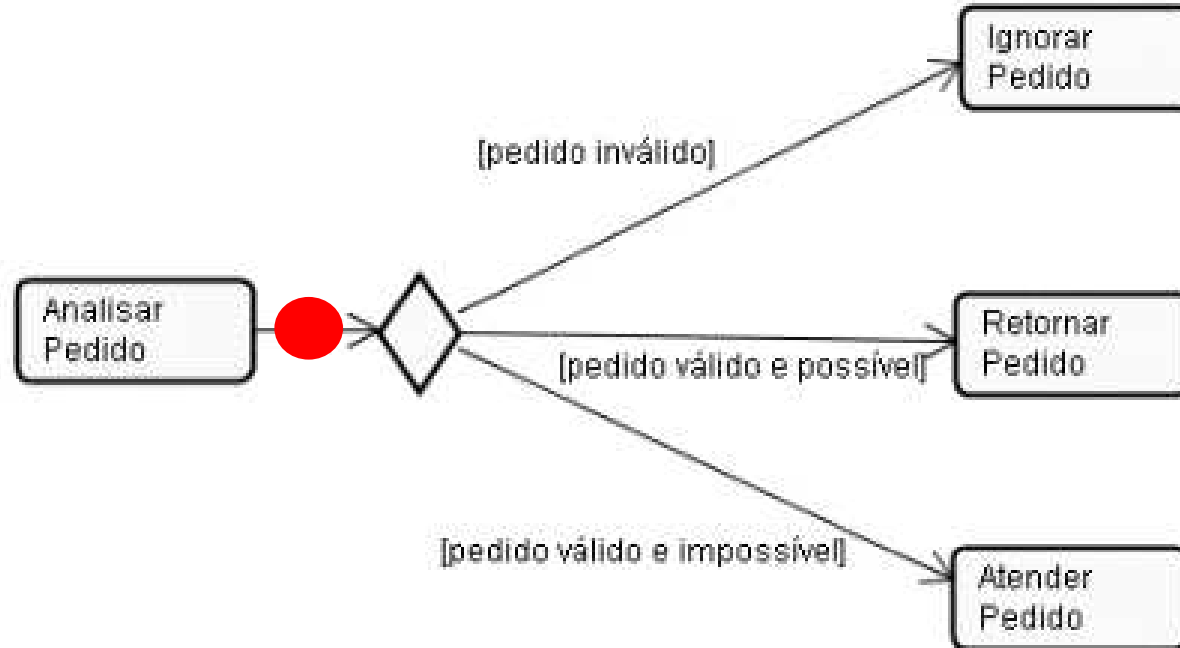
# Decisão e Guardas

➡ Atenção para o formato específico



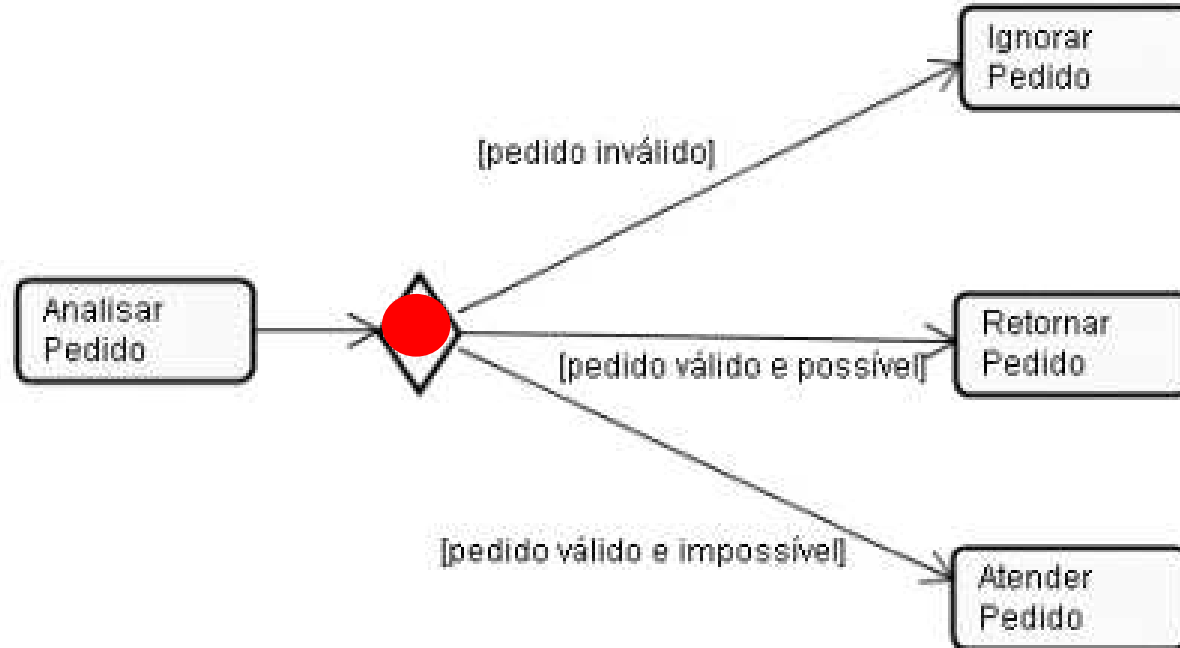
# Decisão e Guardas

➡ Atenção para o formato específico



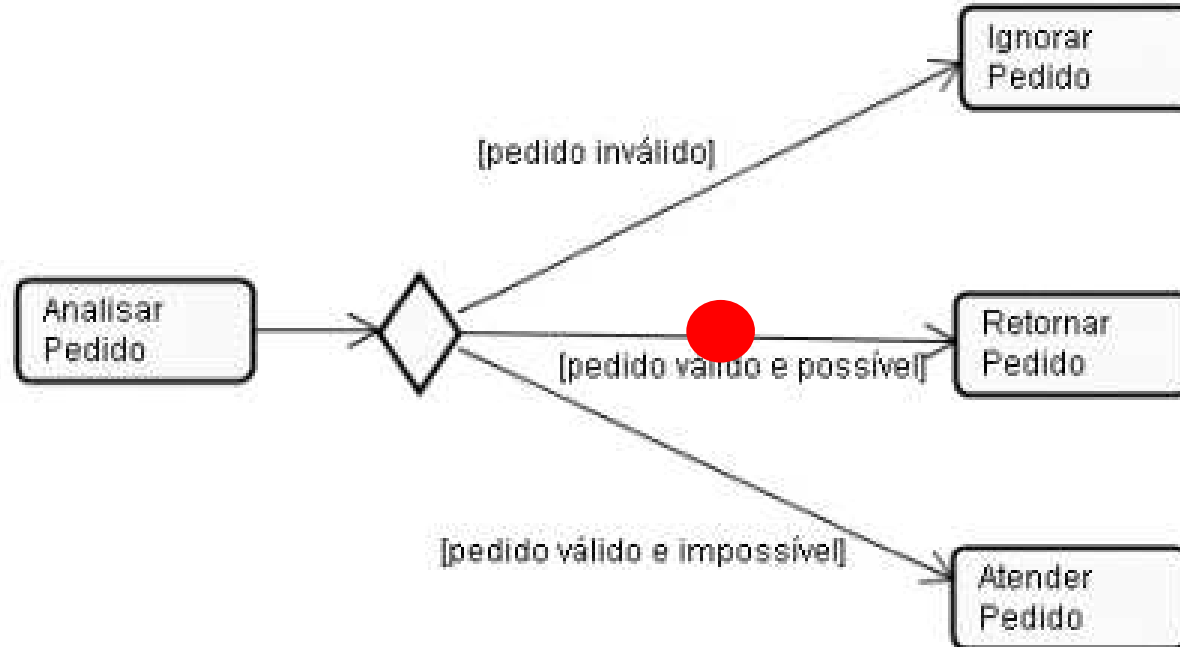
# Decisão e Guardas

➡ Atenção para o formato específico



# Decisão e Guardas

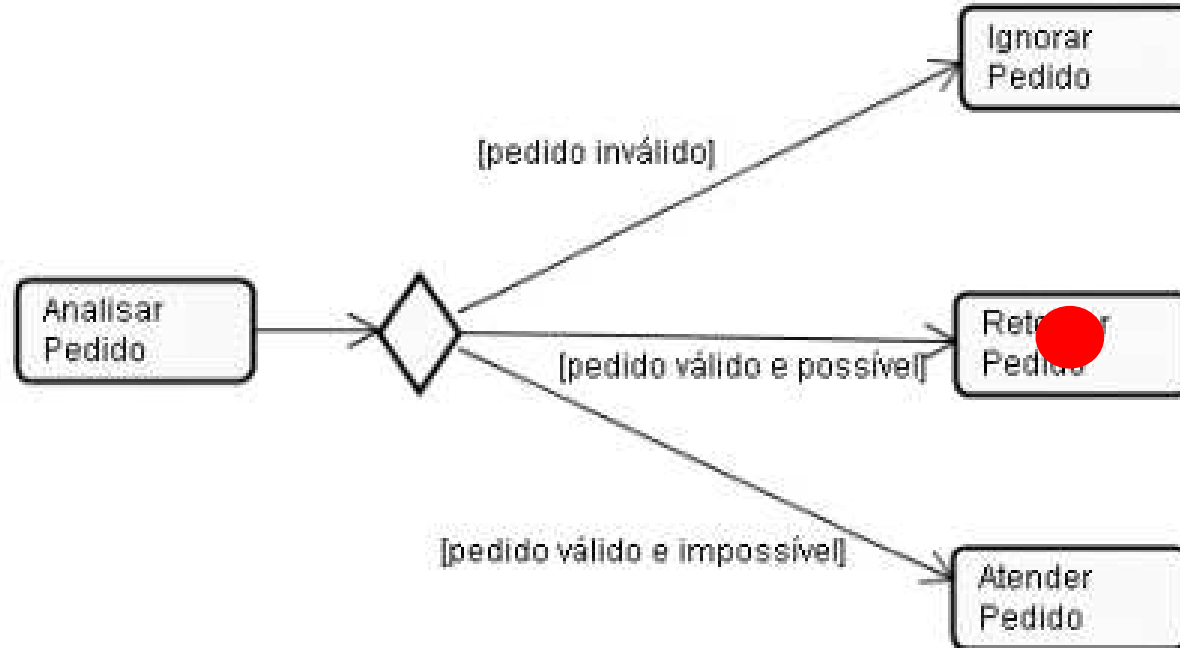
➡ Atenção para o formato específico





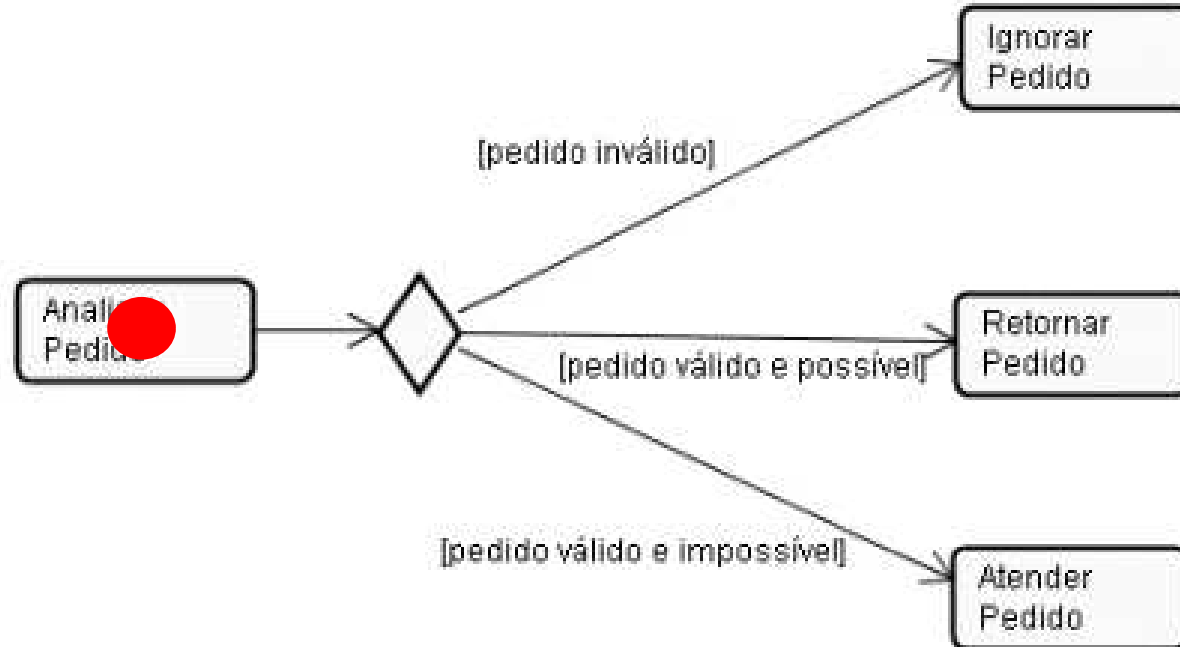
# Decisão e Guardas

➡ Atenção para o formato específico



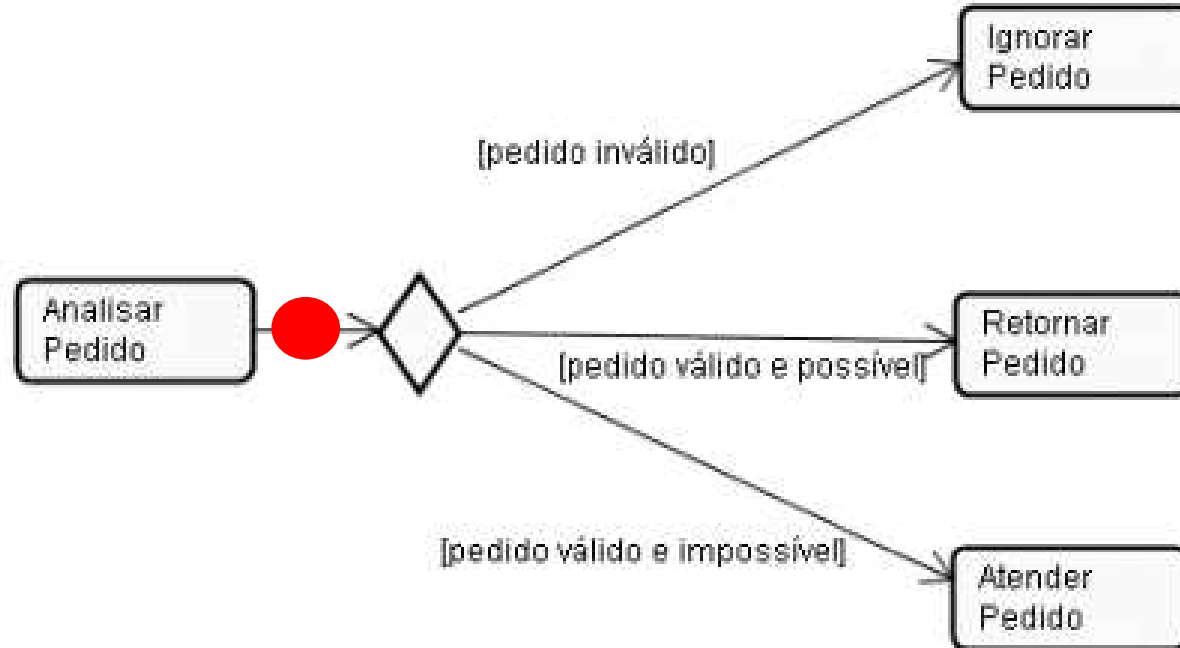
# Decisão e Guardas

➡ Atenção para o formato específico



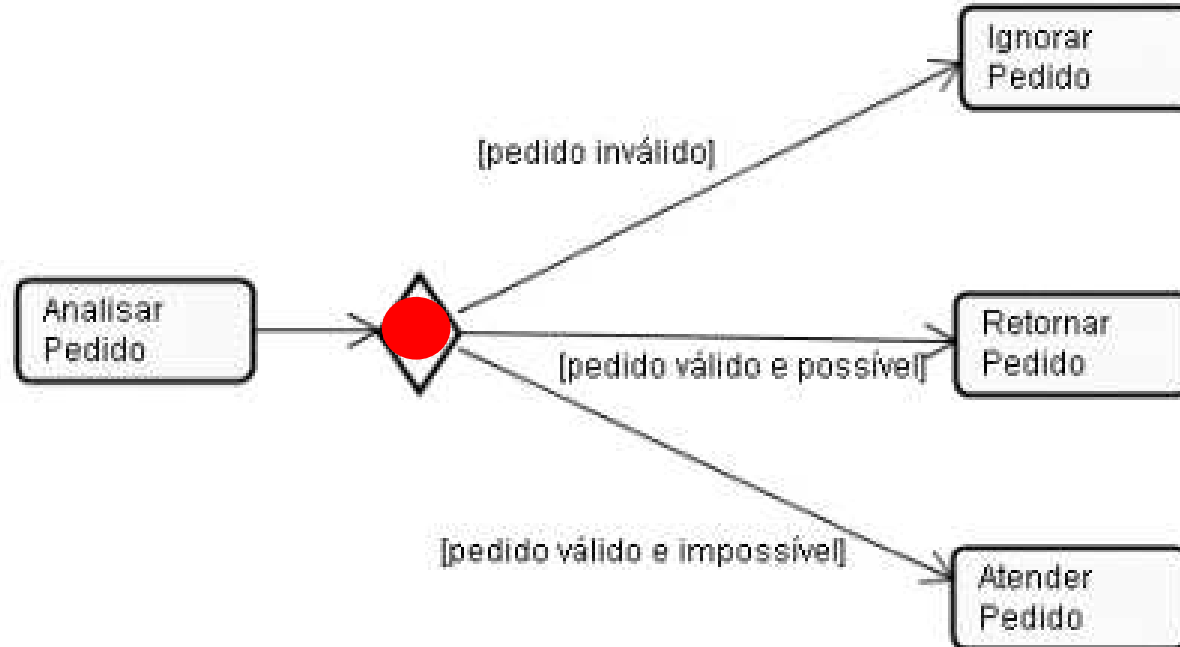
# Decisão e Guardas

➡ Atenção para o formato específico



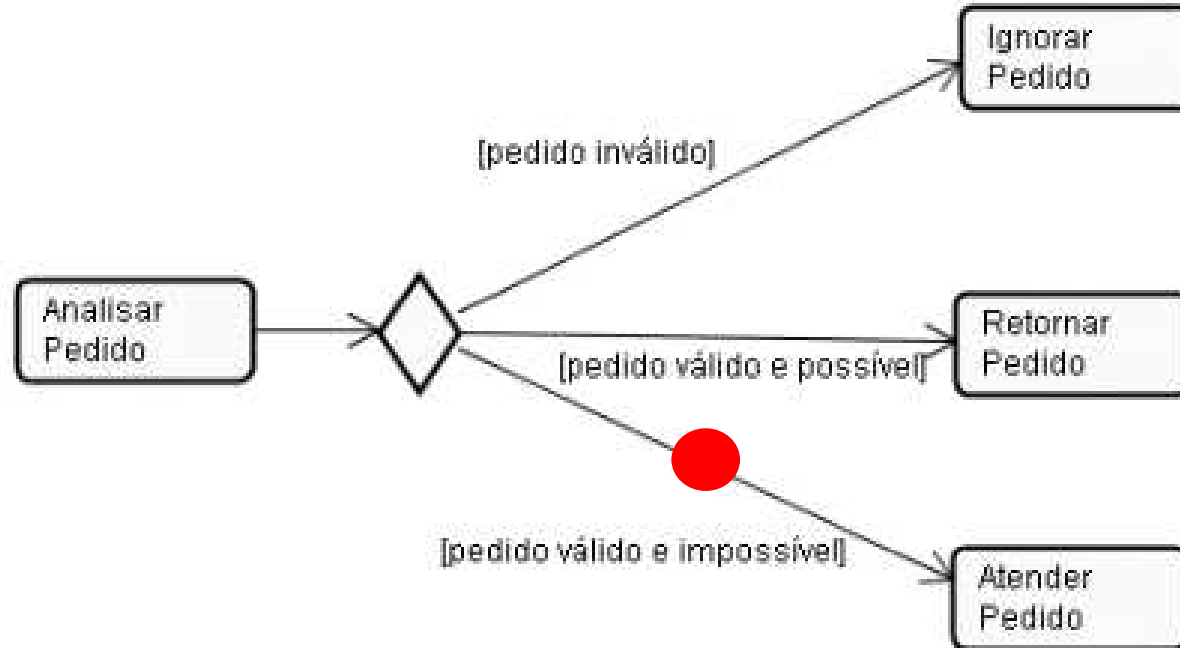
# Decisão e Guardas

➡ Atenção para o formato específico



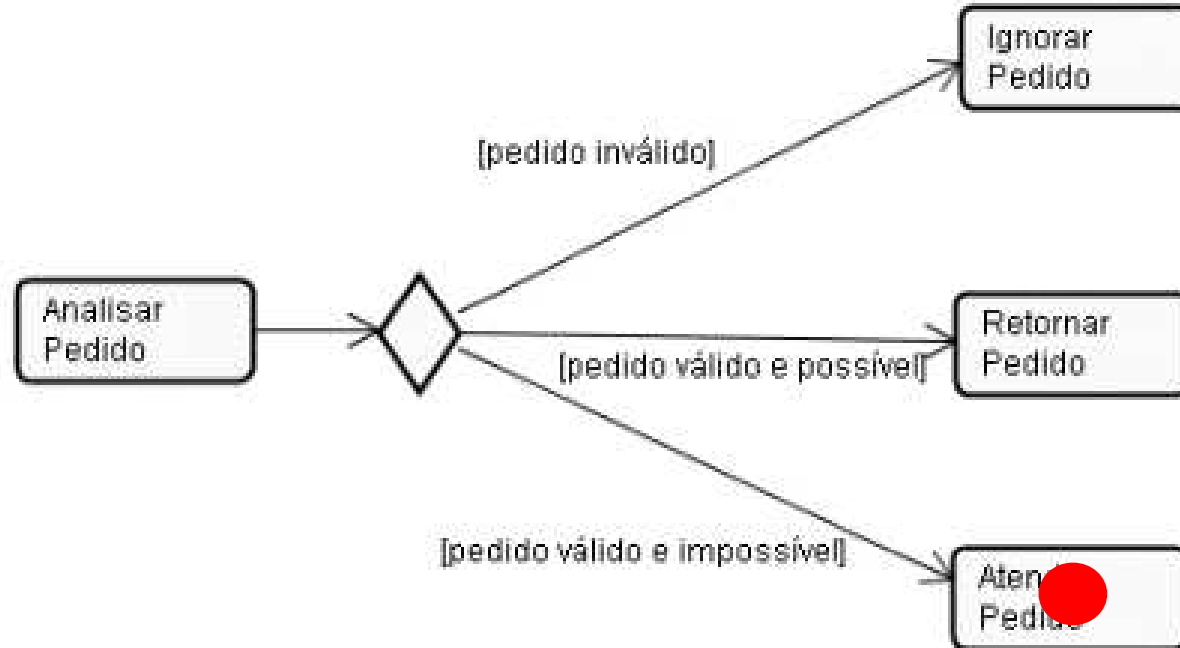
# Decisão e Guardas

➡ Atenção para o formato específico



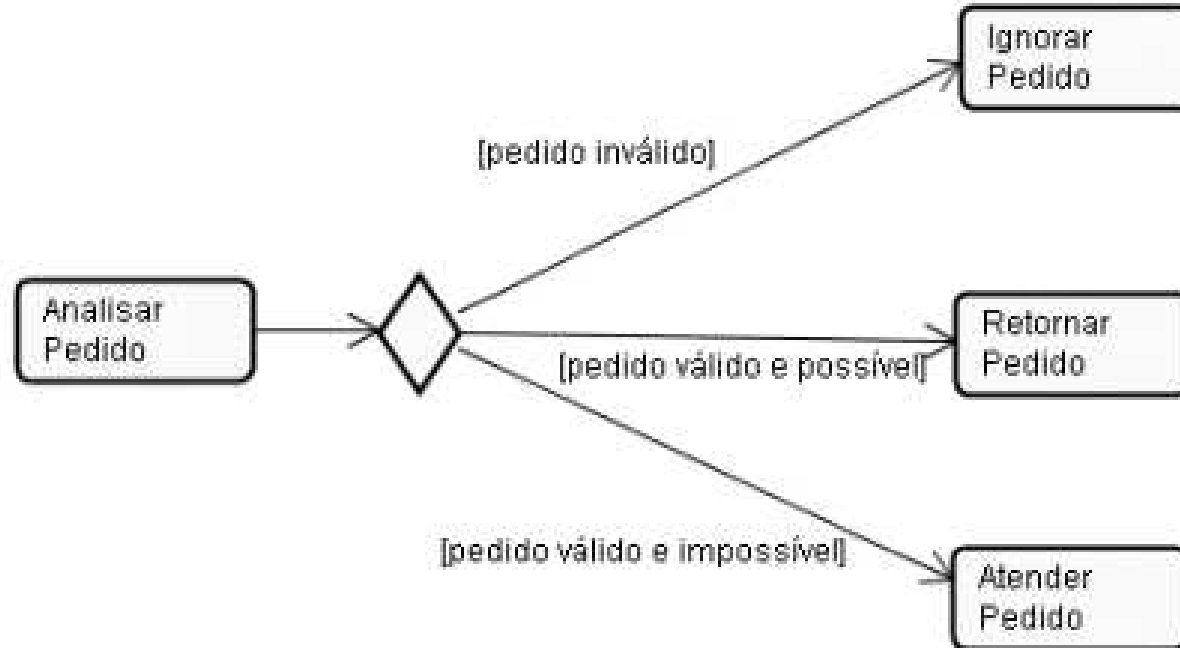
# Decisão e Guardas

➡ Atenção para o formato específico

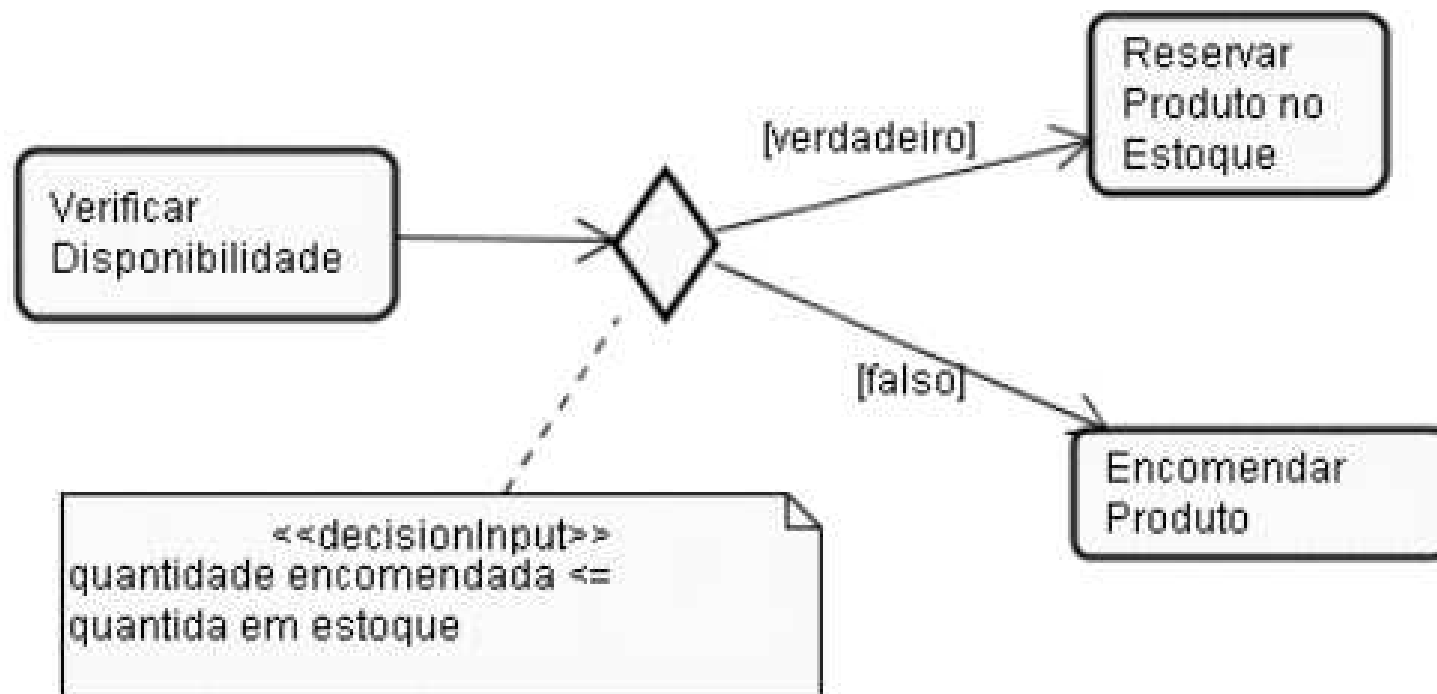


# Decisão e Guardas

➡ Atenção para o formato específico

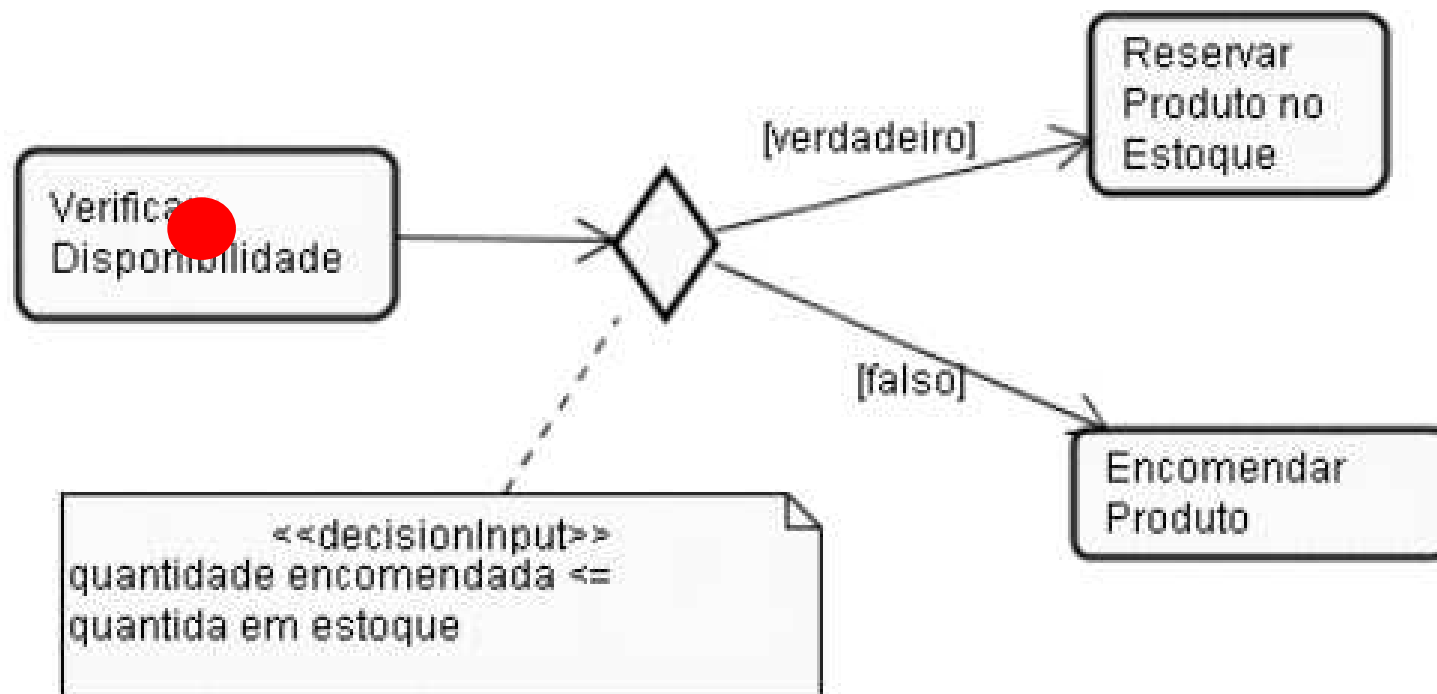


# Notação Alternativa de Decisão

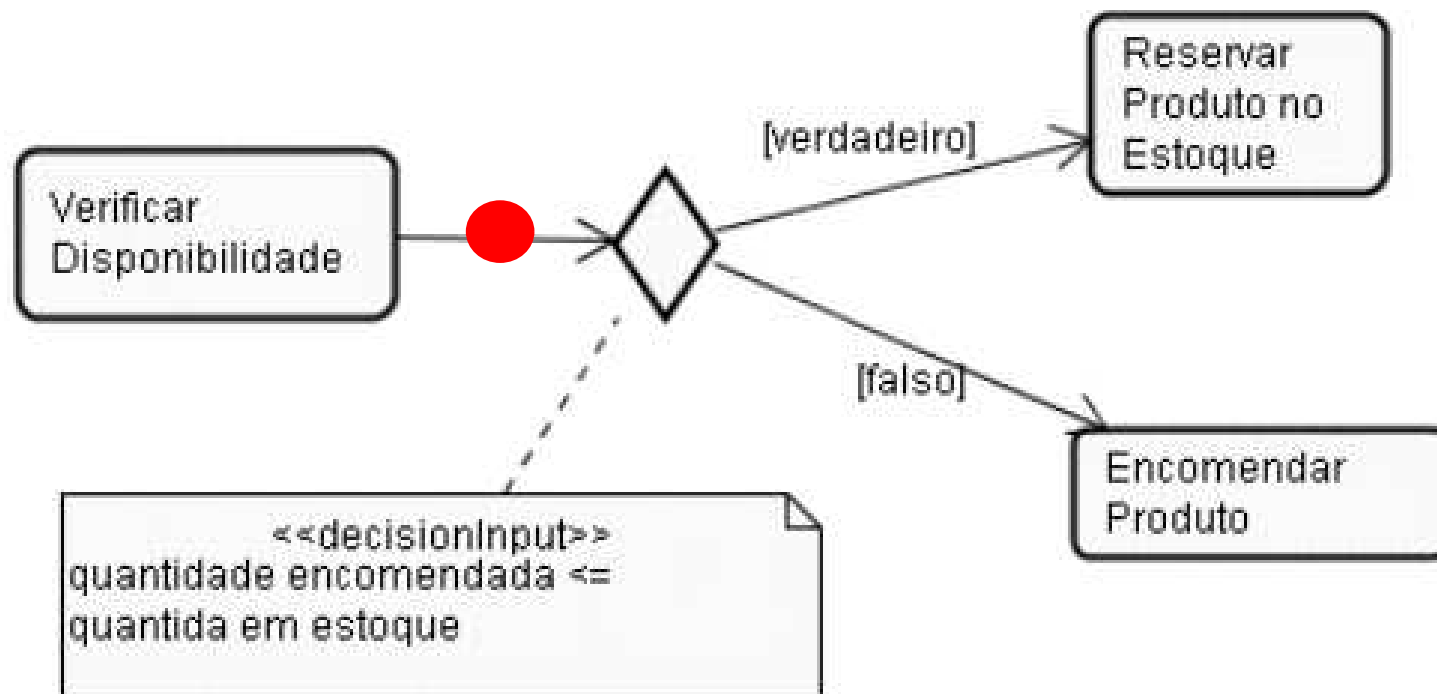




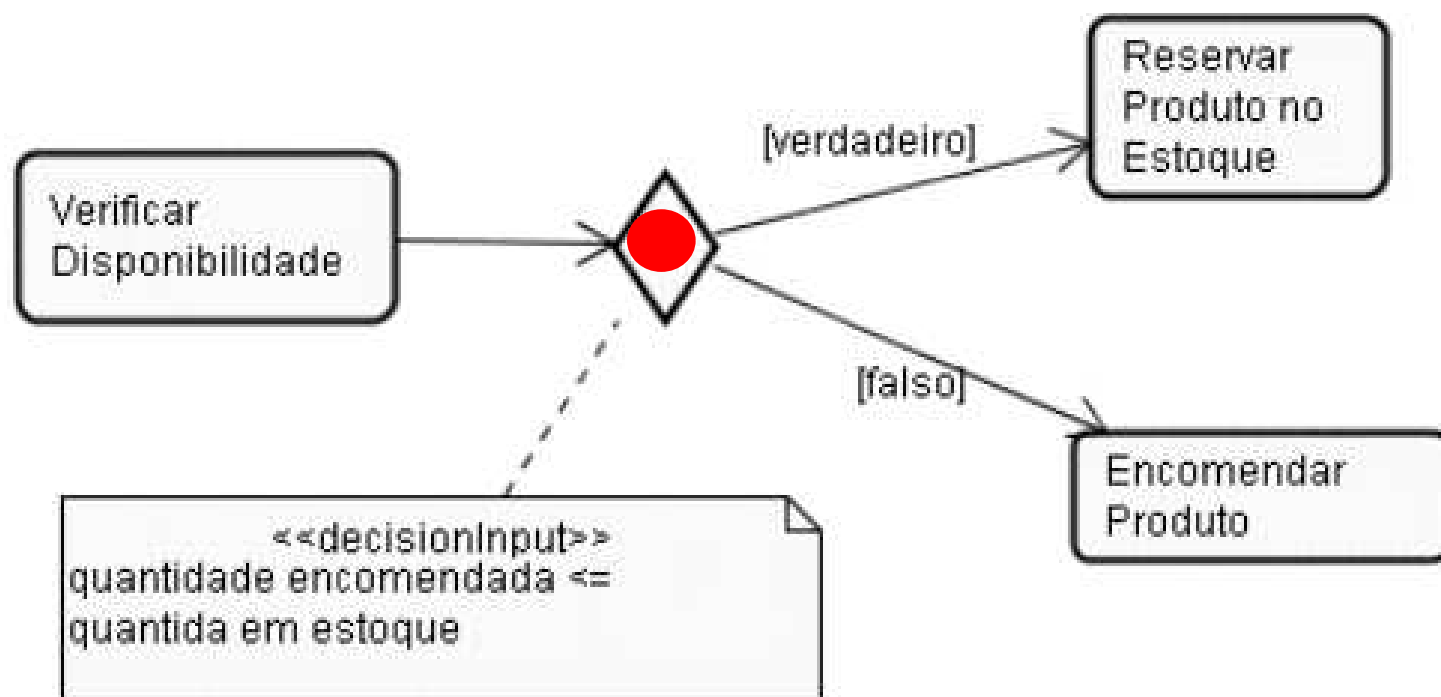
# Notação Alternativa de Decisão



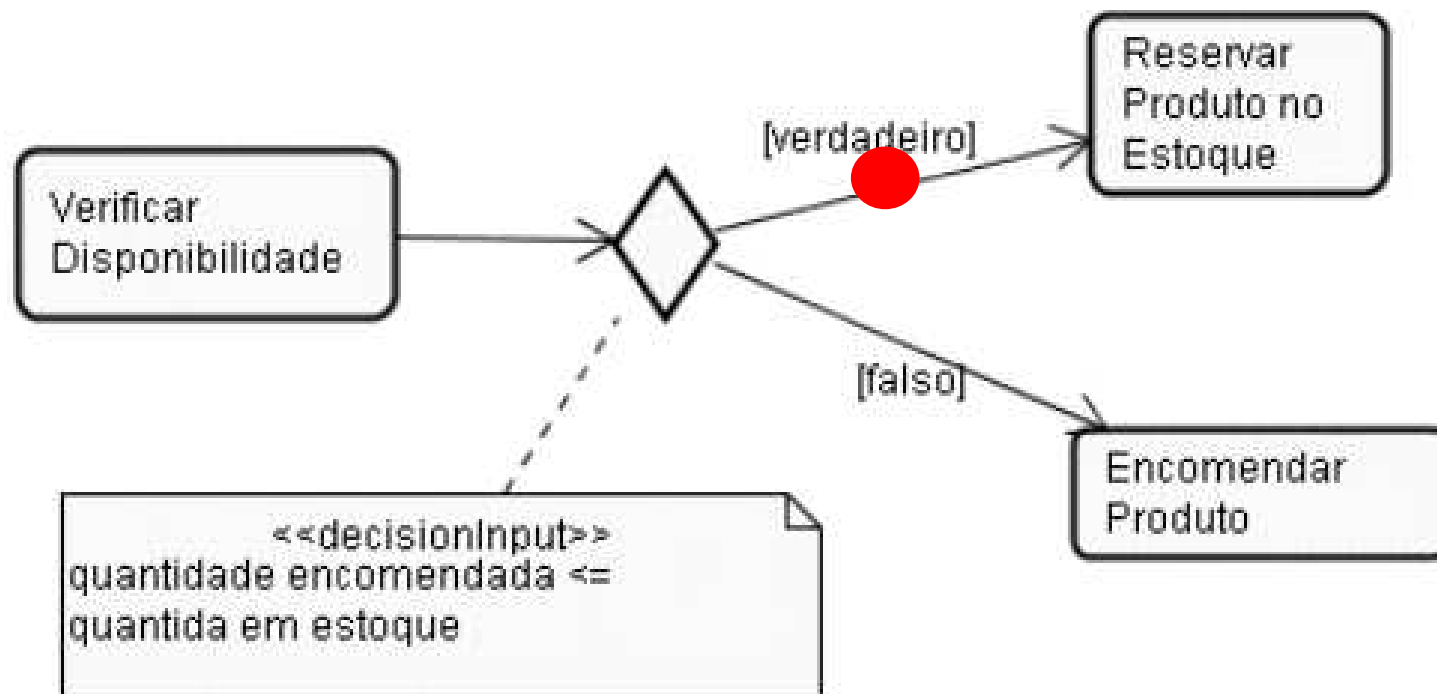
# Notação Alternativa de Decisão



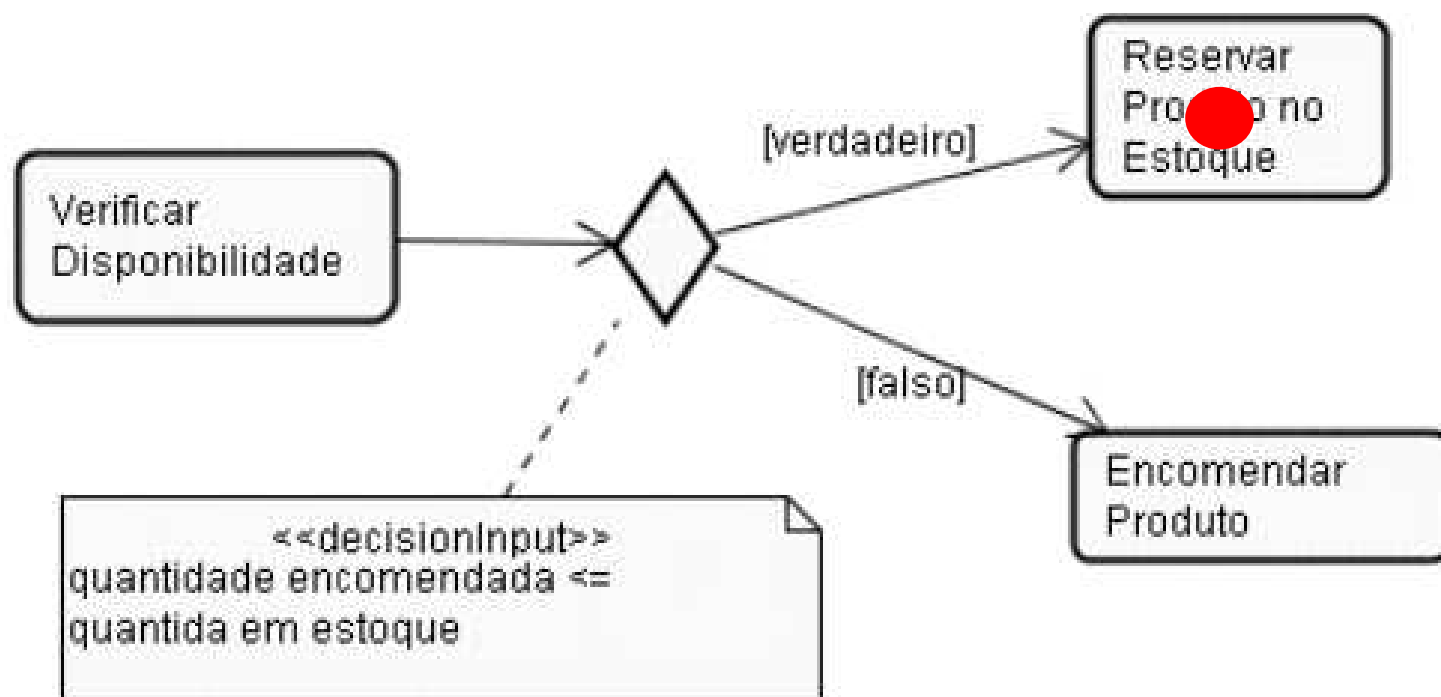
# Notação Alternativa de Decisão



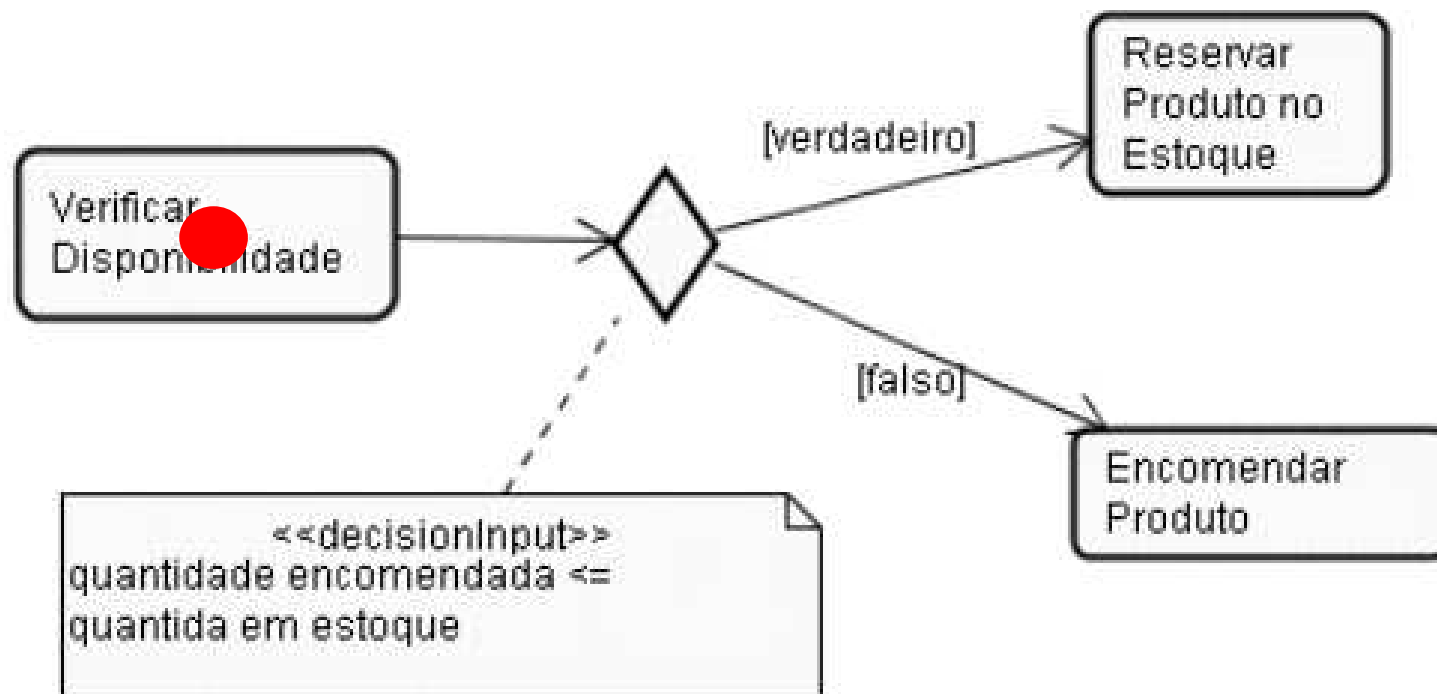
# Notação Alternativa de Decisão



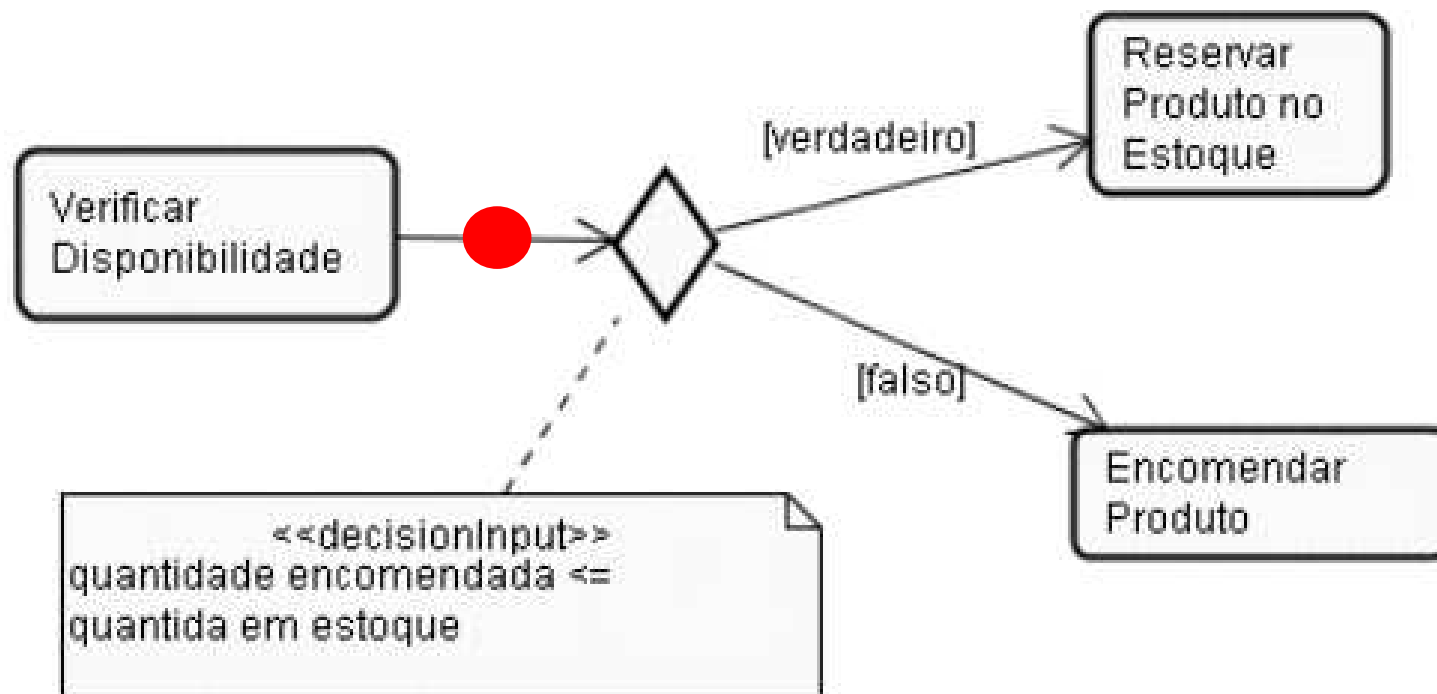
# Notação Alternativa de Decisão



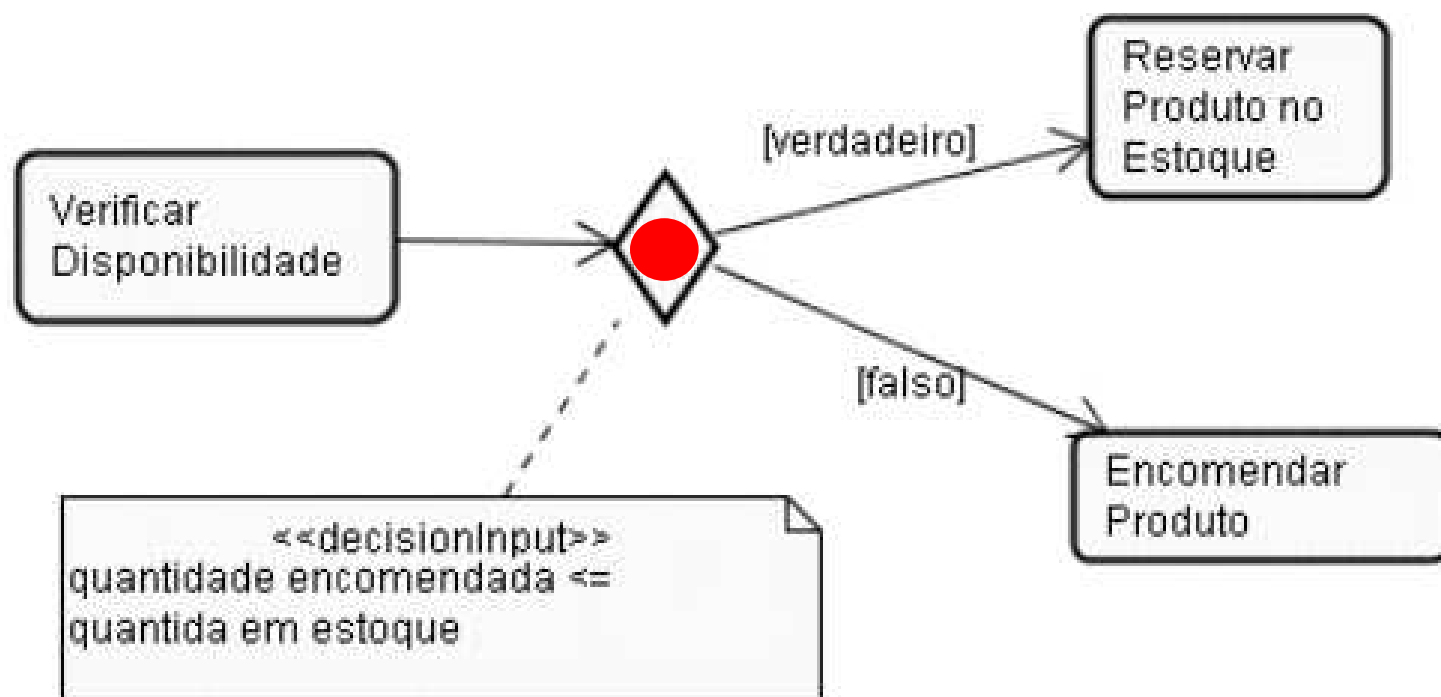
# Notação Alternativa de Decisão



# Notação Alternativa de Decisão

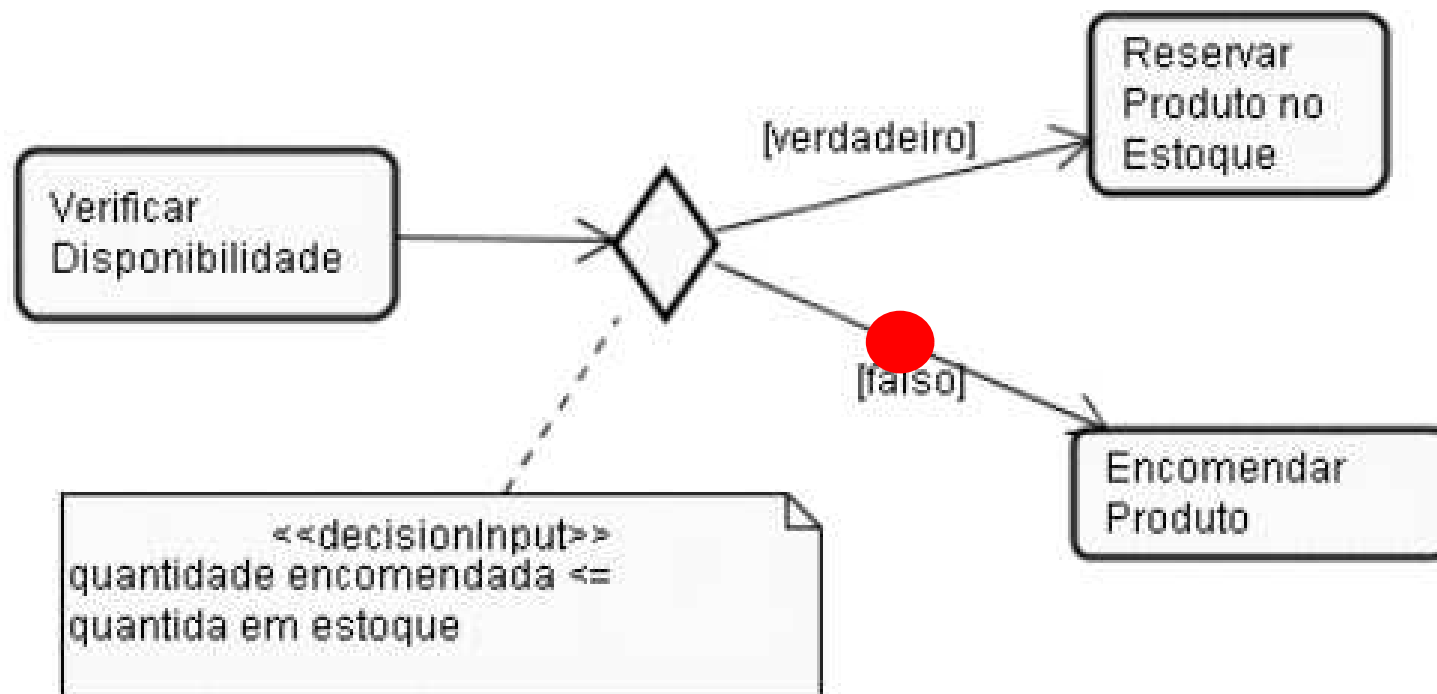


# Notação Alternativa de Decisão

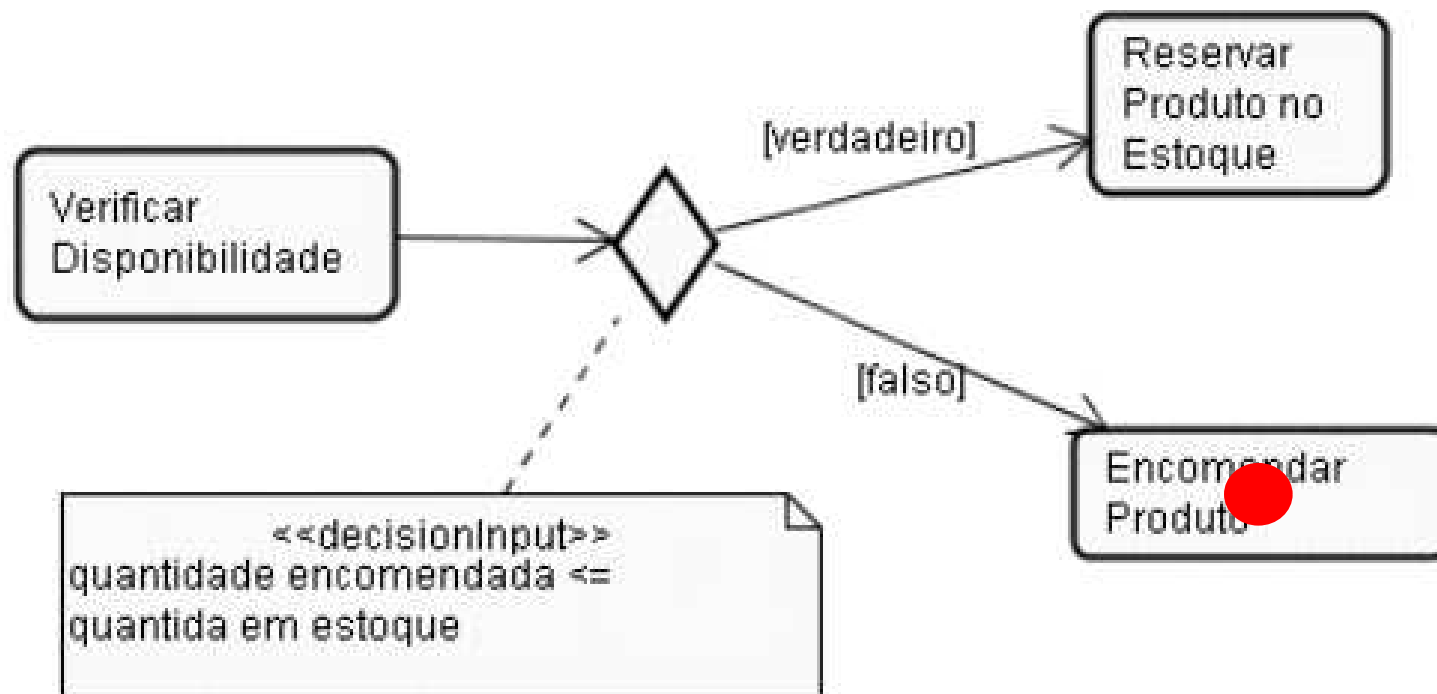




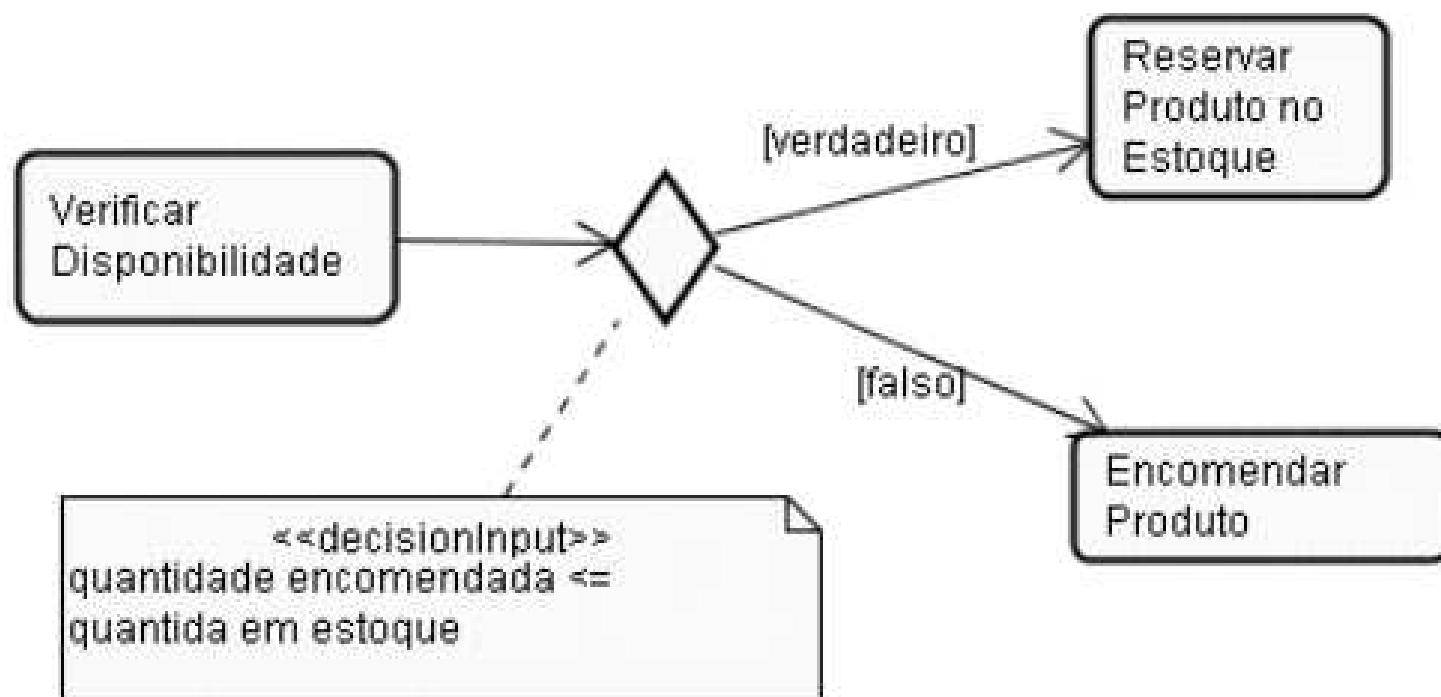
# Notação Alternativa de Decisão



# Notação Alternativa de Decisão



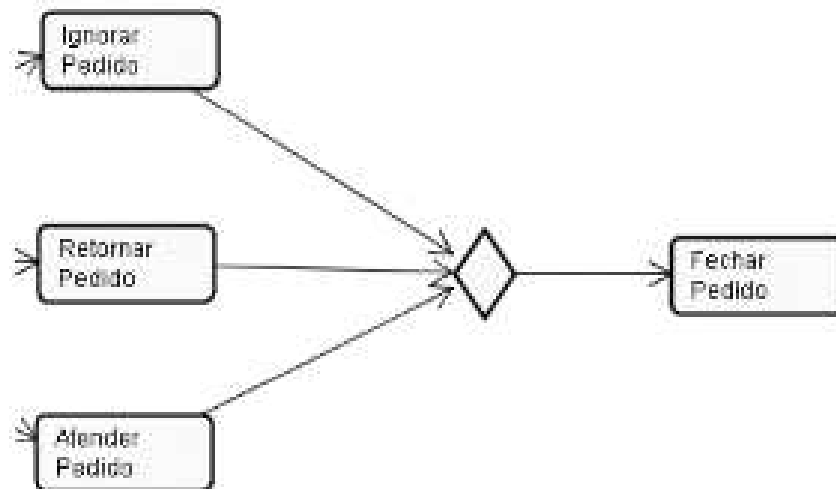
# Notação Alternativa de Decisão



# Nó de "merge" (unificação)

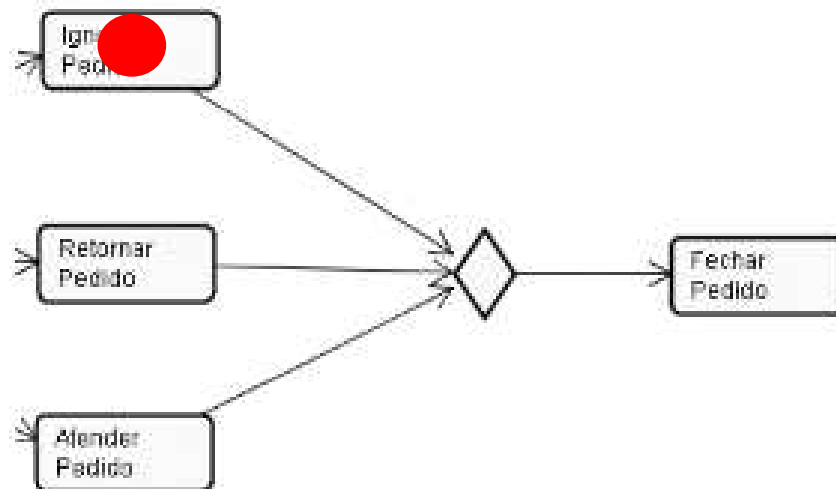
➡ Serve para juntar caminhos separados com o nó de decisão

➡ Usa o mesmo símbolo da decisão



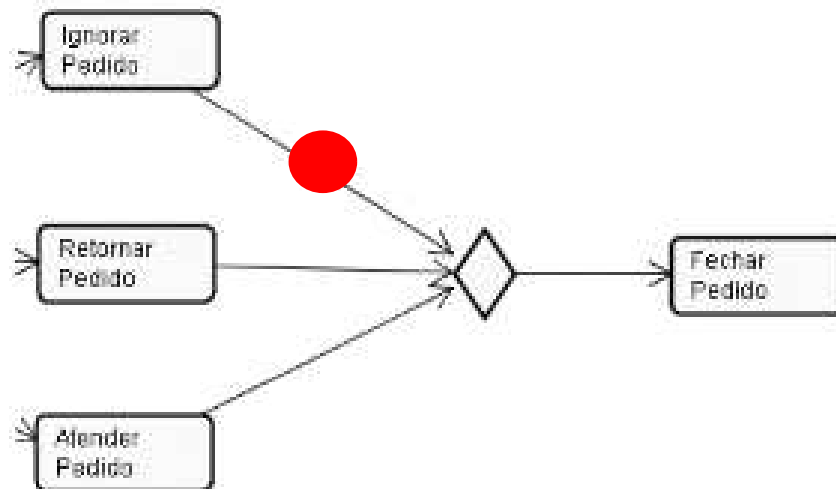
# Nó de "merge" (unificação)

- ➡ Serve para juntar caminhos separados com o nó de decisão
- ➡ Usa o mesmo símbolo da decisão



# Nó de "merge" (unificação)

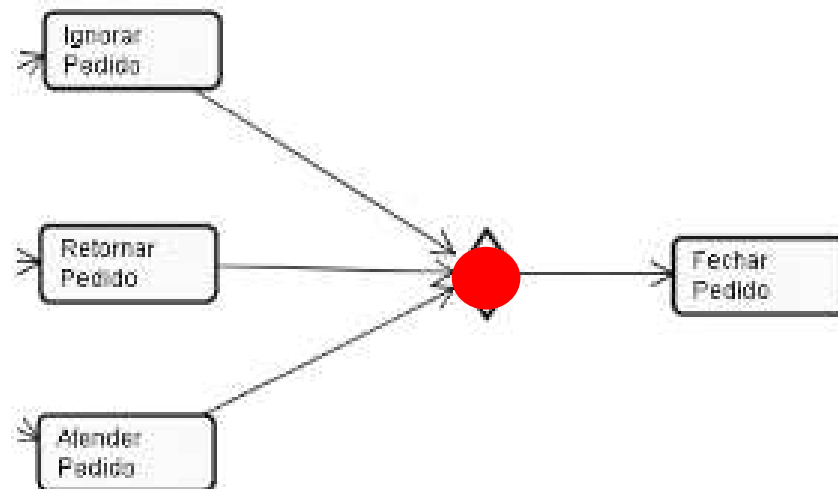
- ➡ Serve para juntar caminhos separados com o nó de decisão
- ➡ Usa o mesmo símbolo da decisão



# Nó de "merge" (unificação)

➡ Serve para juntar caminhos separados com o nó de decisão

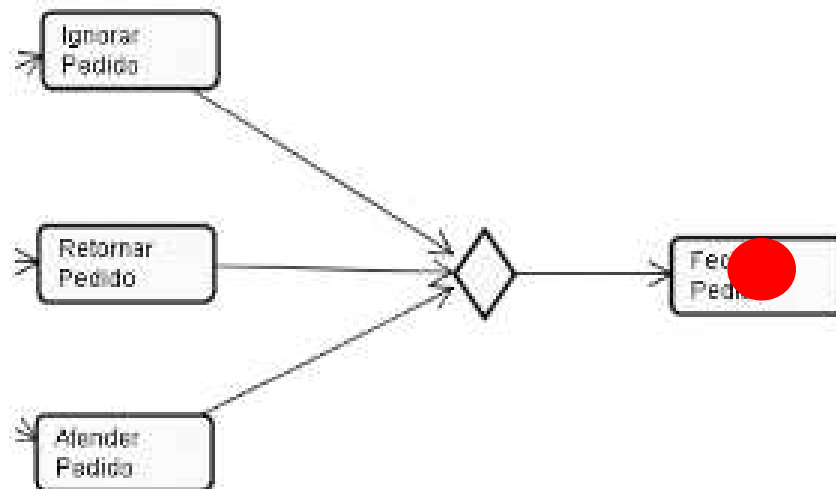
➡ Usa o mesmo símbolo da decisão



# Nó de "merge" (unificação)

➡ Serve para juntar caminhos separados com o nó de decisão

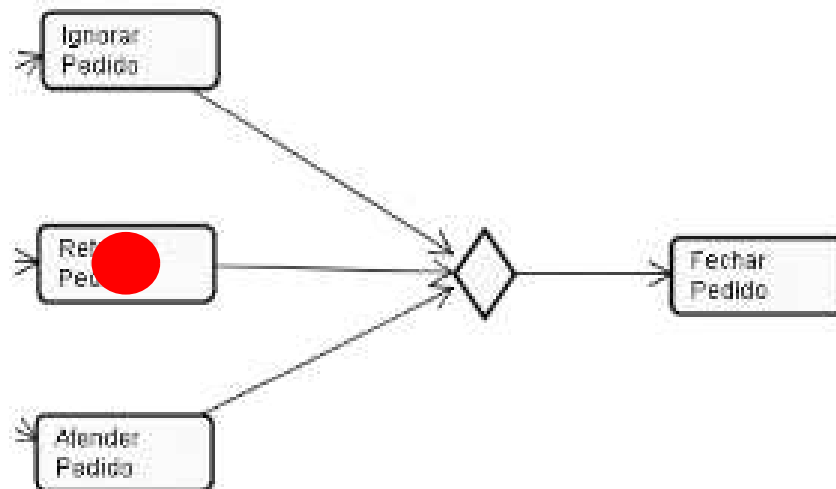
➡ Usa o mesmo símbolo da decisão





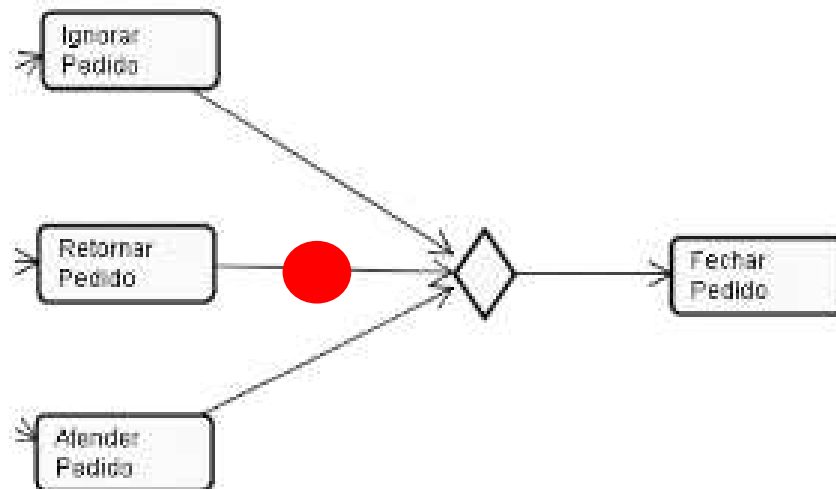
# Nó de "merge" (unificação)

- ➡ Serve para juntar caminhos separados com o nó de decisão
- ➡ Usa o mesmo símbolo da decisão



# Nó de "merge" (unificação)

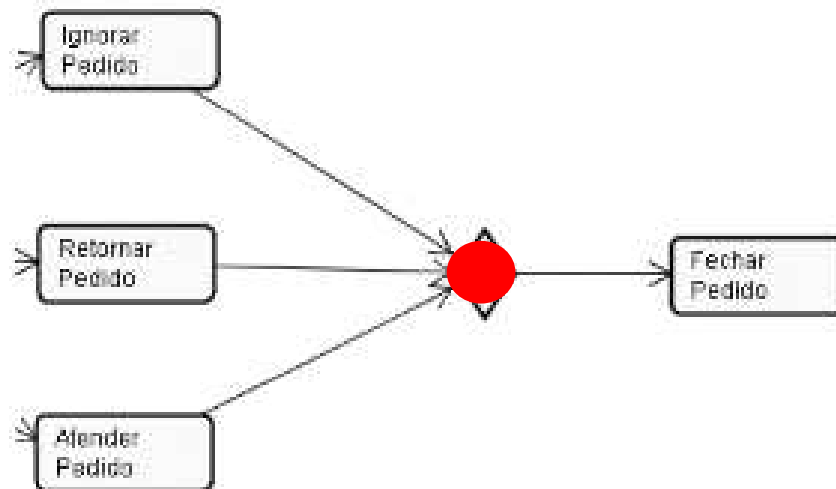
- ➡ Serve para juntar caminhos separados com o nó de decisão
- ➡ Usa o mesmo símbolo da decisão



# Nó de "merge" (unificação)

➡ Serve para juntar caminhos separados com o nó de decisão

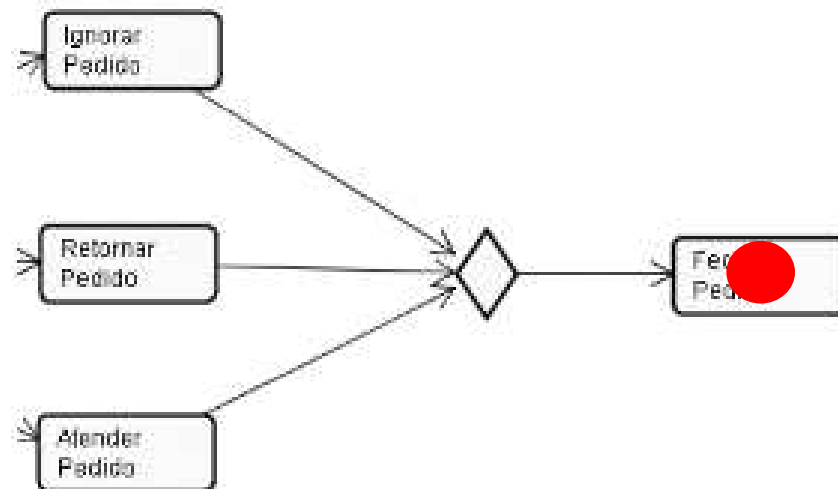
➡ Usa o mesmo símbolo da decisão



# Nó de "merge" (unificação)

➡ Serve para juntar caminhos separados com o nó de decisão

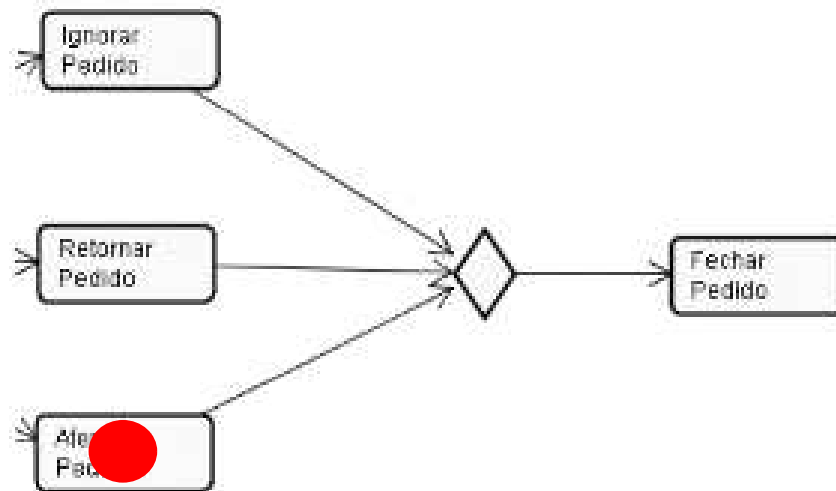
➡ Usa o mesmo símbolo da decisão



# Nó de "merge" (unificação)

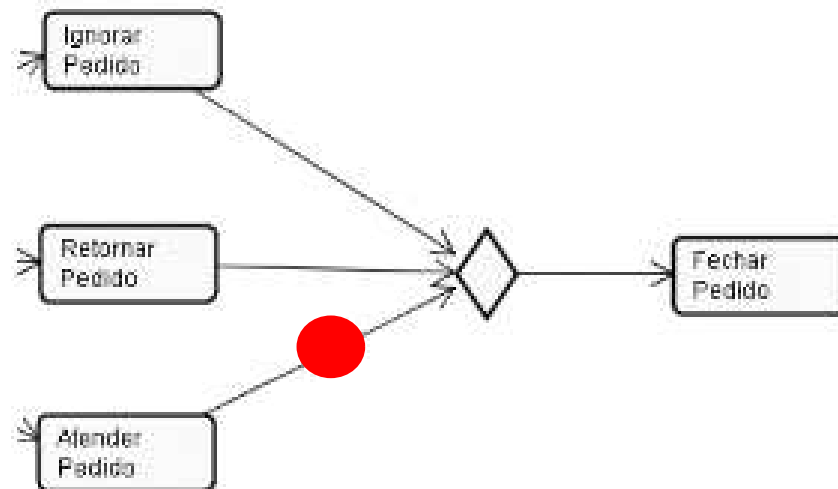
➡ Serve para juntar caminhos separados com o nó de decisão

➡ Usa o mesmo símbolo da decisão



# Nó de "merge" (unificação)

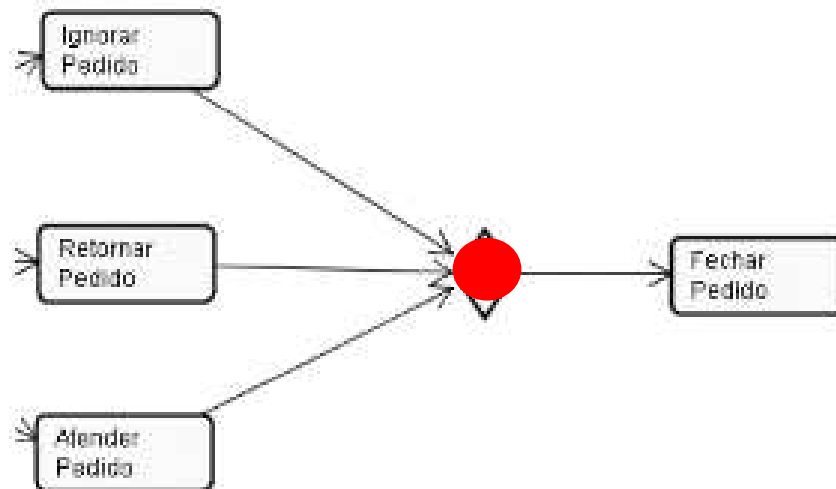
- ➡ Serve para juntar caminhos separados com o nó de decisão
- ➡ Usa o mesmo símbolo da decisão



# Nó de "merge" (unificação)

➡ Serve para juntar caminhos separados com o nó de decisão

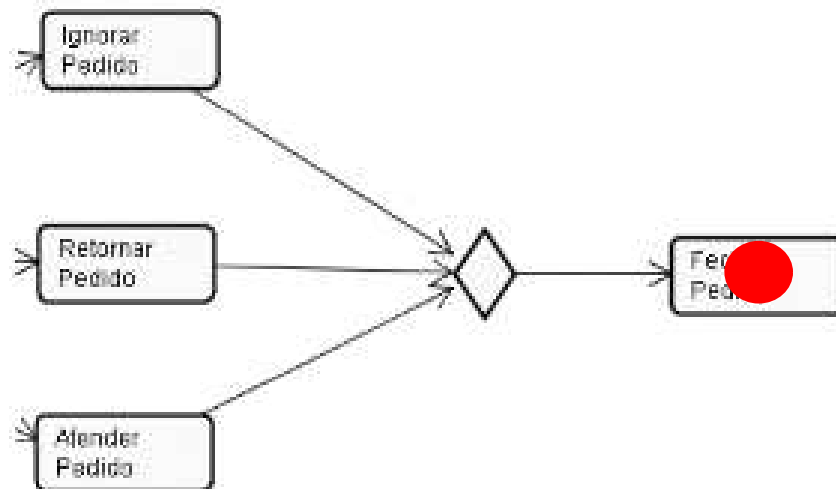
➡ Usa o mesmo símbolo da decisão



# Nó de "merge" (unificação)

➡ Serve para juntar caminhos separados com o nó de decisão

➡ Usa o mesmo símbolo da decisão

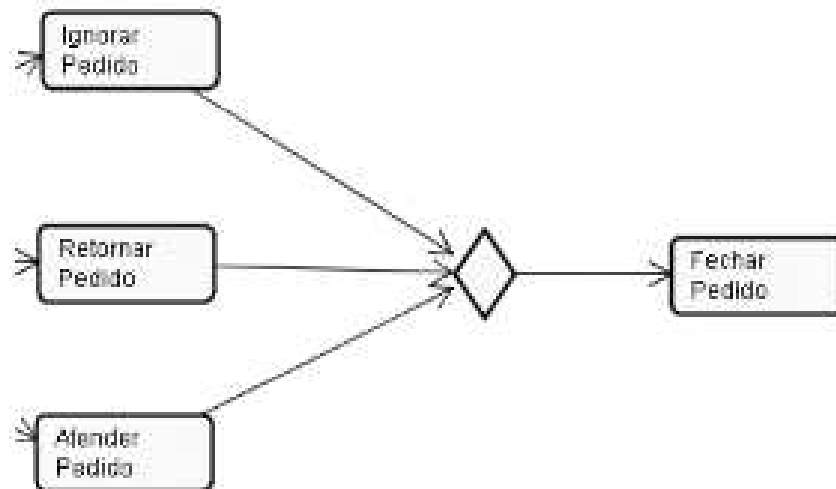




# Nó de "merge" (unificação)

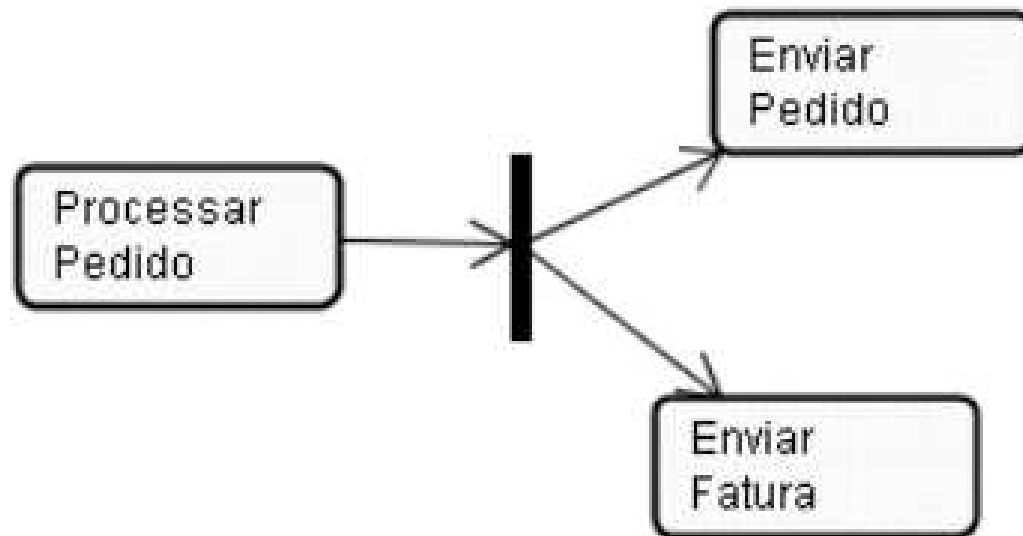
➡ Serve para juntar caminhos separados com o nó de decisão

➡ Usa o mesmo símbolo da decisão



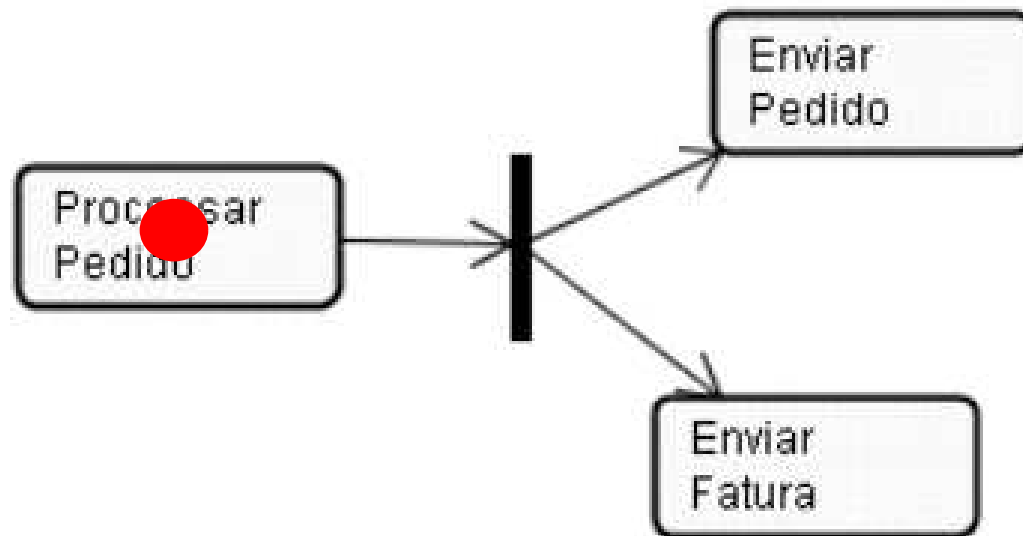
# Nó de "fork"

- ➡ Divide um fluxo em vários fluxos concorrentes (copiando o token)



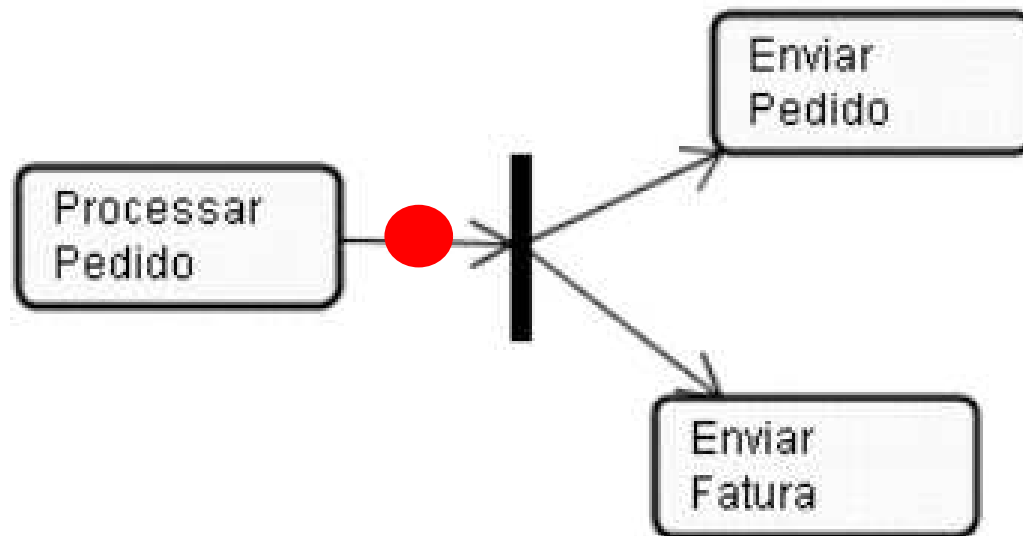
# Nó de "fork"

- ➡ Divide um fluxo em vários fluxos concorrentes (copiando o token)



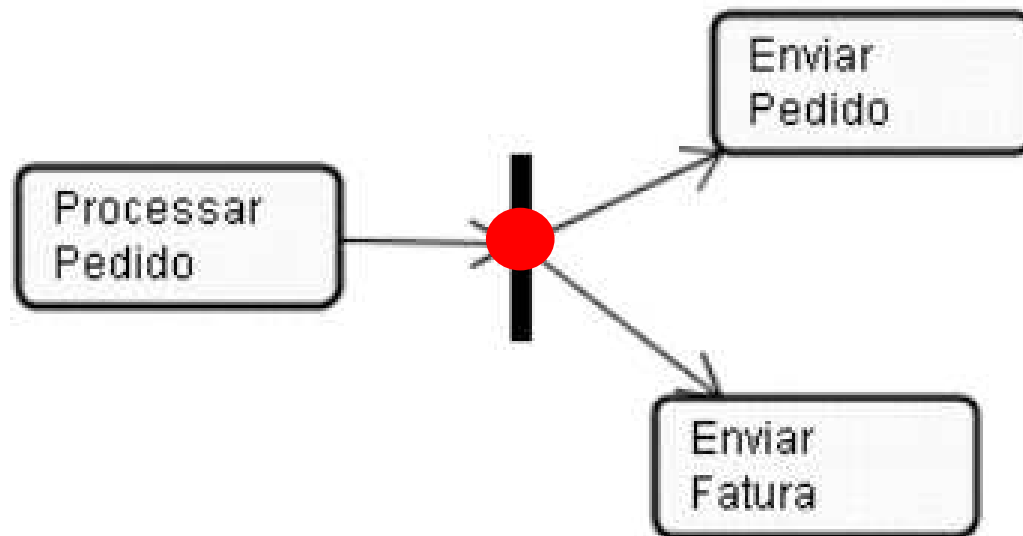
# Nó de "fork"

- ➡ Divide um fluxo em vários fluxos concorrentes (copiando o token)



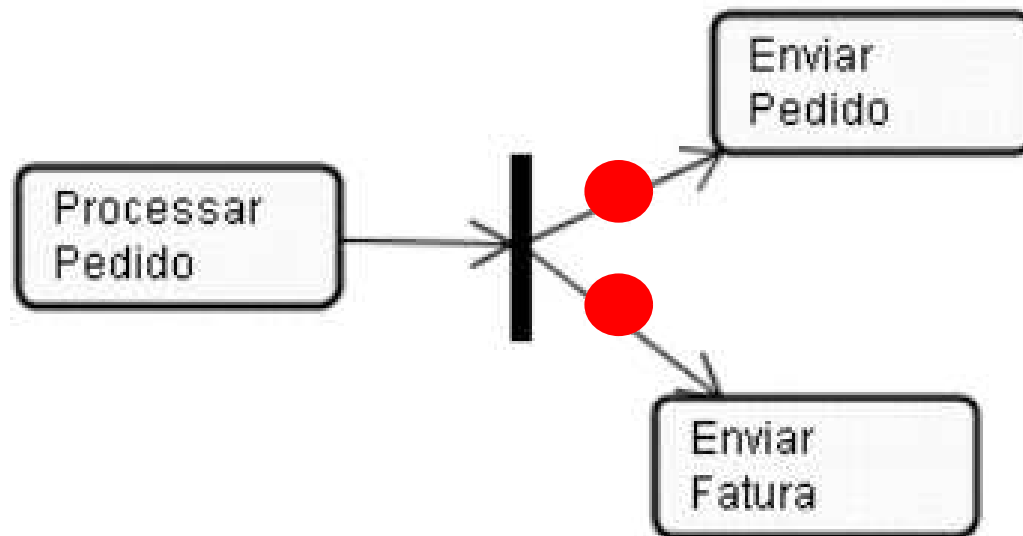
# Nó de "fork"

- ➡ Divide um fluxo em vários fluxos concorrentes (copiando o token)



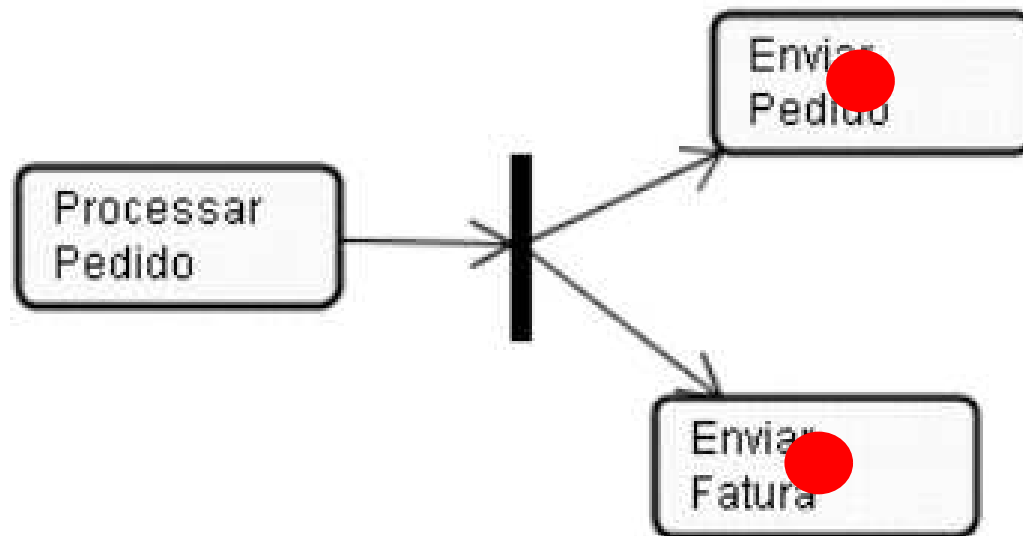
# Nó de "fork"

- ➡ Divide um fluxo em vários fluxos concorrentes (copiando o token)



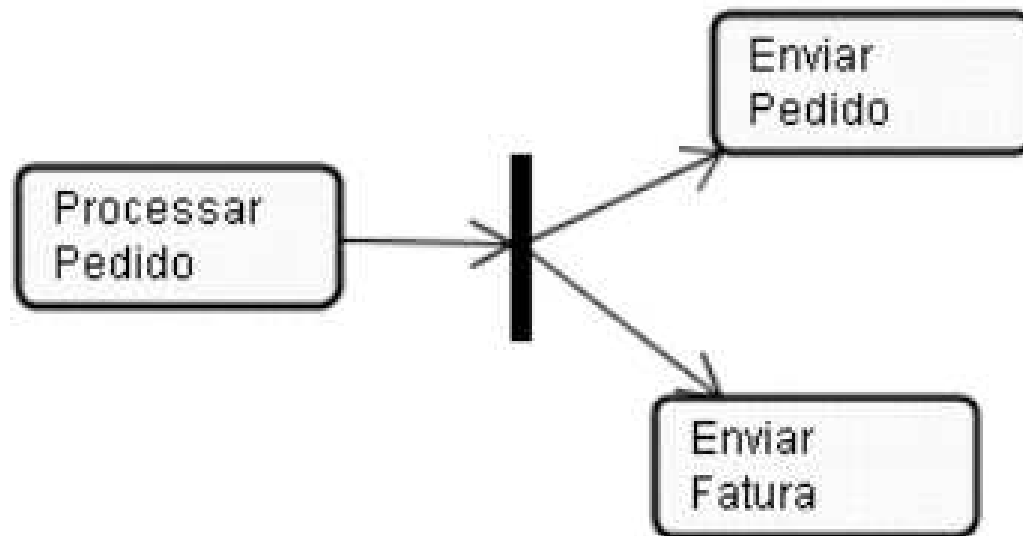
# Nó de "fork"

- ➡ Divide um fluxo em vários fluxos concorrentes (copiando o token)



# Nó de "fork"

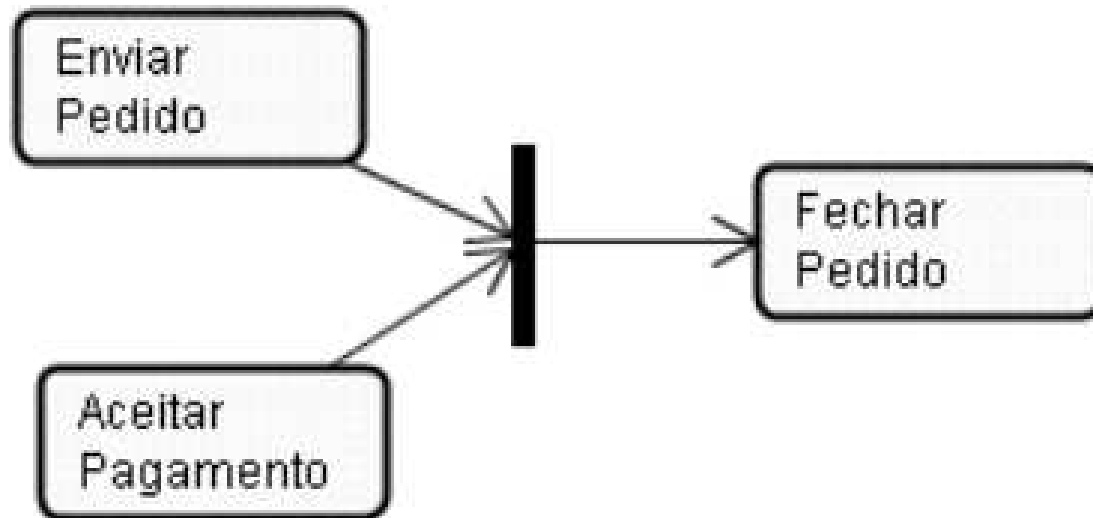
- ➡ Divide um fluxo em vários fluxos concorrentes (copiando o token)





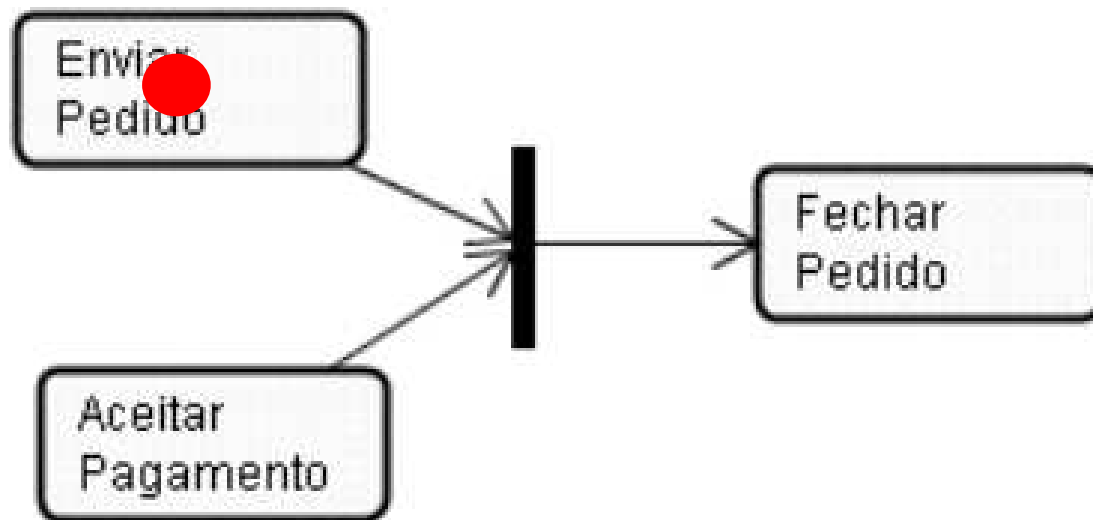
# Nó de "join" (sincronização)

- ➡ Unifica e sincroniza fluxos concorrentes
- ➡ Consome os tokens simultaneamente



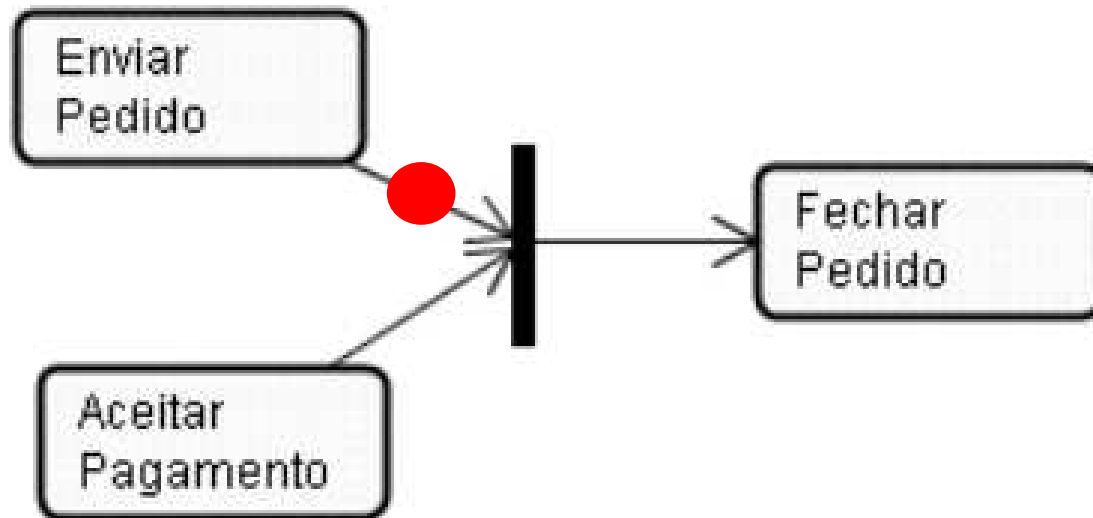
# Nó de "join" (sincronização)

- ➡ Unifica e sincroniza fluxos concorrentes
- ➡ Consome os tokens simultaneamente



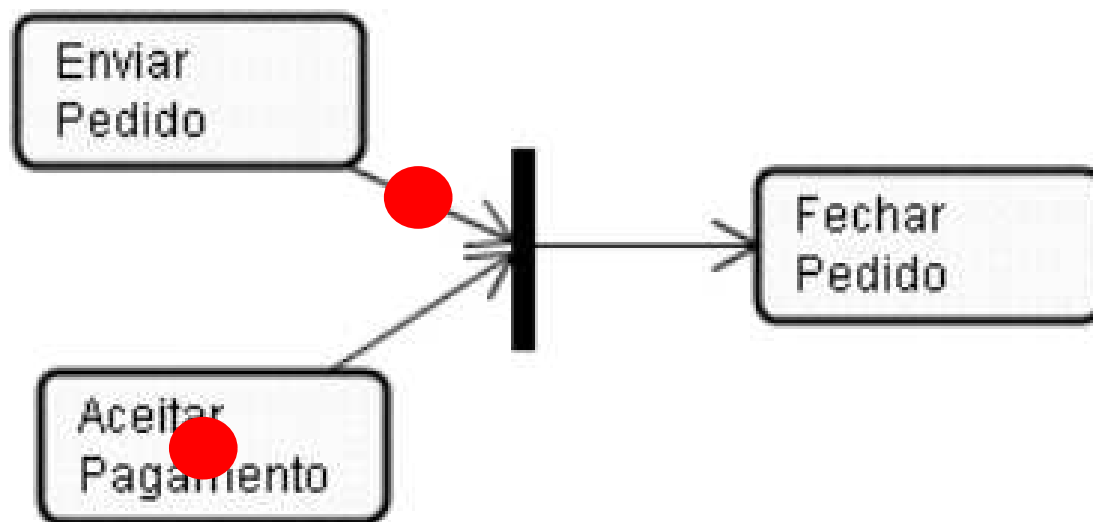
# Nó de "join" (sincronização)

- ➡ Unifica e sincroniza fluxos concorrentes
- ➡ Consome os tokens simultaneamente



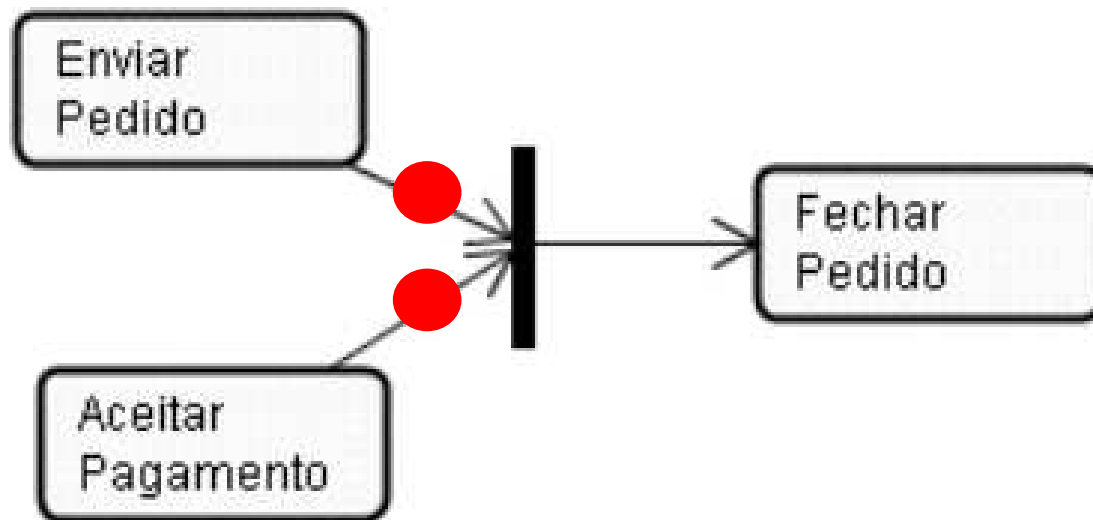
# Nó de "join" (sincronização)

- ➡ Unifica e sincroniza fluxos concorrentes
- ➡ Consome os tokens simultaneamente



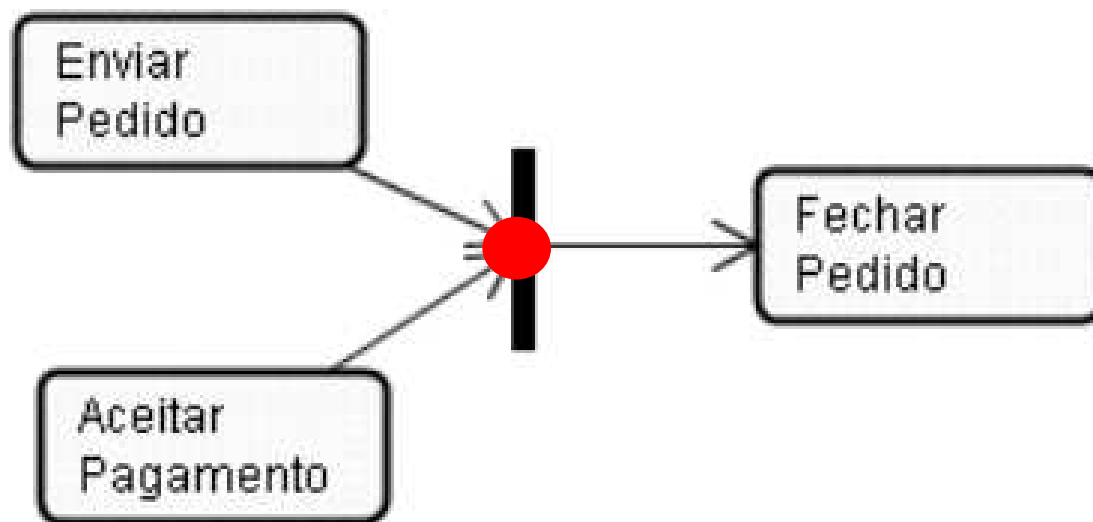
# Nó de "join" (sincronização)

- ➡ Unifica e sincroniza fluxos concorrentes
- ➡ Consome os tokens simultaneamente



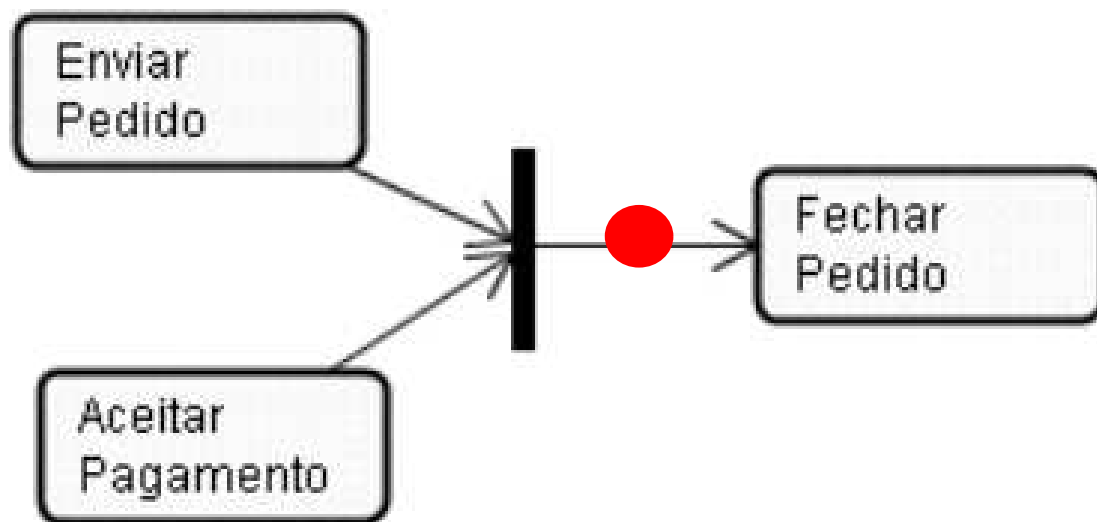
# Nó de "join" (sincronização)

- ➡ Unifica e sincroniza fluxos concorrentes
- ➡ Consome os tokens simultaneamente



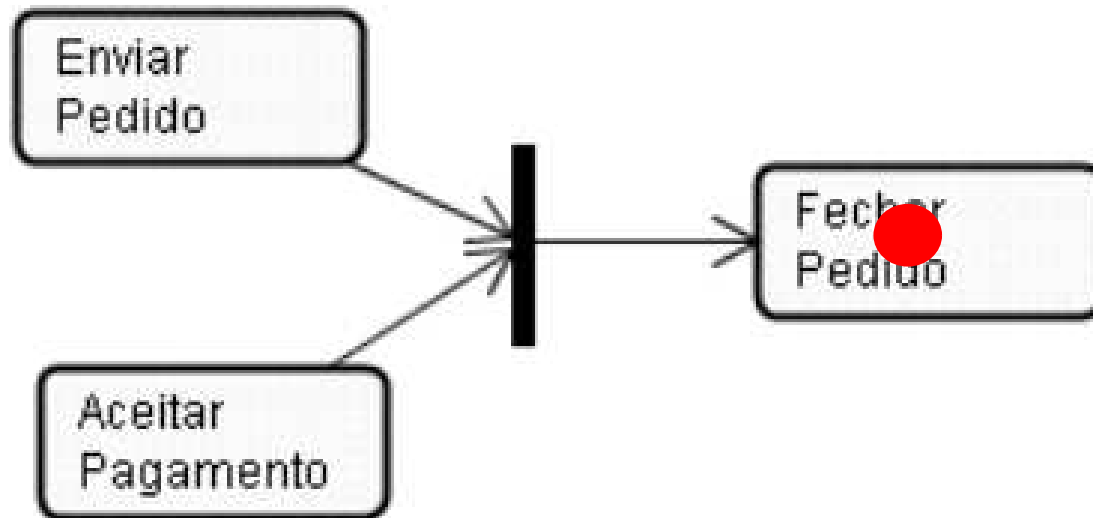
# Nó de "join" (sincronização)

- ➡ Unifica e sincroniza fluxos concorrentes
- ➡ Consome os tokens simultaneamente



# Nó de "join" (sincronização)

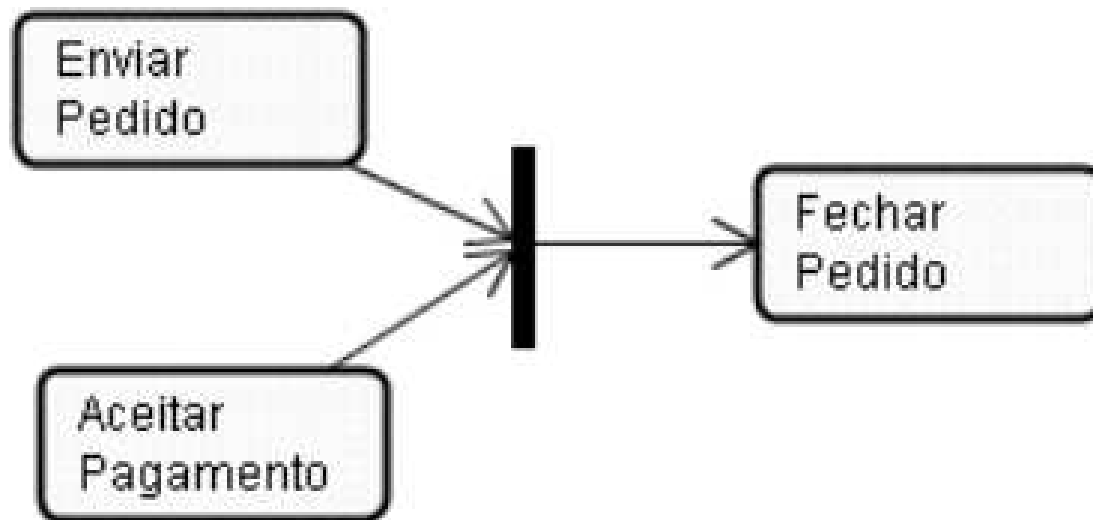
- ➡ Unifica e sincroniza fluxos concorrentes
- ➡ Consome os tokens simultaneamente





# Nó de "join" (sincronização)

- ➡ Unifica e sincroniza fluxos concorrentes
- ➡ Consome os tokens simultaneamente



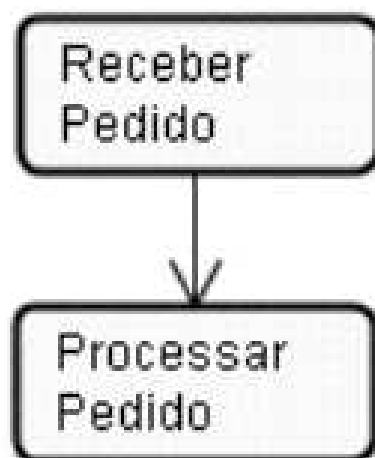
# Nó de Objeto

- ➡ Representa um objeto
- ➡ Aparece em fluxos de objeto entre duas ações

Nome do Objeto

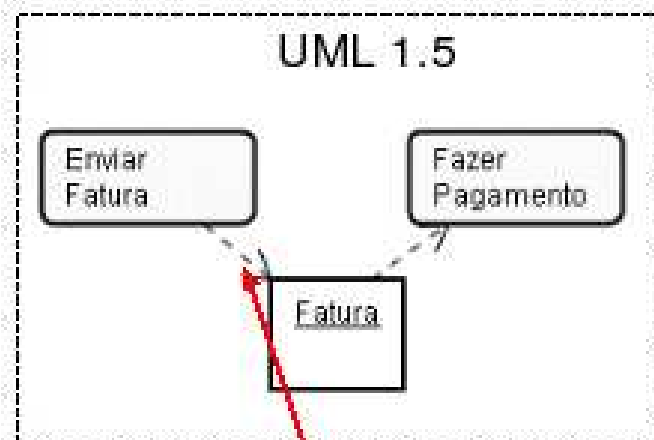
# Fluxo de Controle

➡ Setas simples

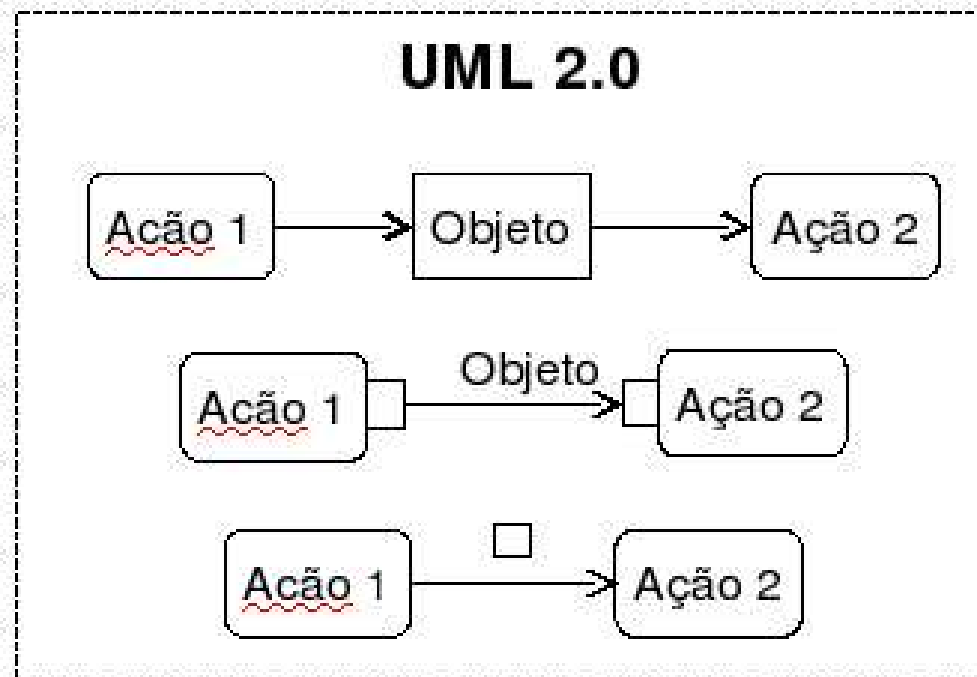


# Fluxo de Objeto

➡ Indica o objeto que é passado entre as ações



Ferramentas utilizadas  
ainda não atendem  
plenamente a UML 2.0

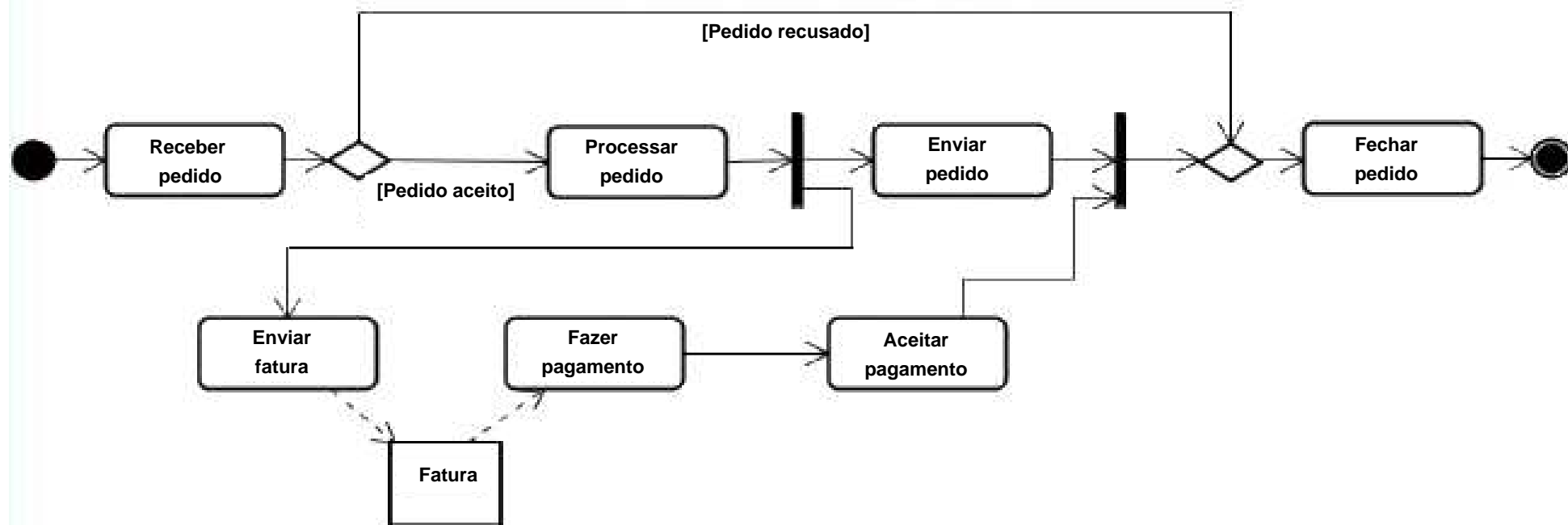


# Objetos que são sinais

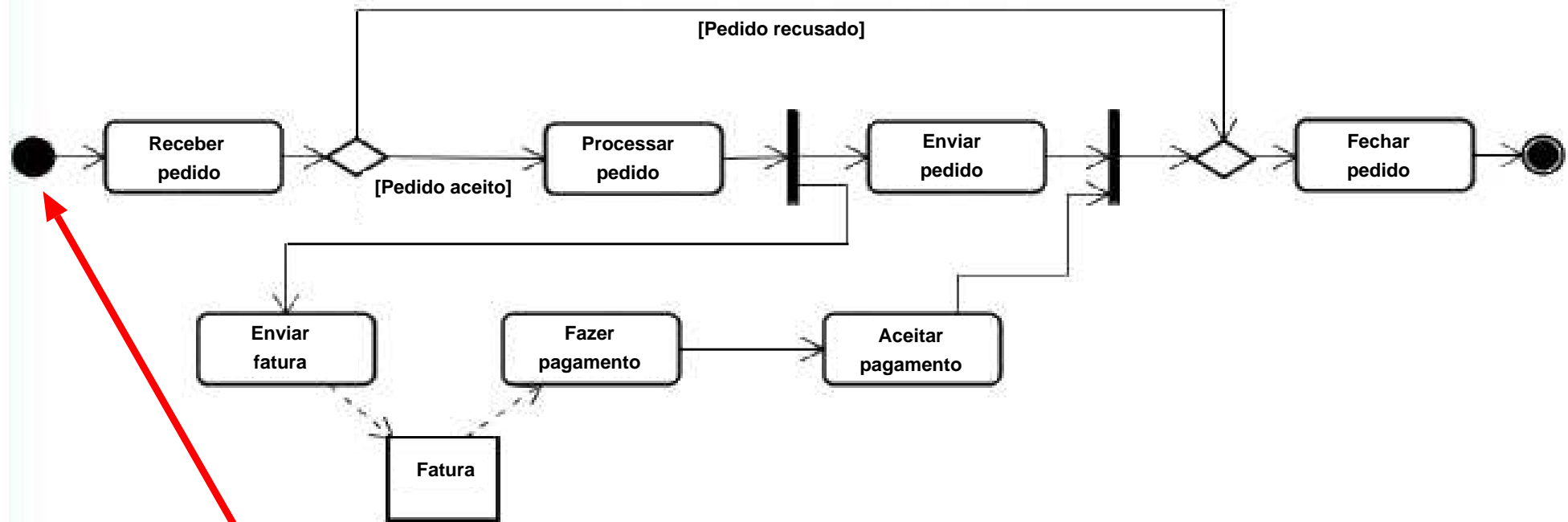
➡ Objetos que são sinais usam uma notação especial



# Entendendo o Exemplo

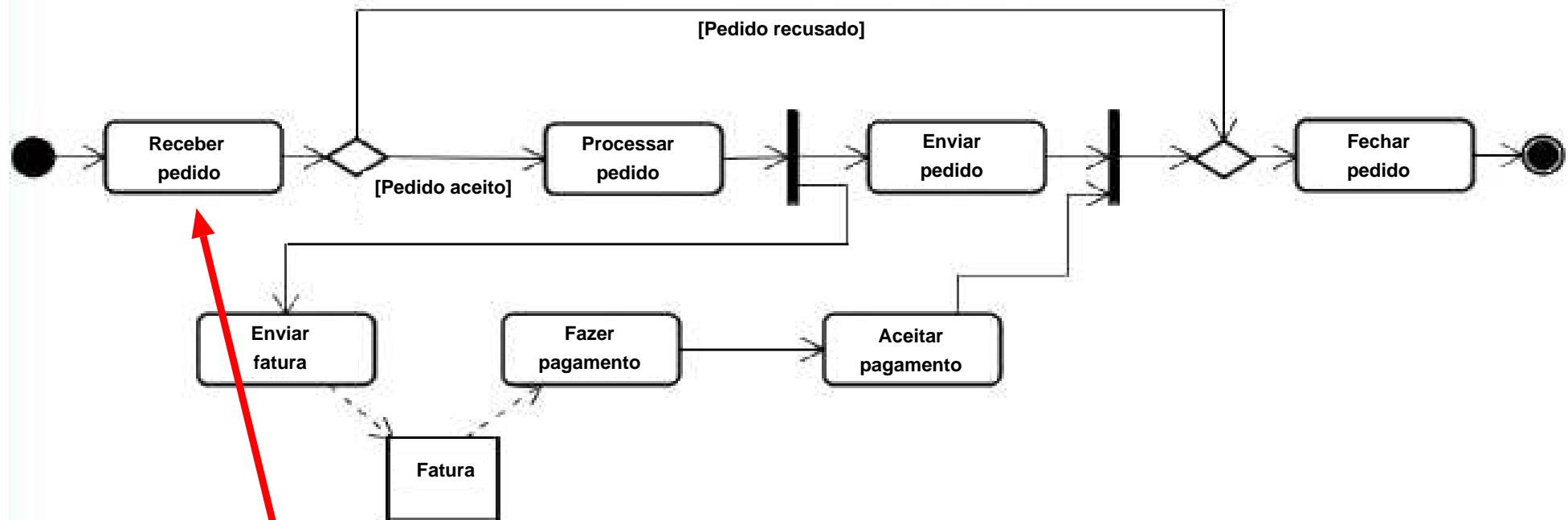


# Entendendo o Exemplo



(Nó Inicial)  
A atividade se inicia a partir desse nó

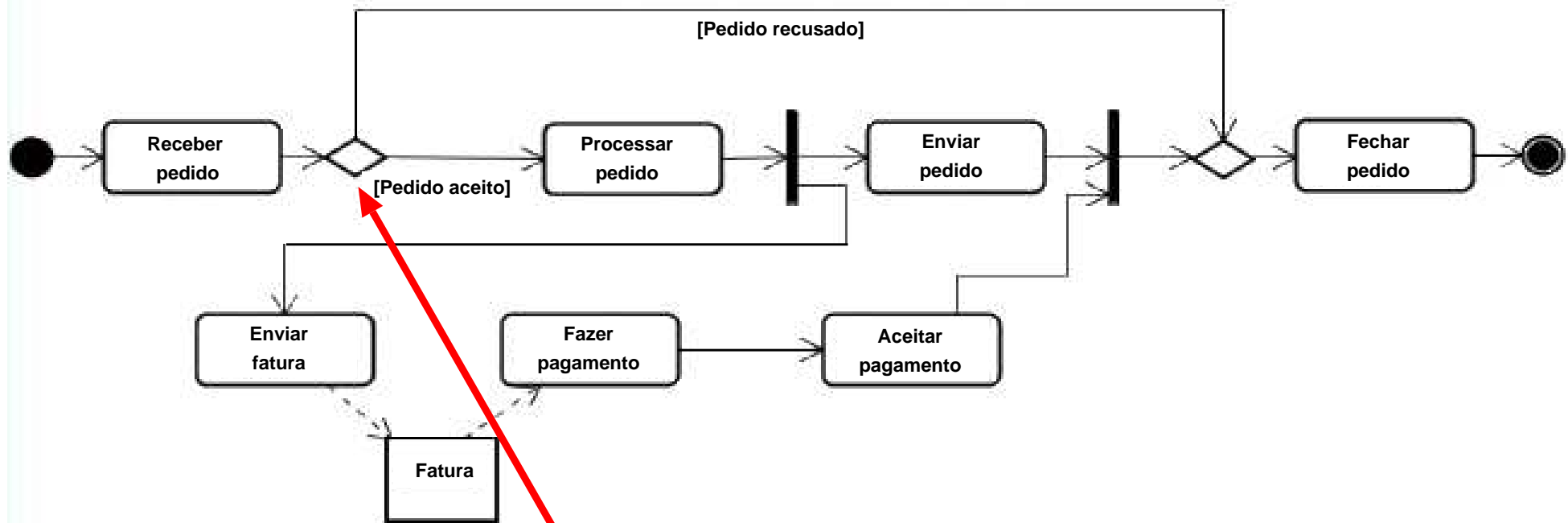
# Entendendo o Exemplo



(Ação)  
Essa é a primeira ação da atividade

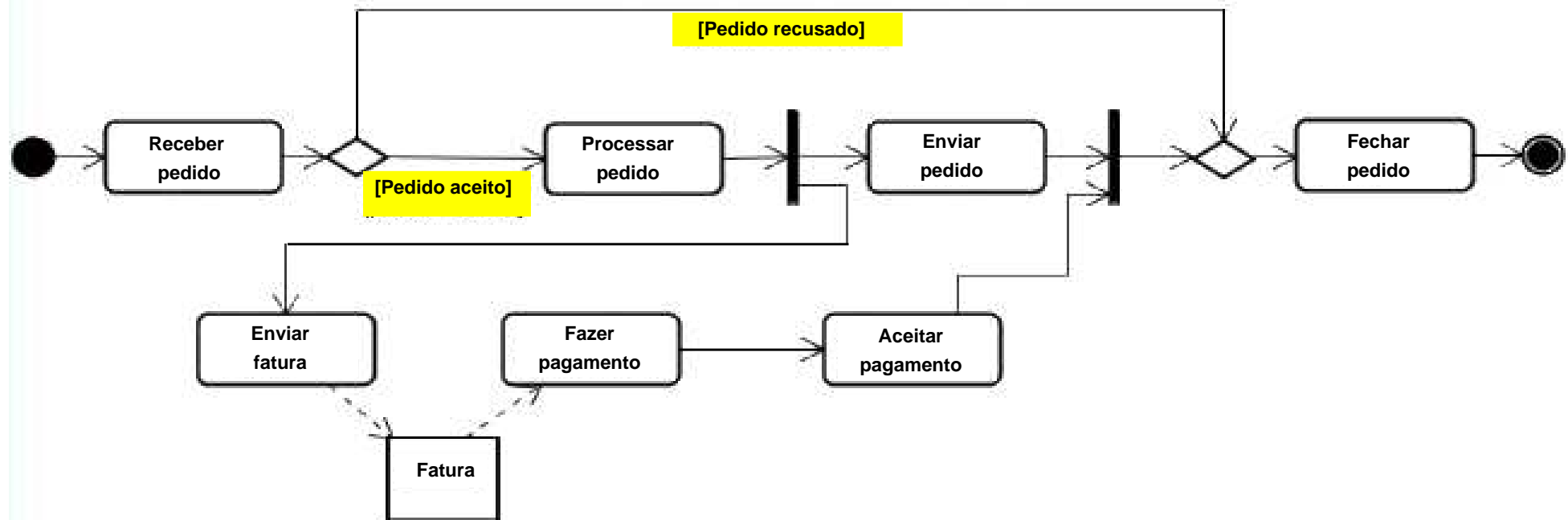


# Entendendo o Exemplo

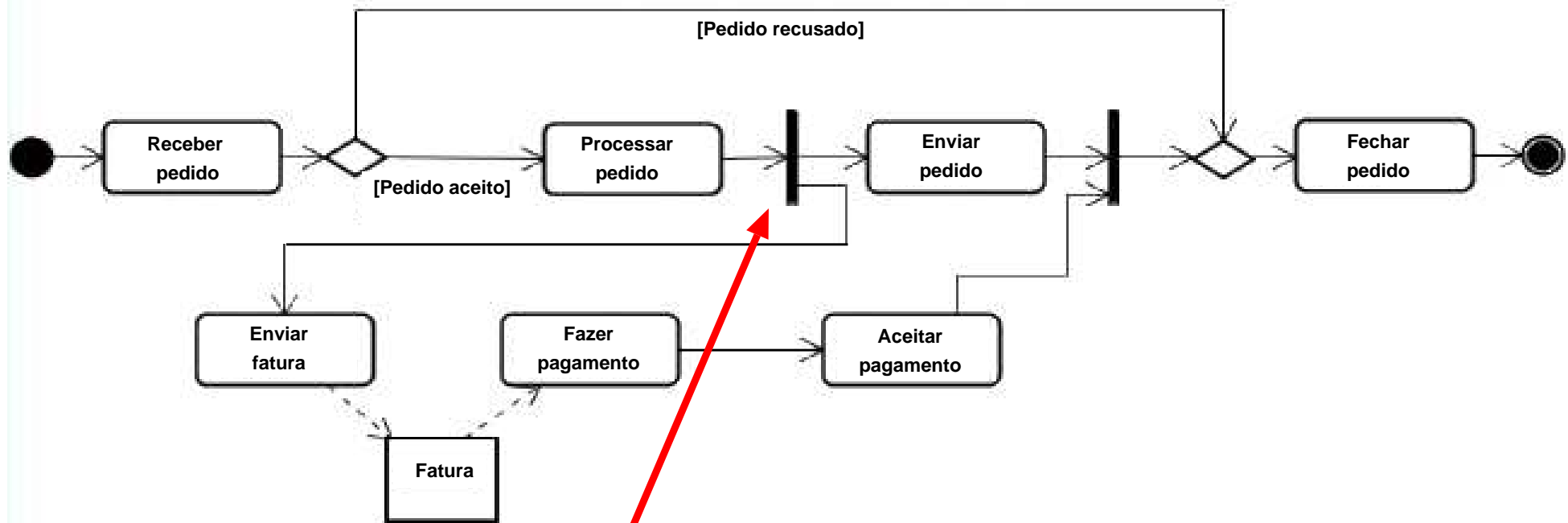


(Decisão)  
Agora uma decisão.  
O pedido foi recusado ou aceito?  
Veja as arestas com as "guardas"

# Entendendo o Exemplo

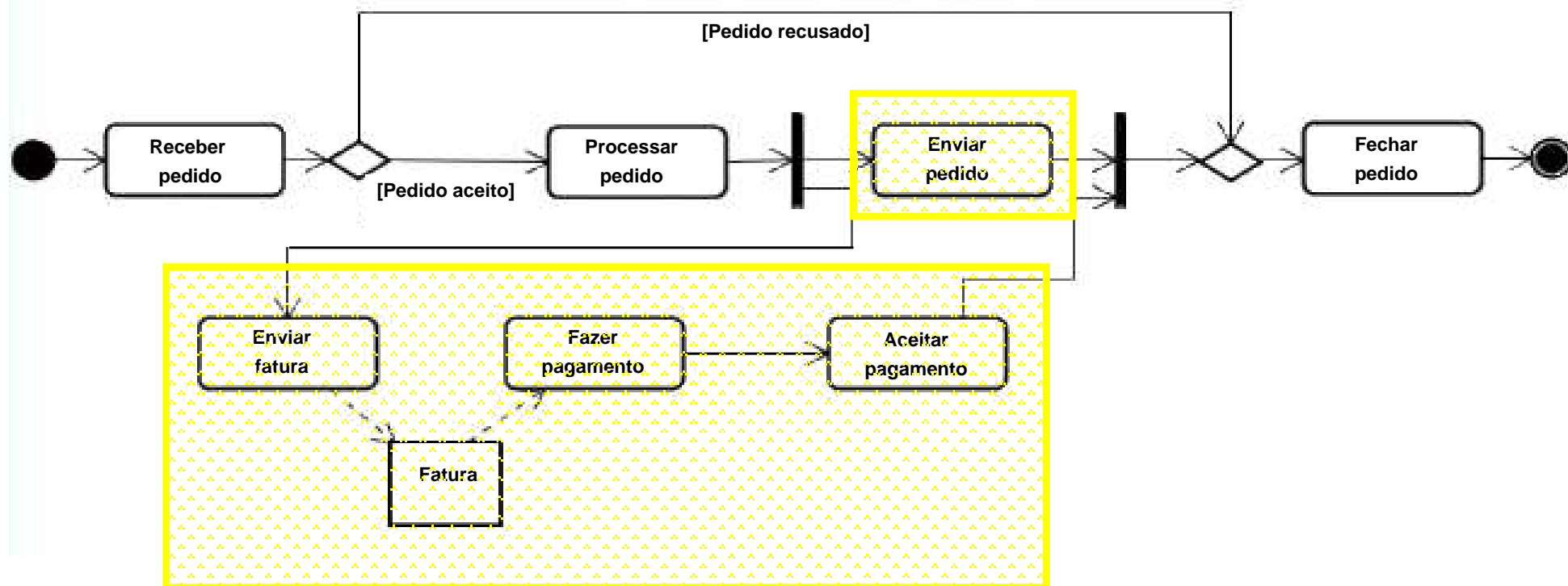


# Entendendo o Exemplo

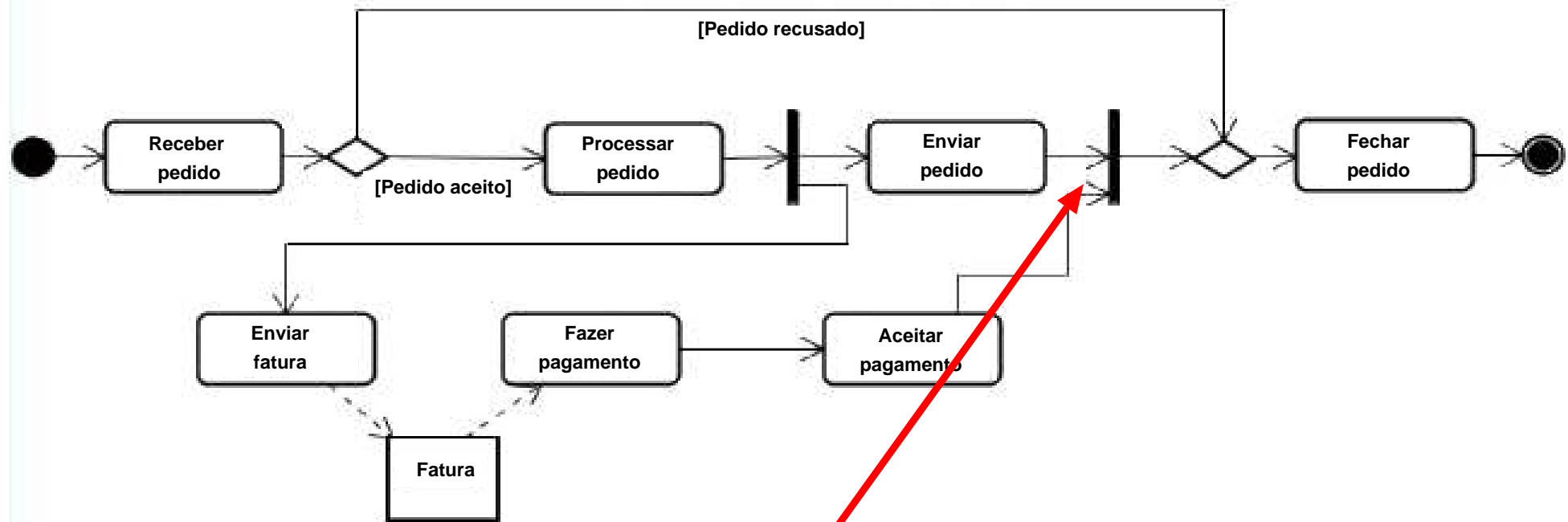


(Fork)  
Aqui iniciamos dois caminhos paralelos

# Entendendo o Exemplo

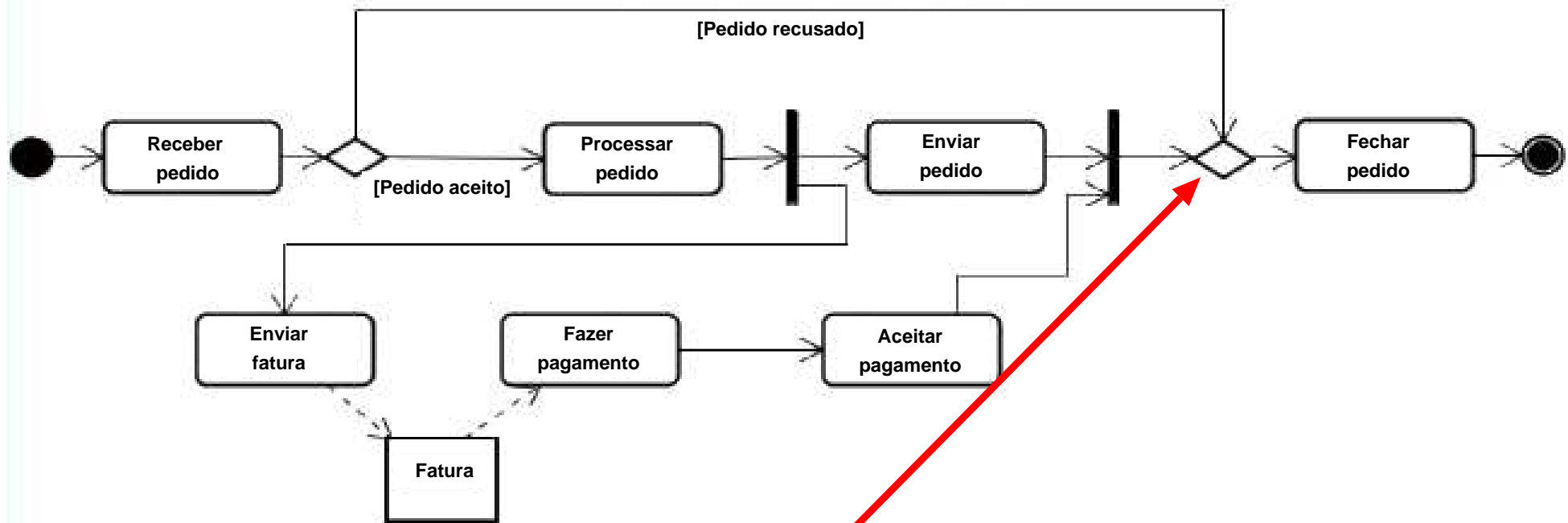


# Entendendo o Exemplo



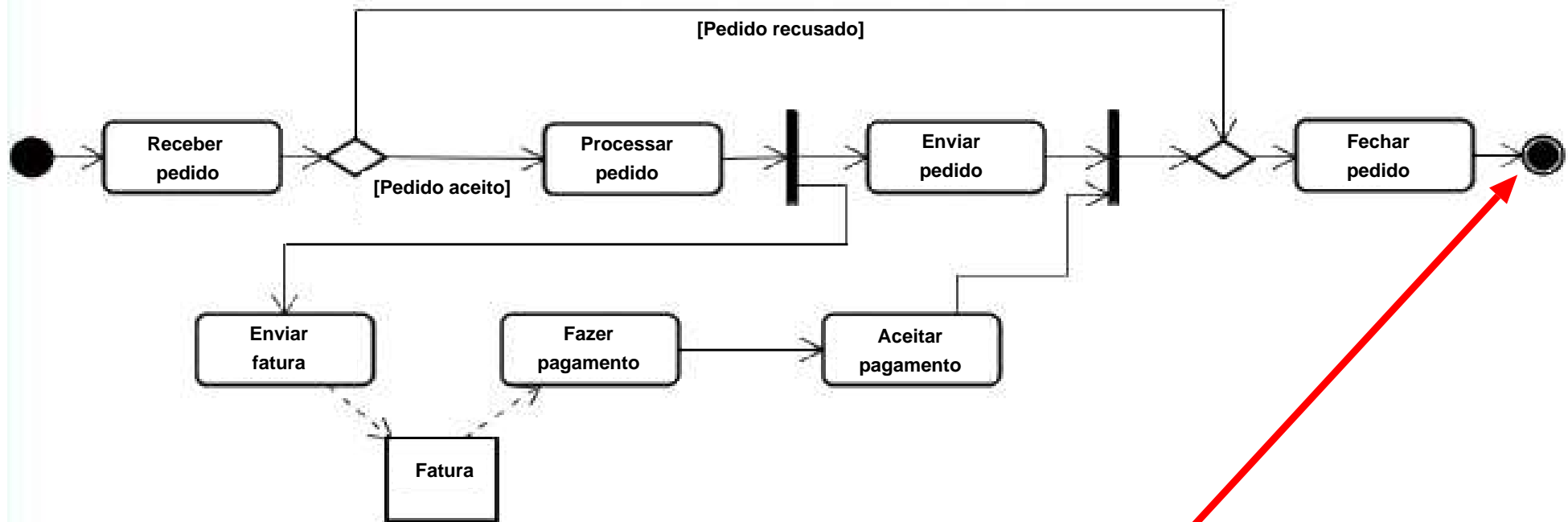
(Join)  
Aqui os caminhos são unidos  
e sincronizados

# Entendendo o Exemplo



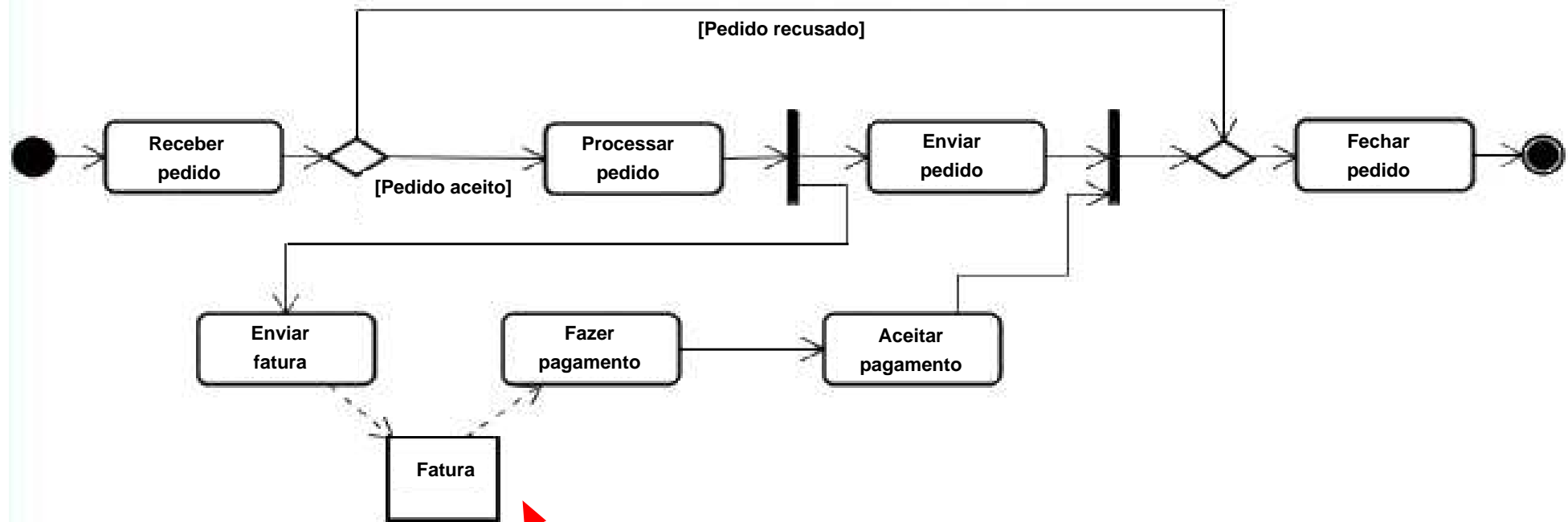
(Merge)  
Se o pedido não foi aceito, "pulamos"  
direto para cá

# Entendendo o Exemplo



(Nó Final)  
Aqui acaba a atividade

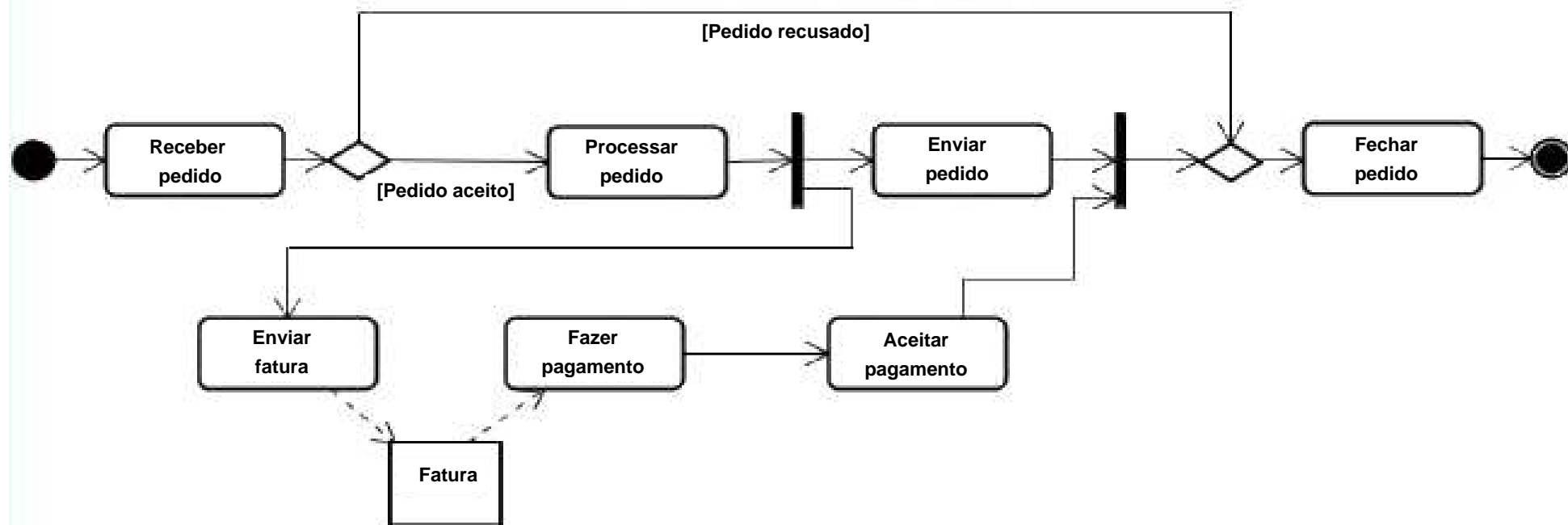
# Entendendo o Exemplo



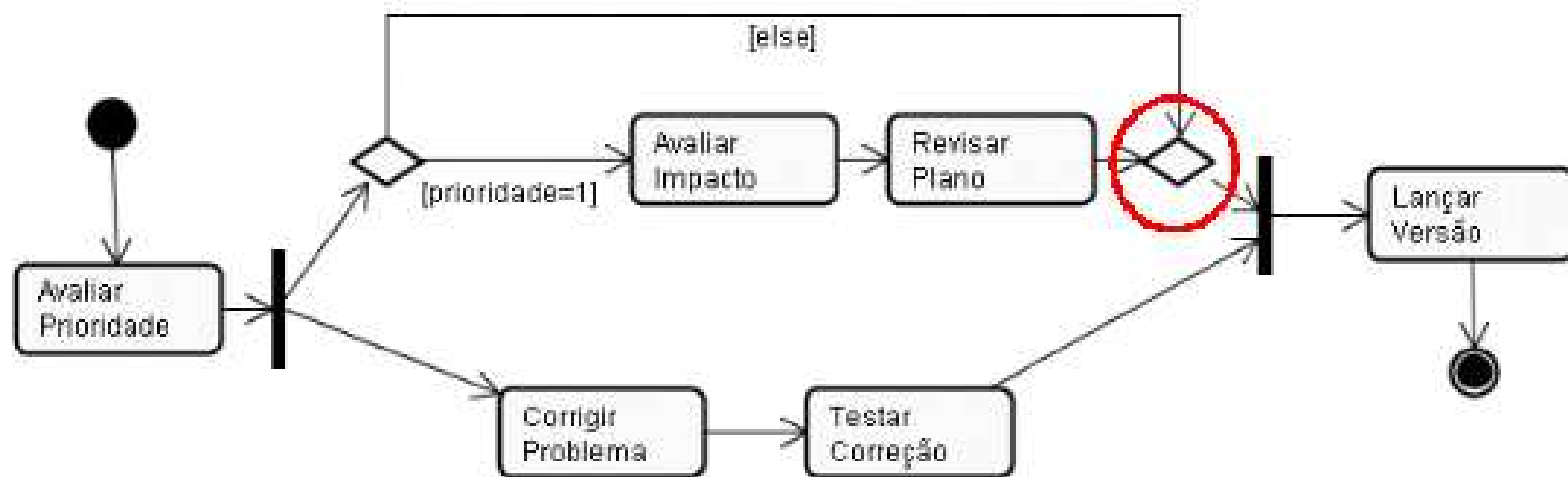
(Objeto)  
Objeto passando entre as ações.



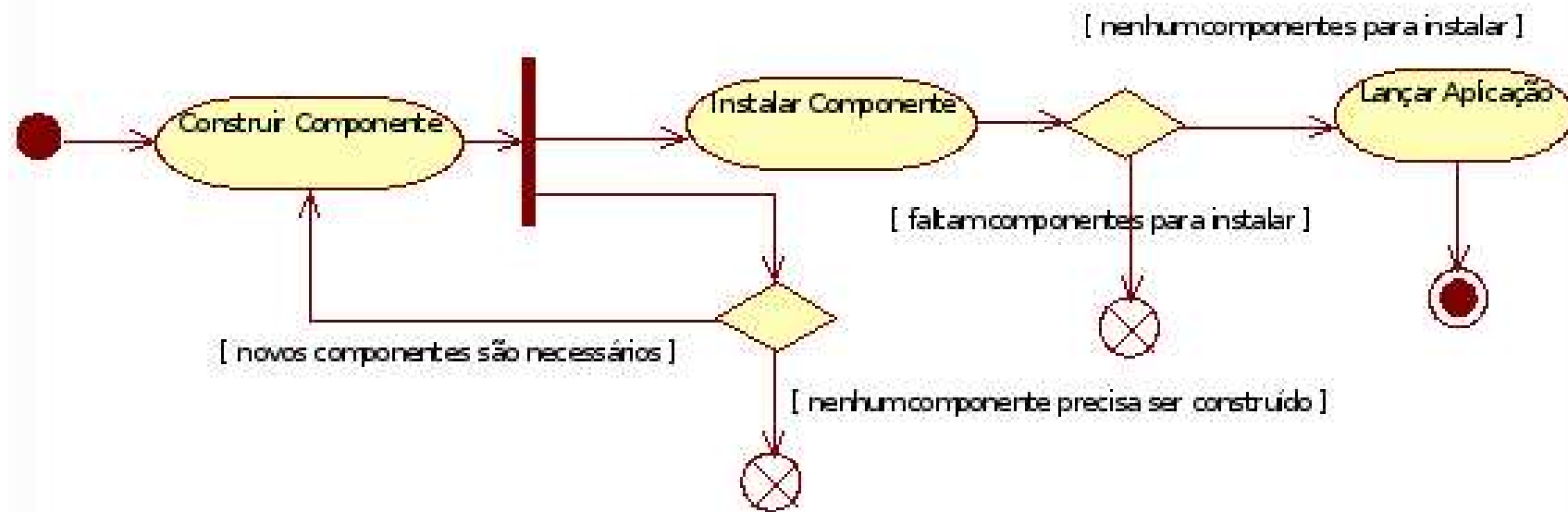
# Entendendo o Exemplo



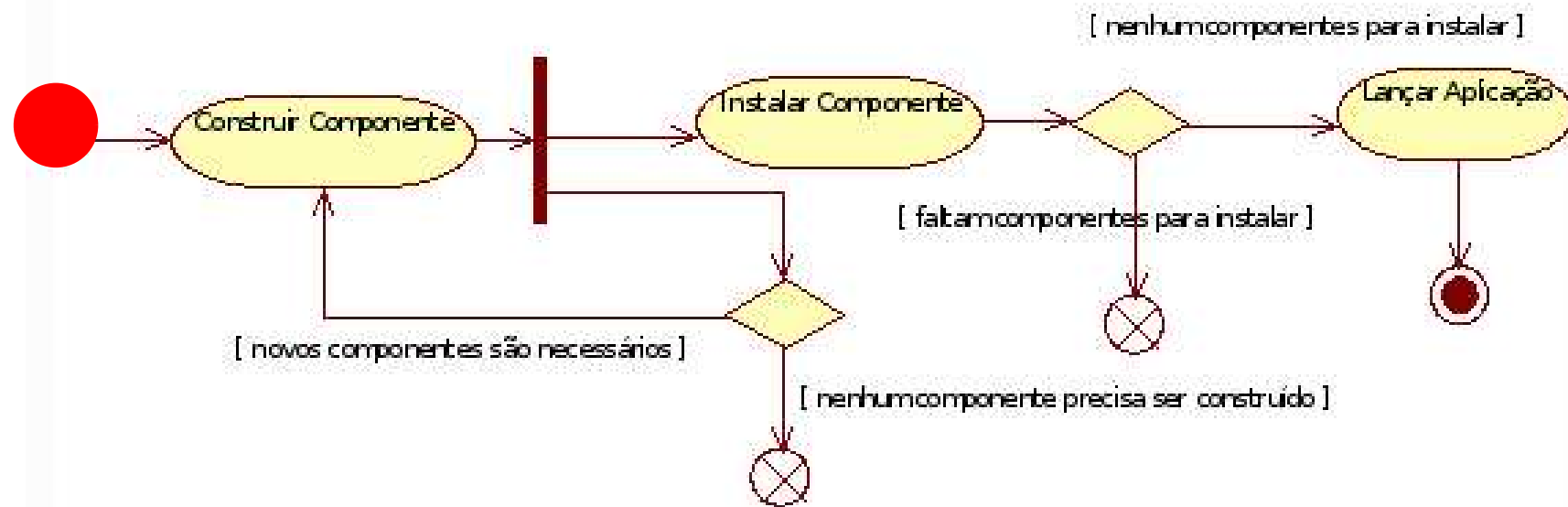
# Cuidados: O sincronismo exige que todos os fluxos cheguem



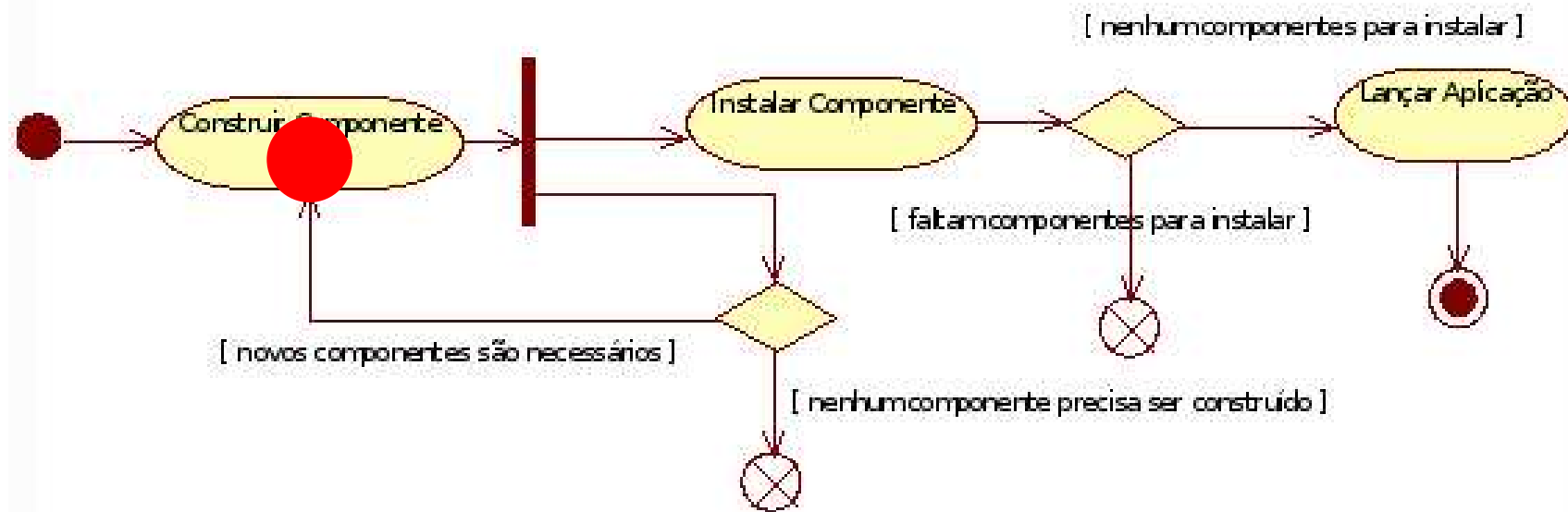
# Fluxos Terminando



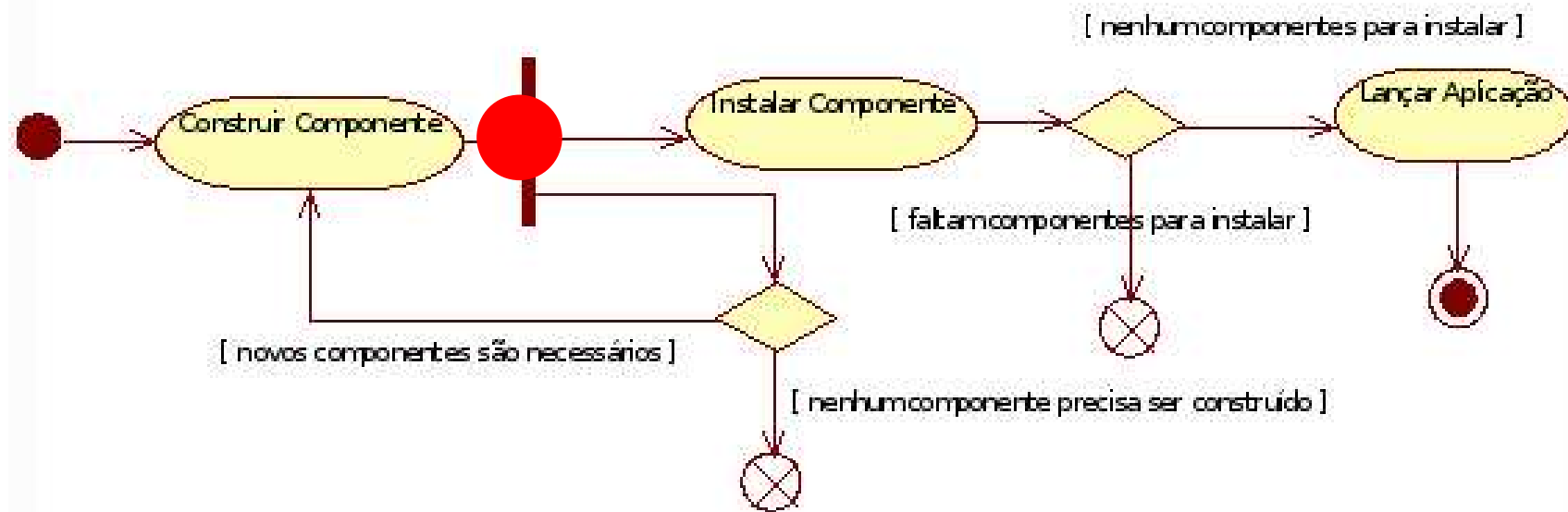
# Fluxos Terminando



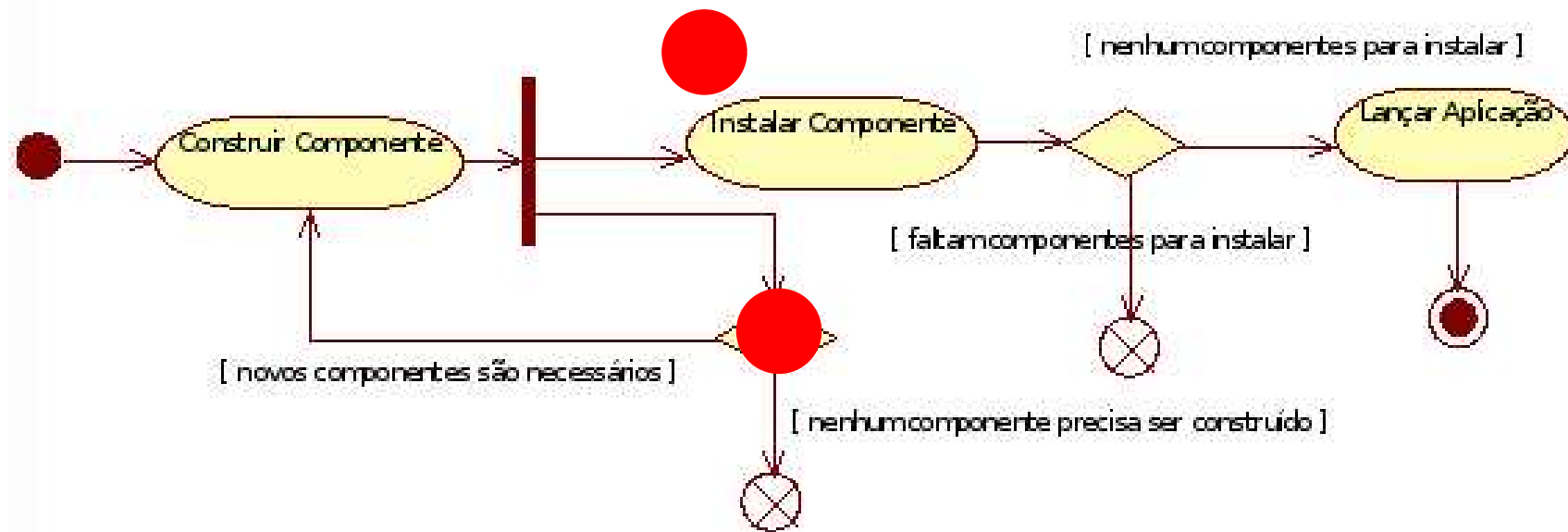
# Fluxos Terminando



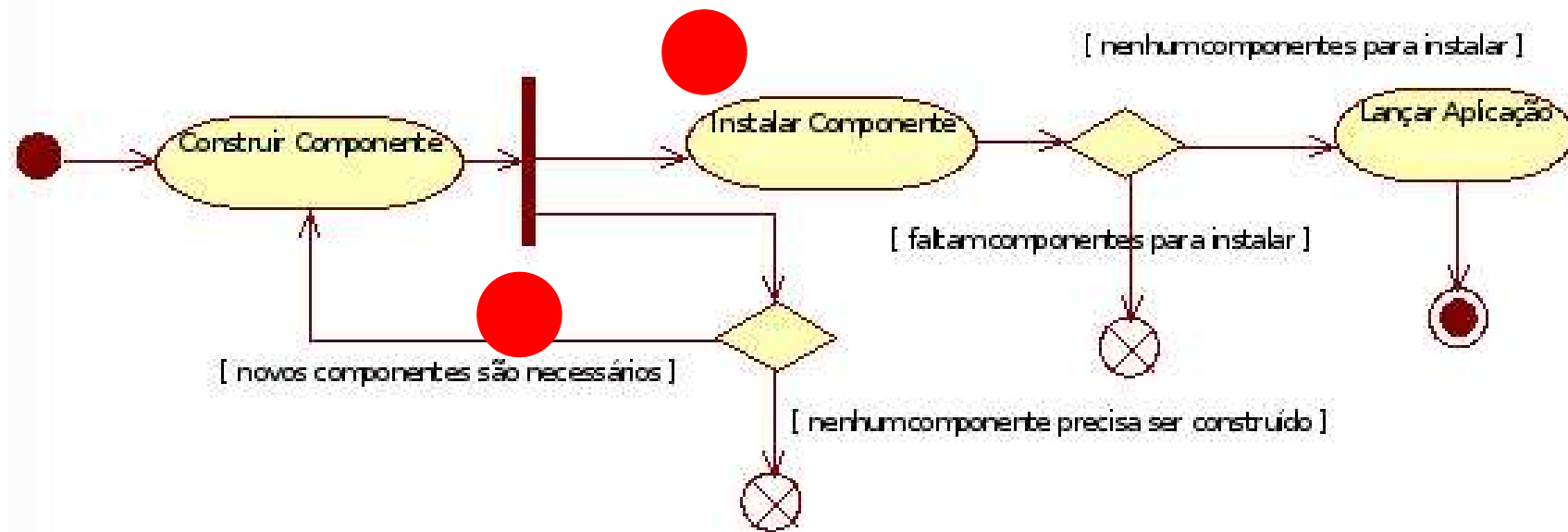
# Fluxos Terminando



# Fluxos Terminando

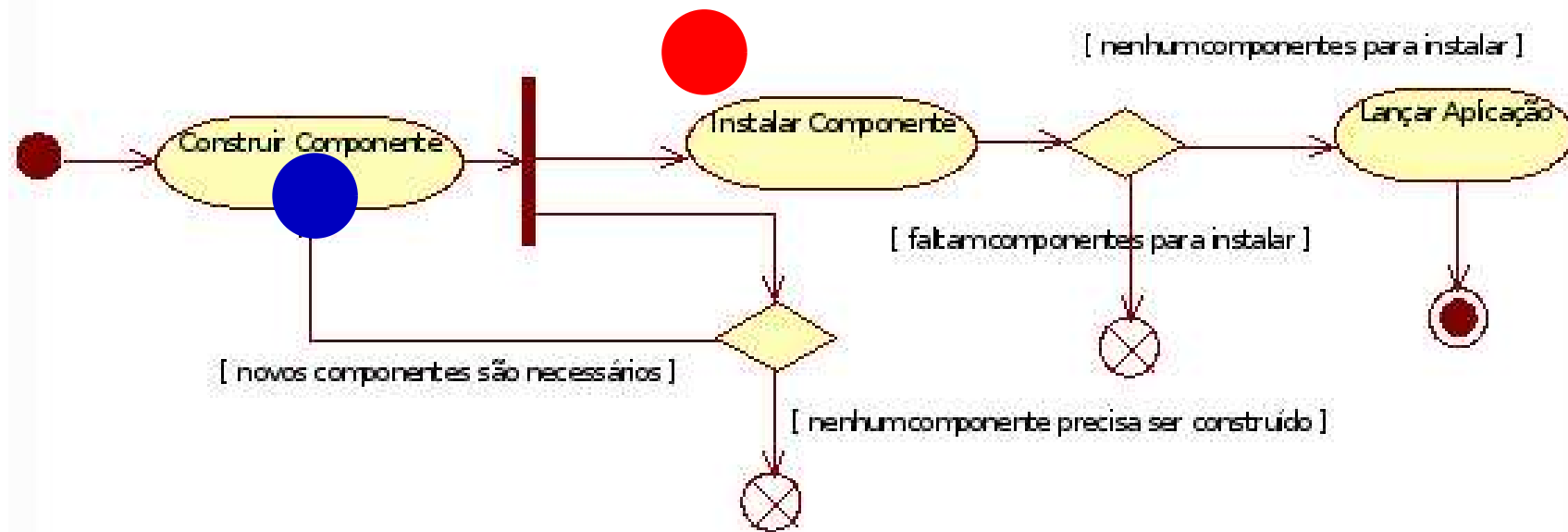


# Fluxos Terminando

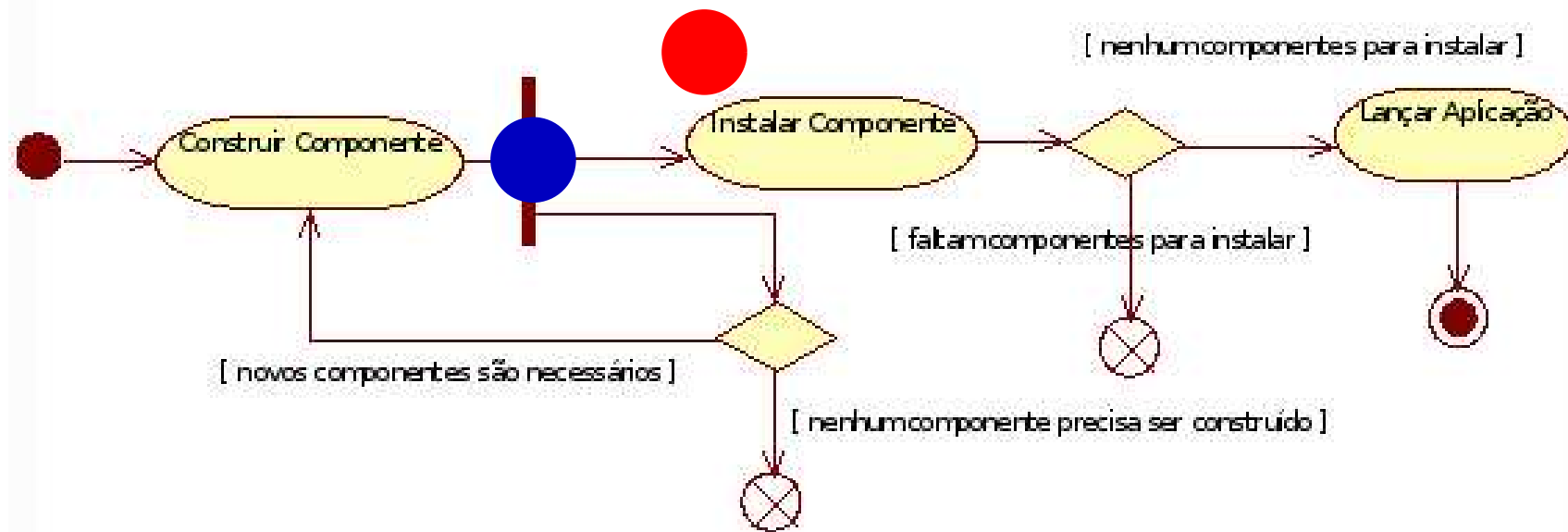




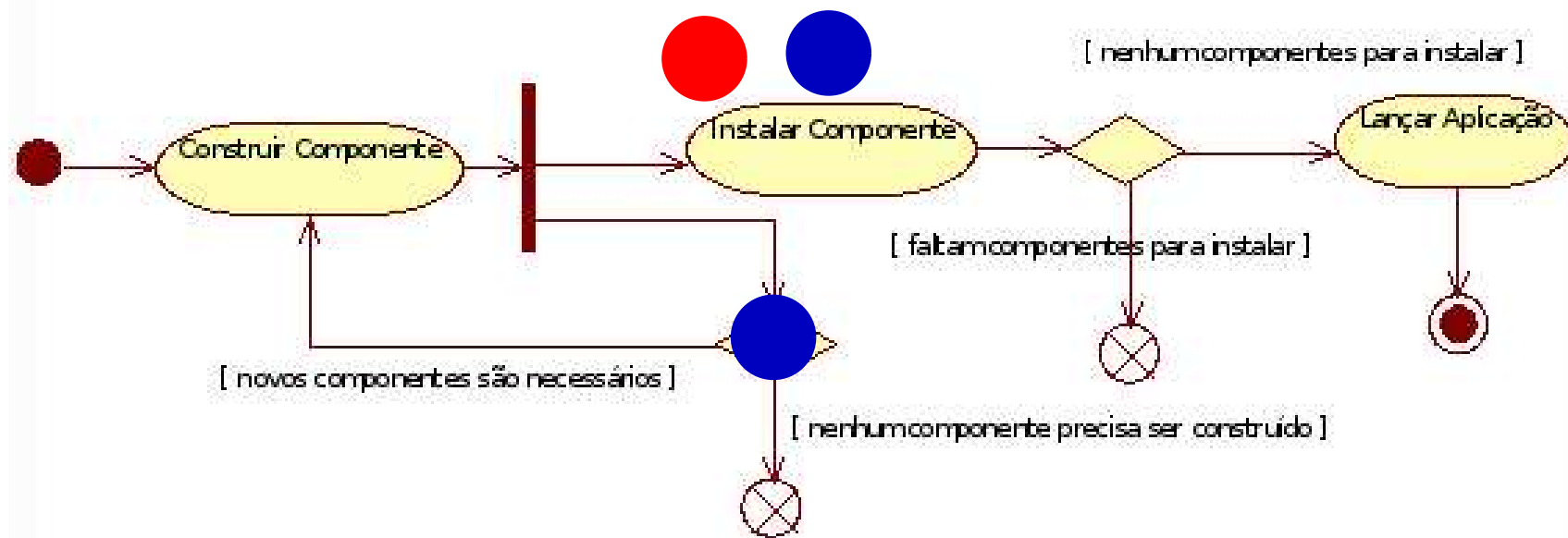
# Fluxos Terminando



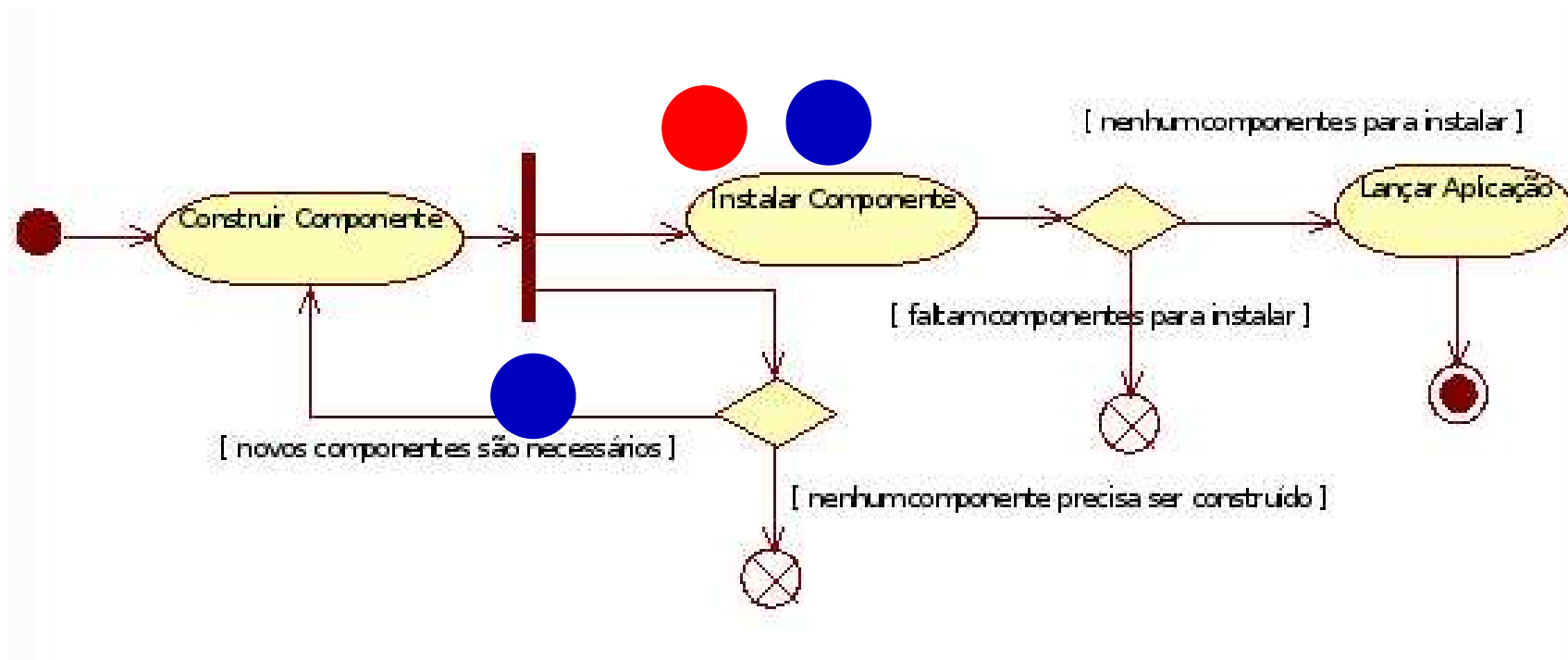
# Fluxos Terminando



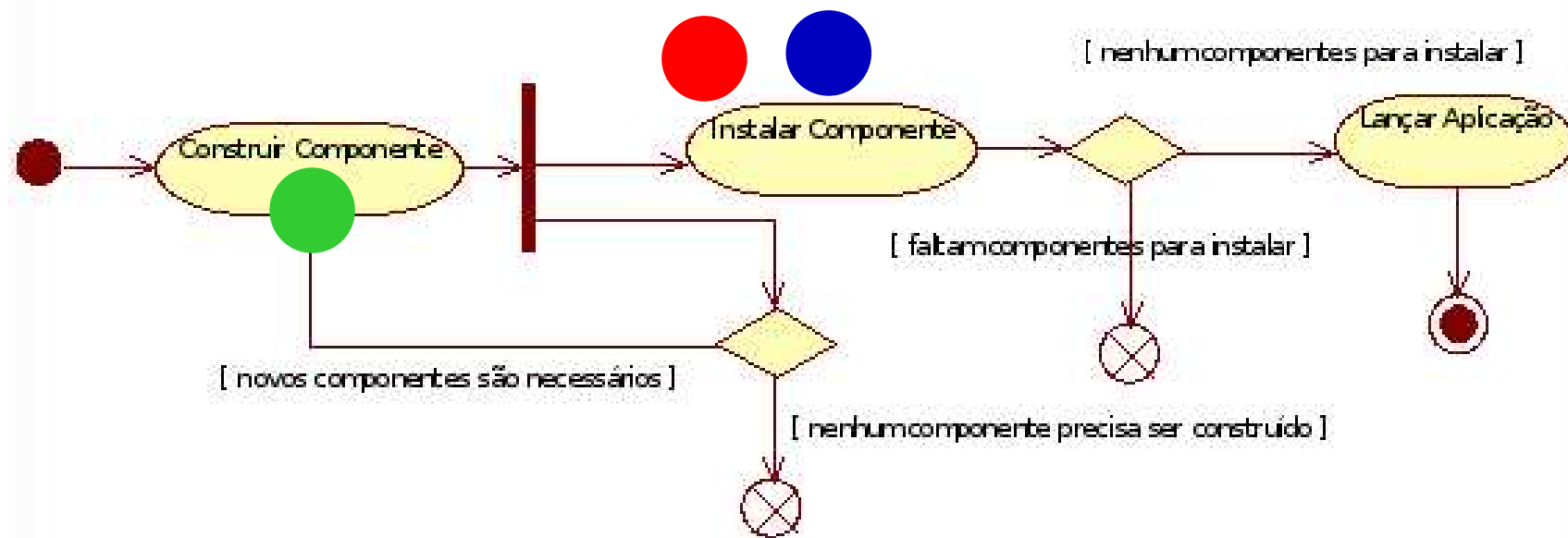
# Fluxos Terminando



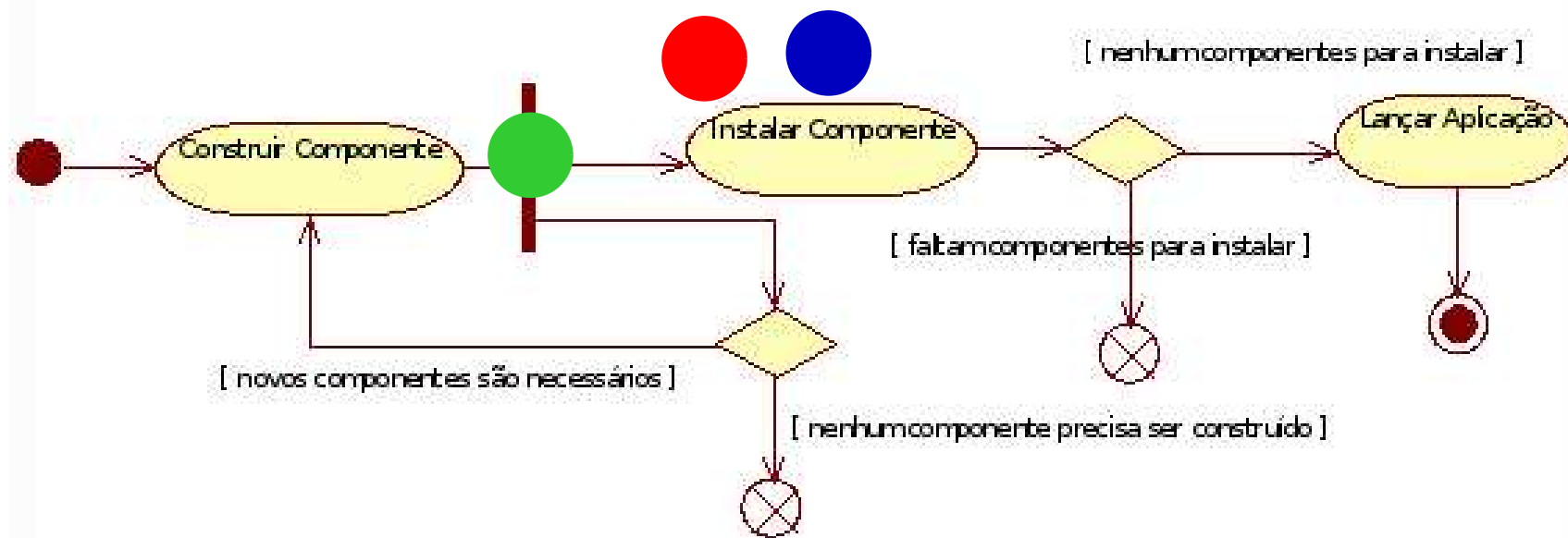
# Fluxos Terminando



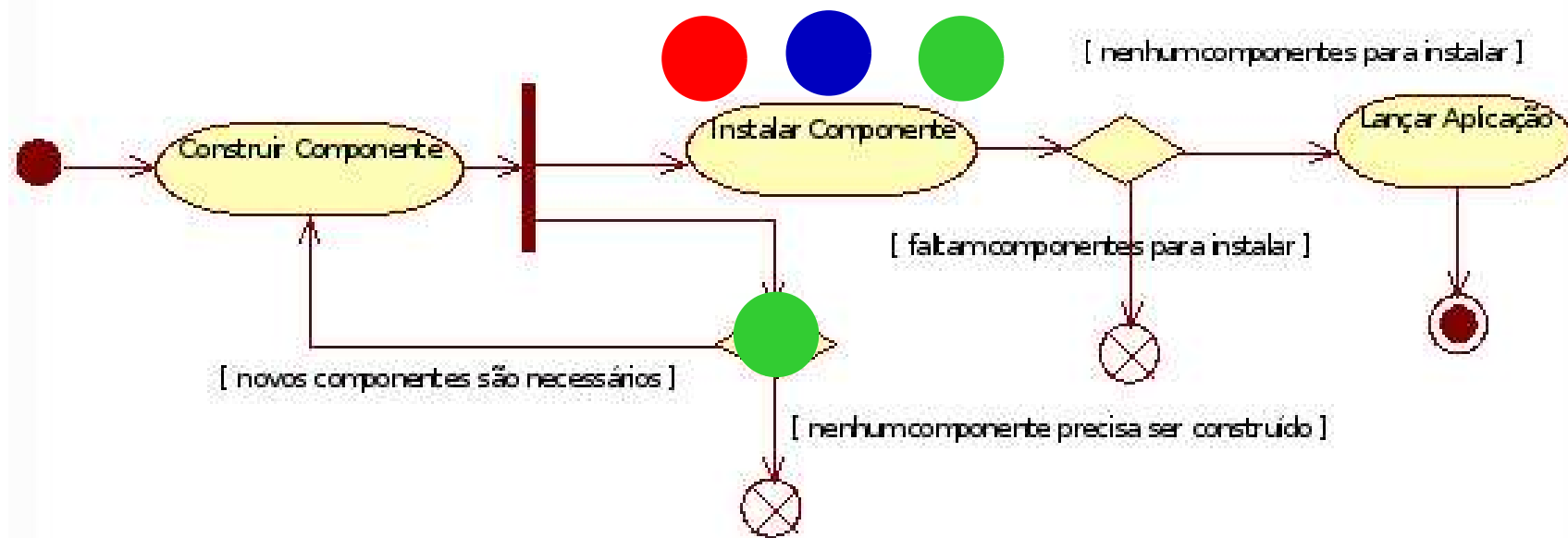
# Fluxos Terminando



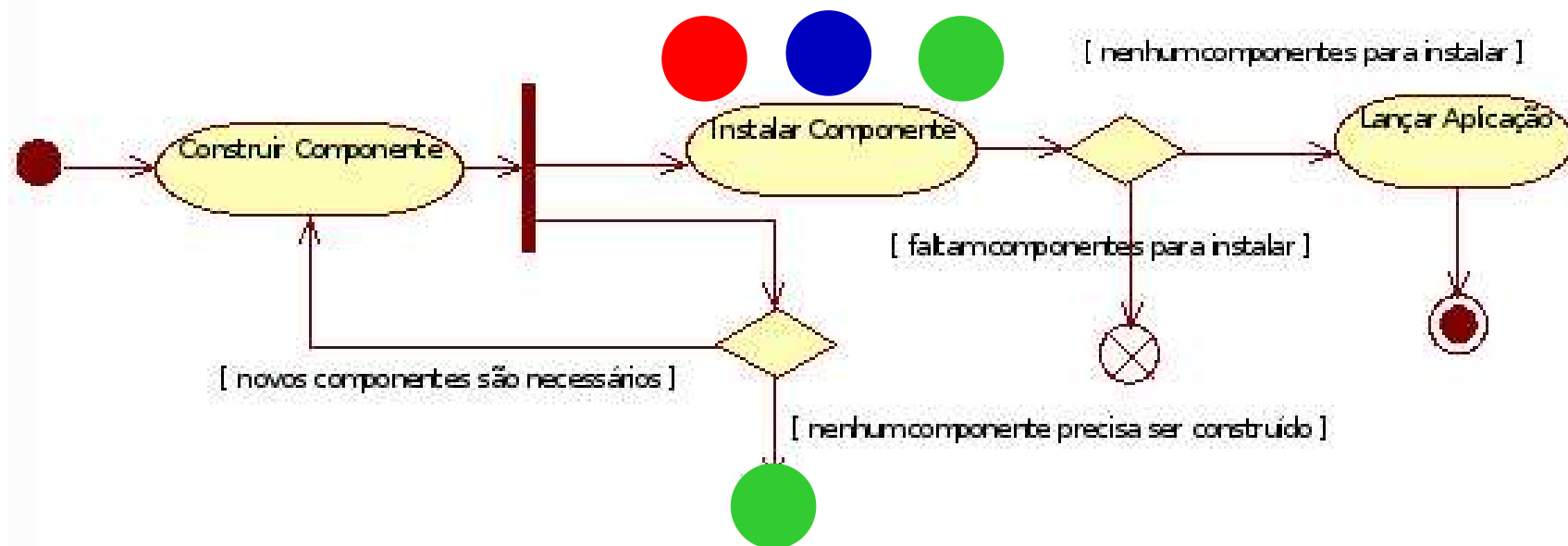
# Fluxos Terminando



# Fluxos Terminando

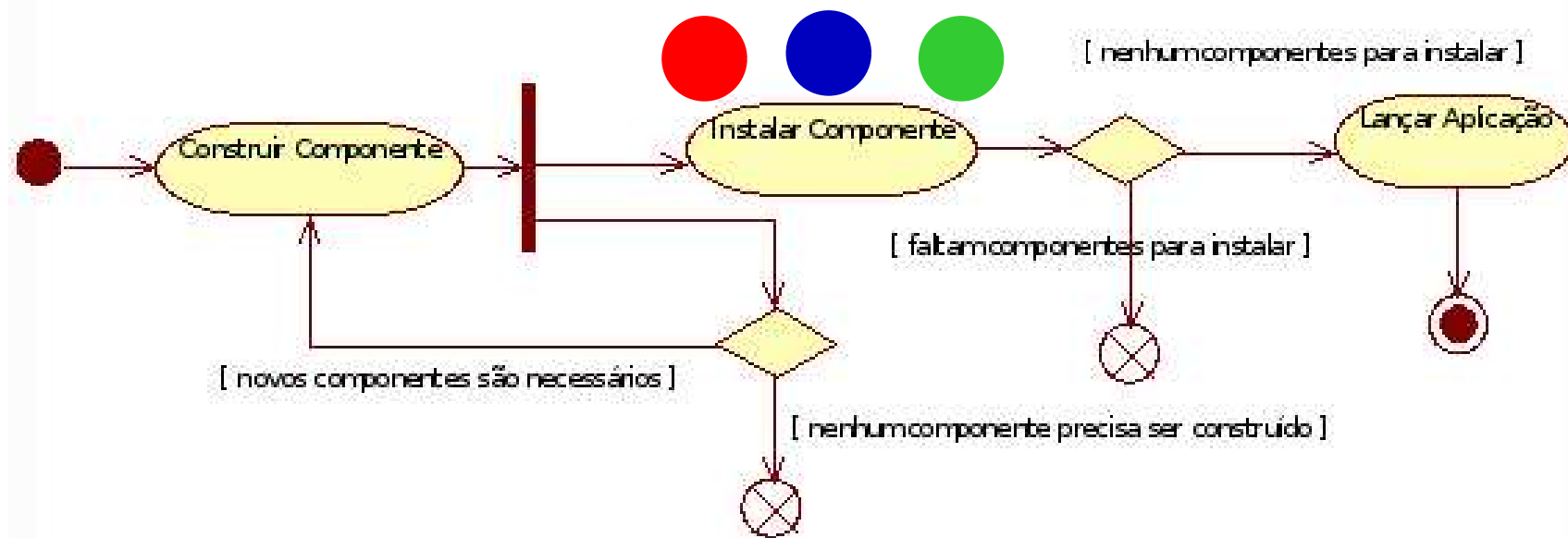


# Fluxos Terminando

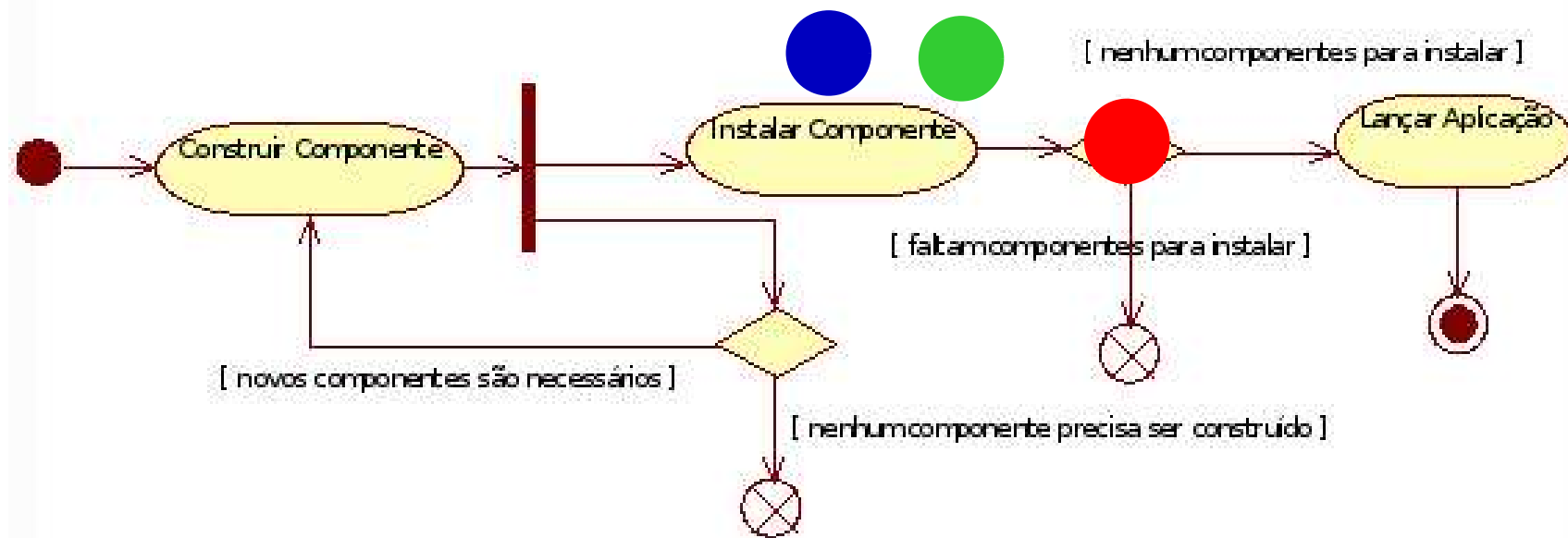




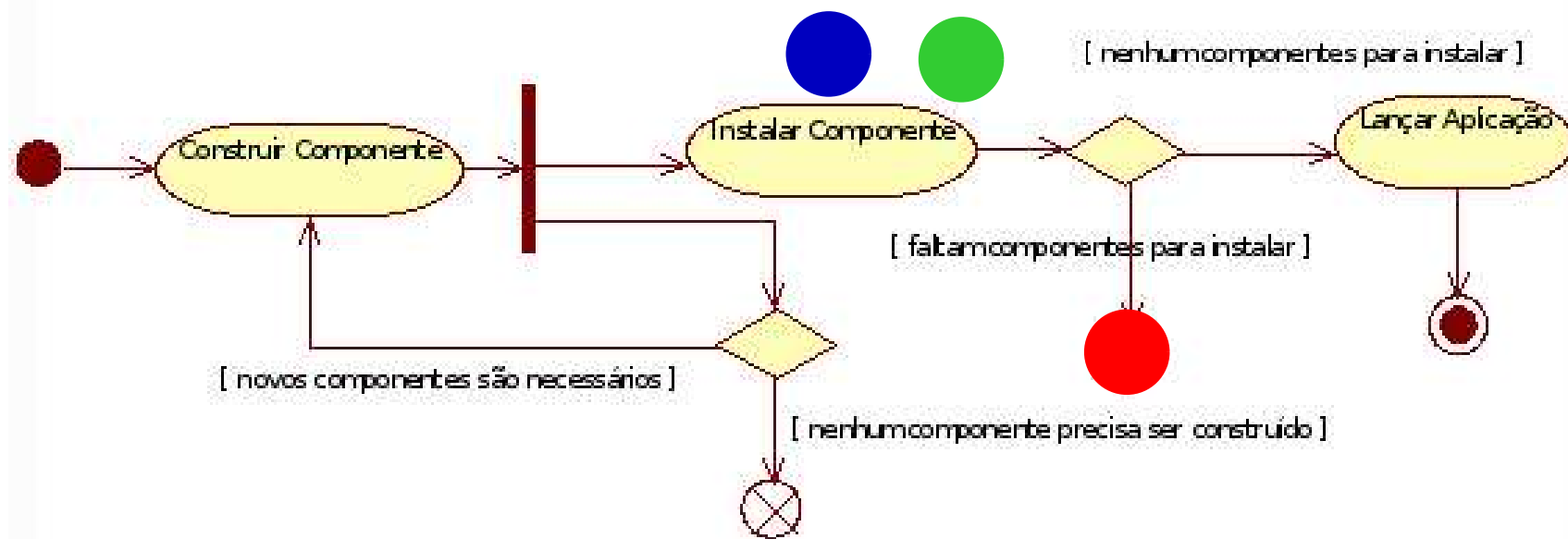
# Fluxos Terminando



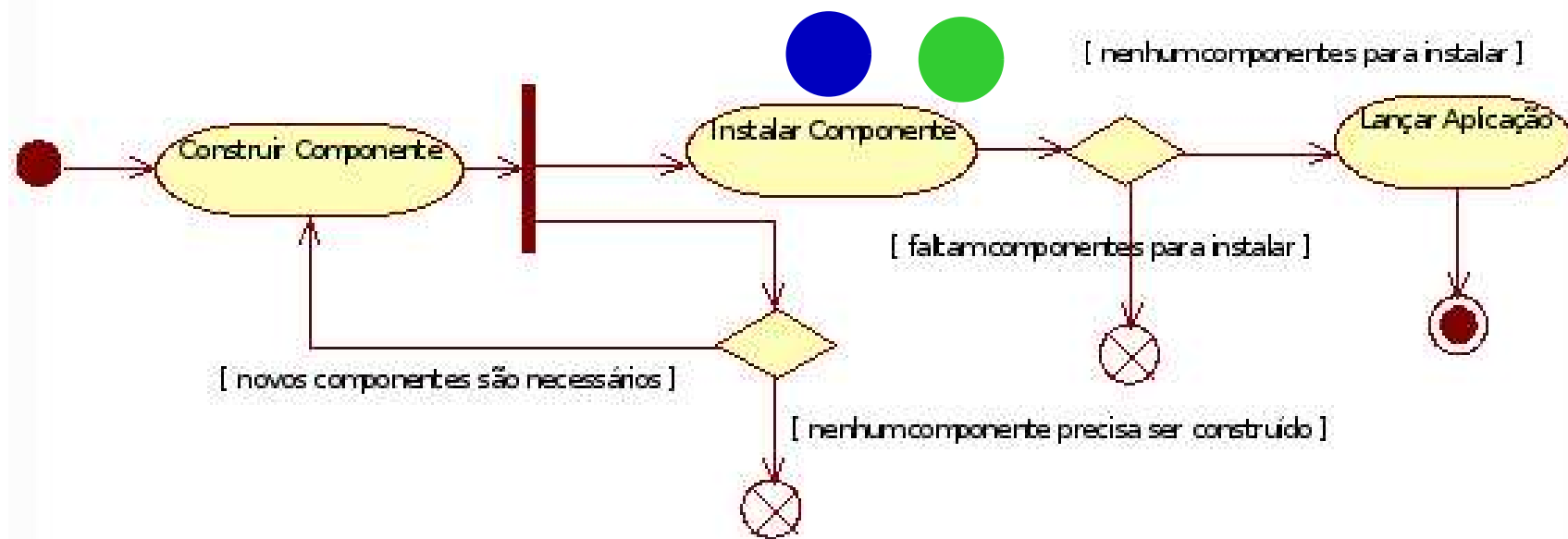
# Fluxos Terminando



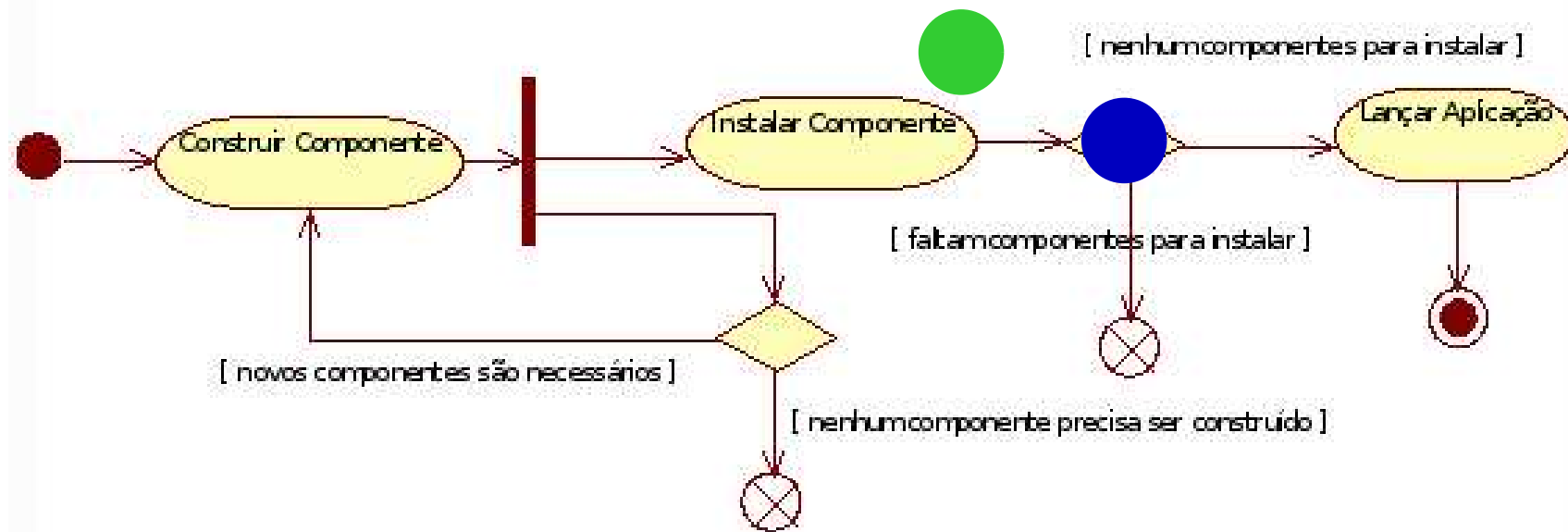
# Fluxos Terminando



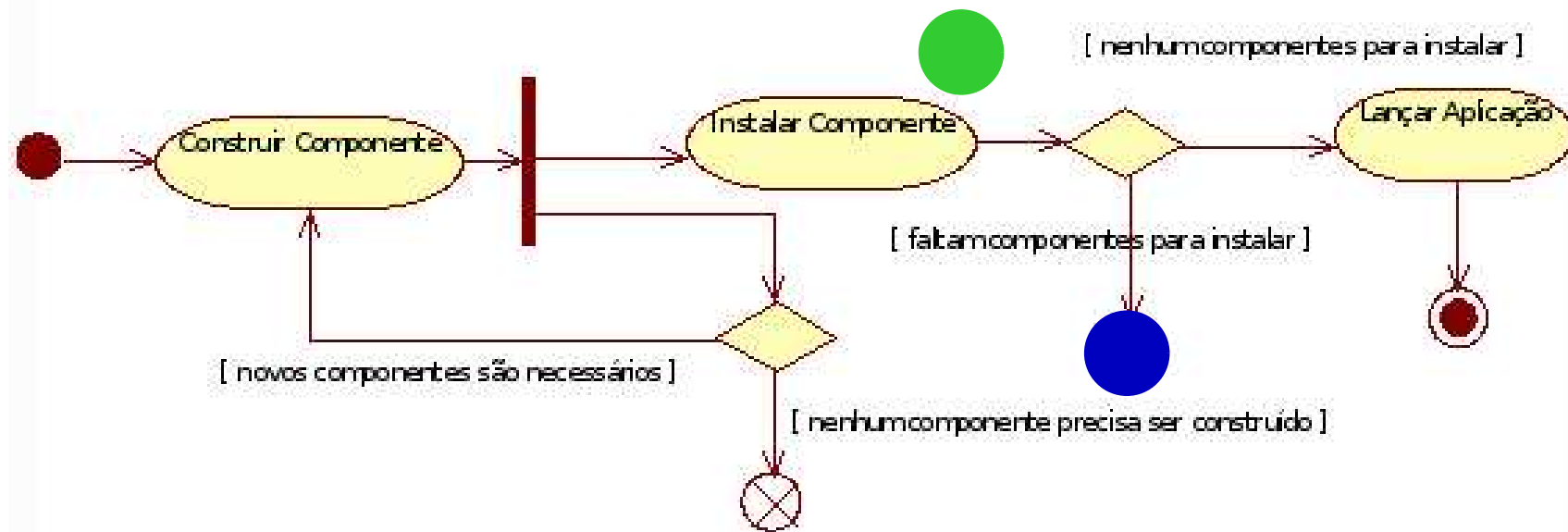
# Fluxos Terminando



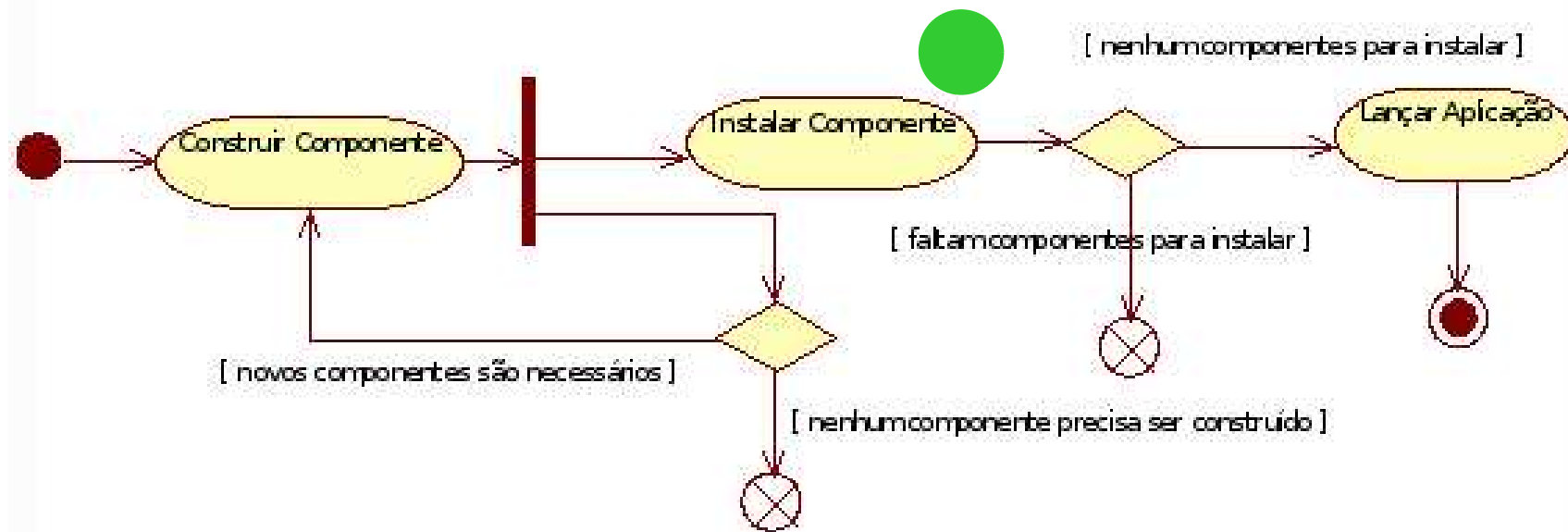
# Fluxos Terminando



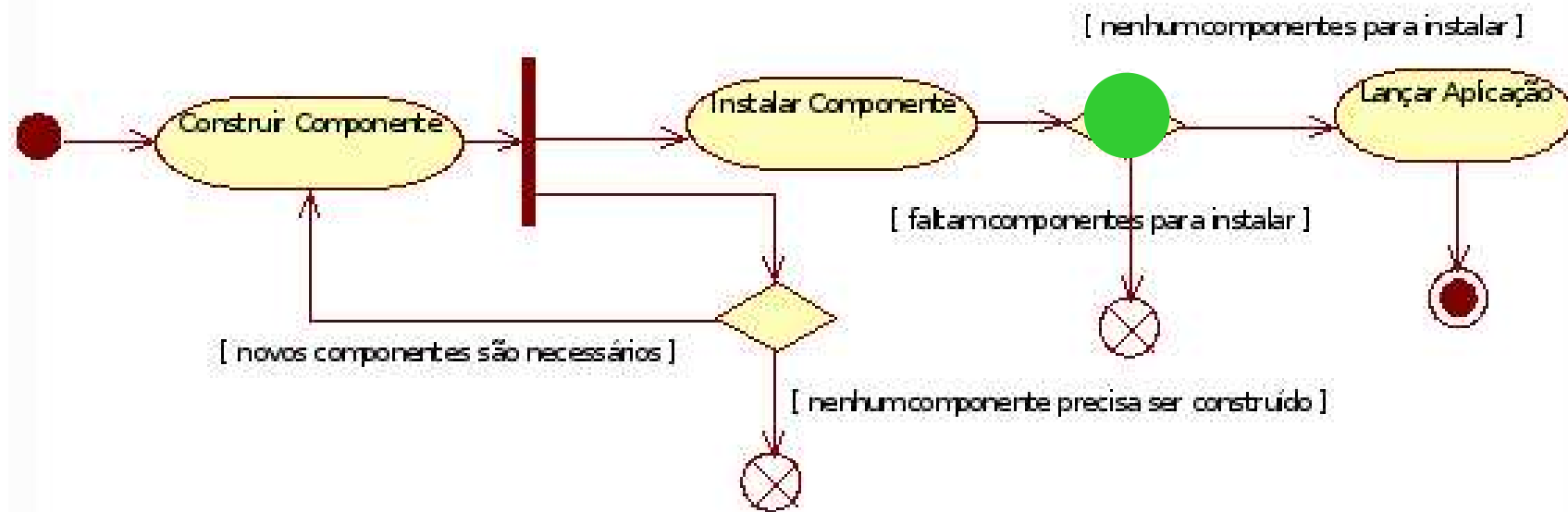
# Fluxos Terminando



# Fluxos Terminando

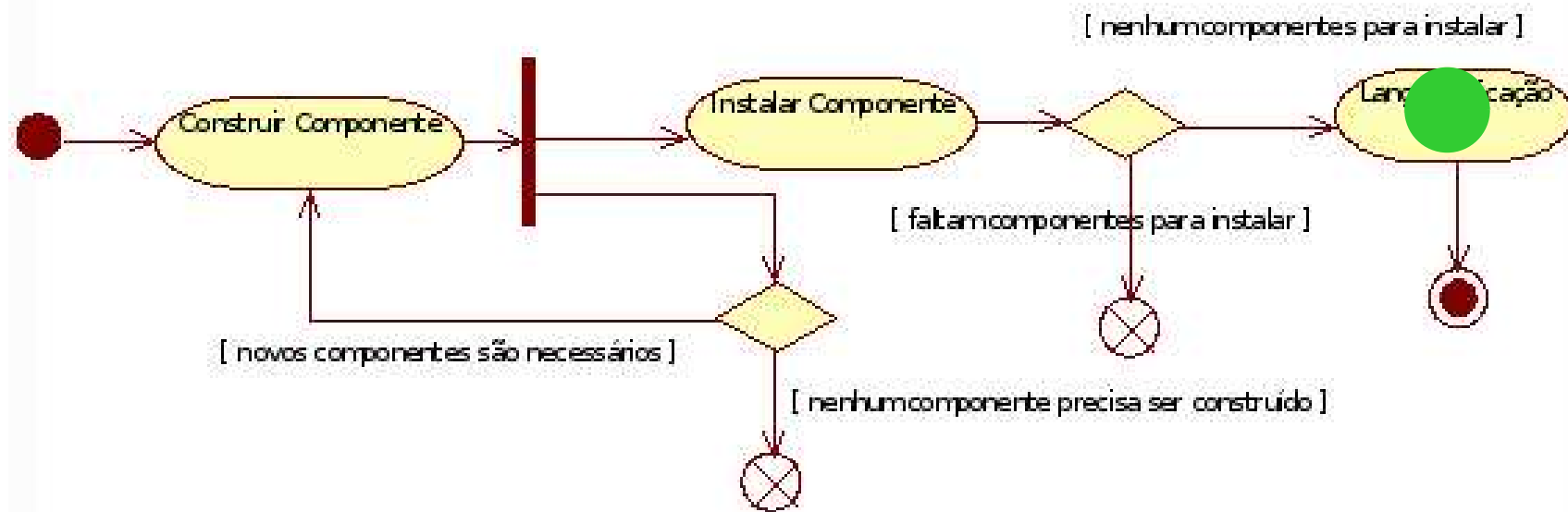


# Fluxos Terminando

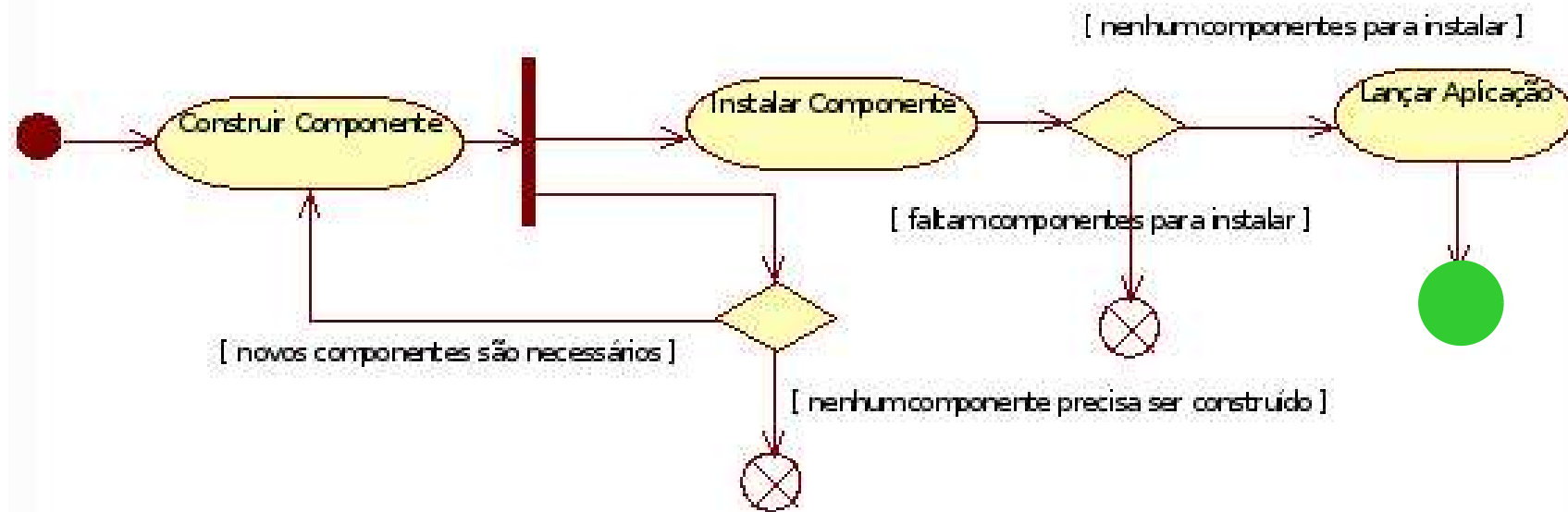




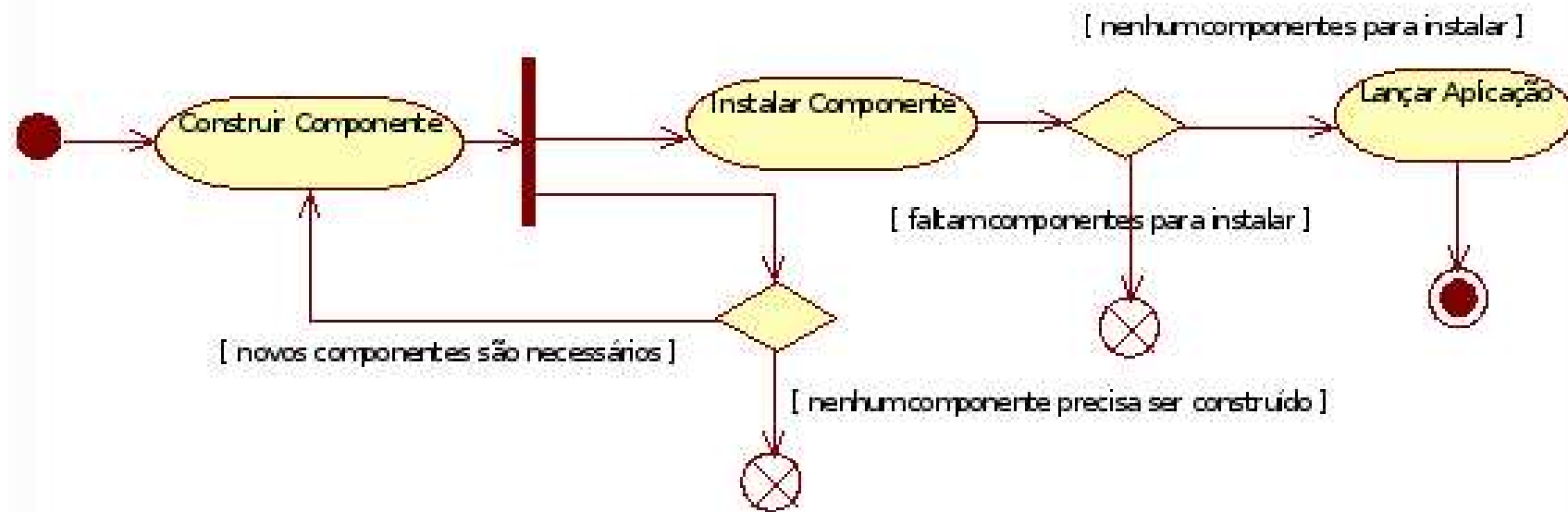
# Fluxos Terminando



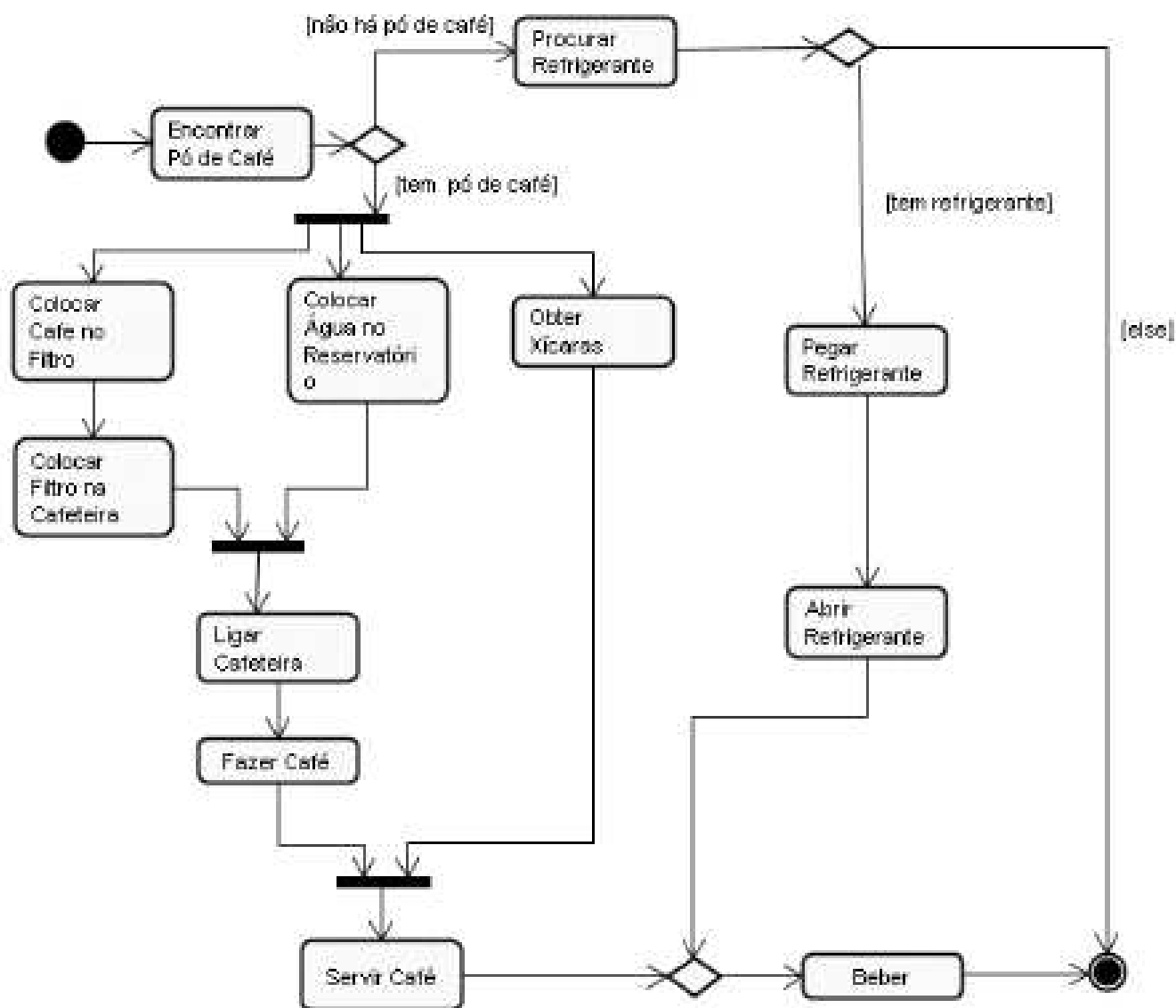
# Fluxos Terminando



# Fluxos Terminando

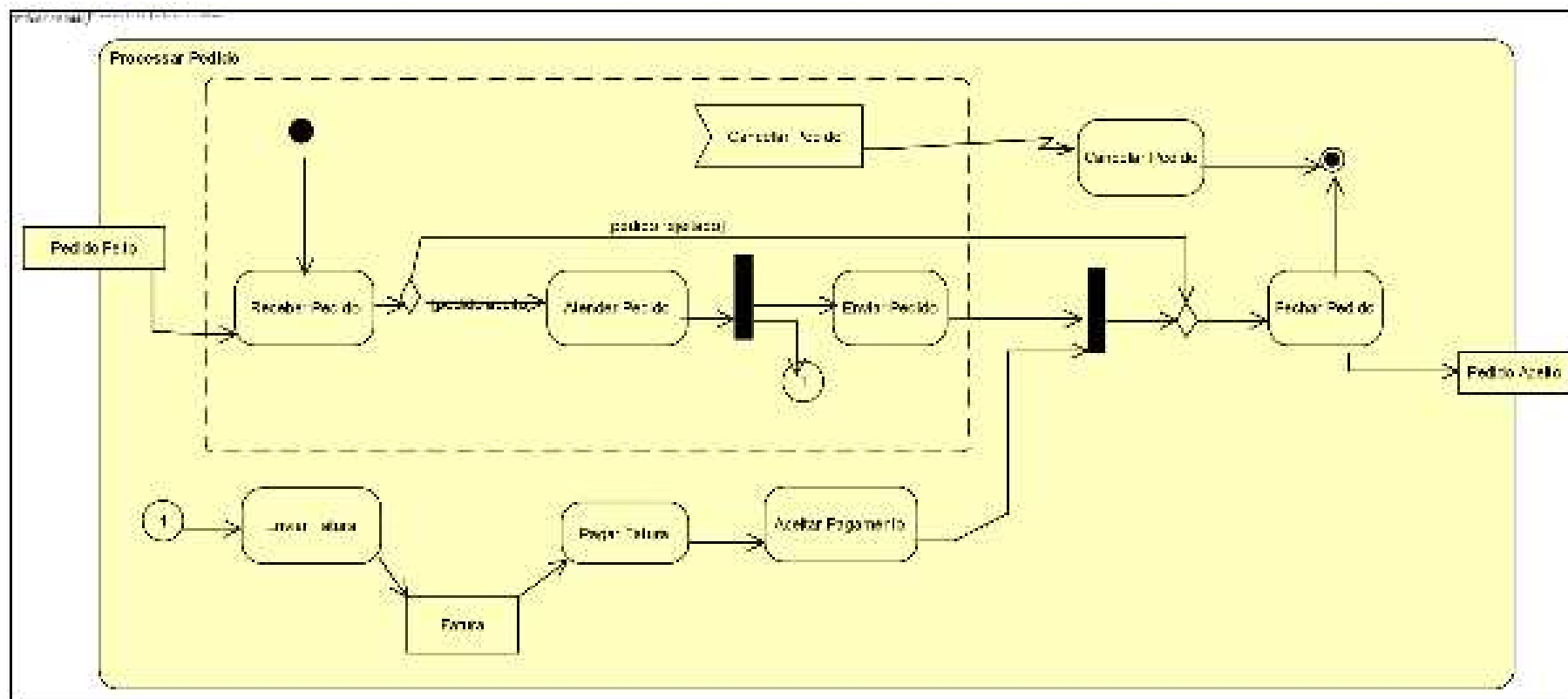


# Outro Exemplo



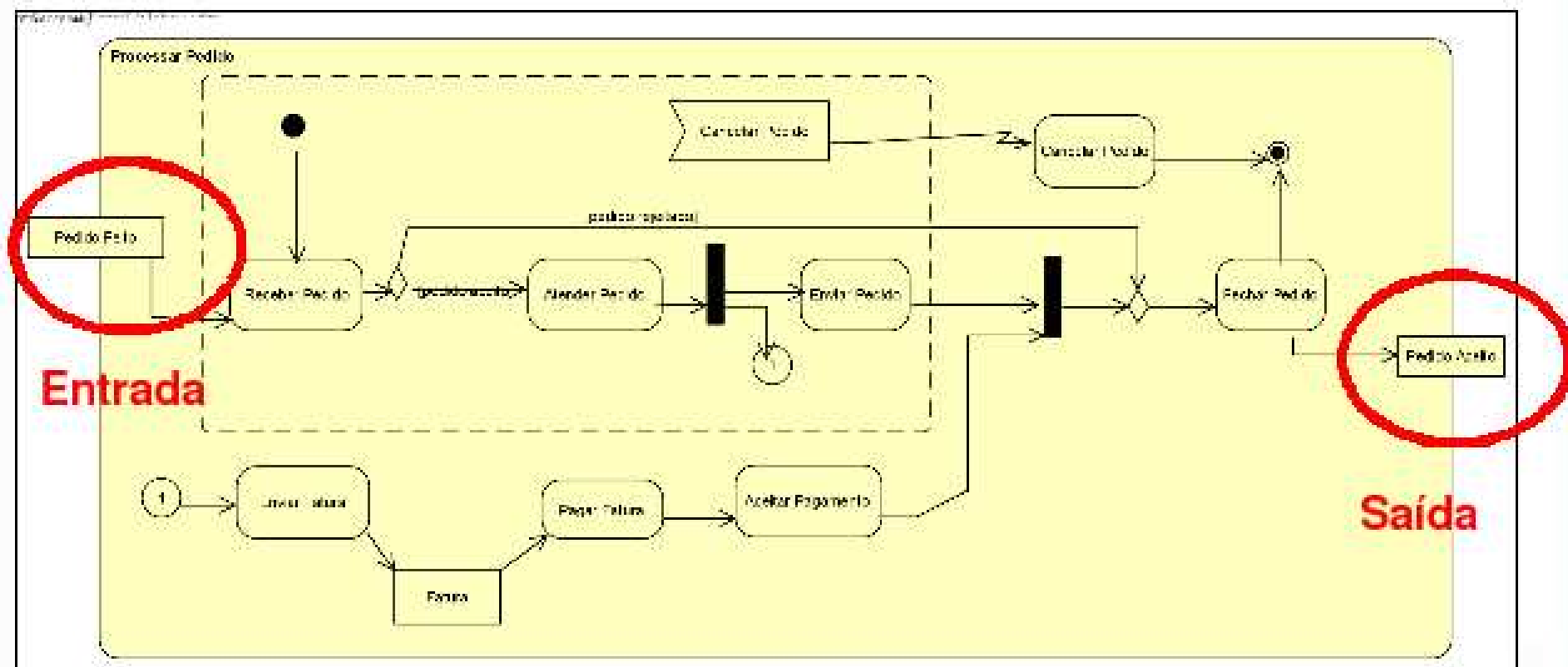
# Atividade

➡ Descreve uma atividade específica em um DA



# Parâmetros de Entrada e Saída

➡ Descrevem dados e objetos que são parâmetros de entrada e saída



# Partições

- ➡ Qualquer diagrama pode ser construído dentro de um espaço que modele alguma partição dessas atividades.
- ➡ Essas partições representam alguma divisão do sistema

# Partições Comuns

- ➡ Atores que realizam as atividades
  - ▬ Departamentos
  - ▬ Empresas
- ➡ Localidades
- ➡ Etc...



# Raias (Swimlanes)

- ➡ Normalmente o que se faz é utilizar colunas (ou linhas) para modelar os agentes que realizam a atividade específica, dando a impressão de "raias" no diagrama (swimlanes).
  - ▬ Podem ser usadas várias partições, de forma hierárquica ou multi-dimensional

# Partição Simples

Vendas	Produção	Estoque

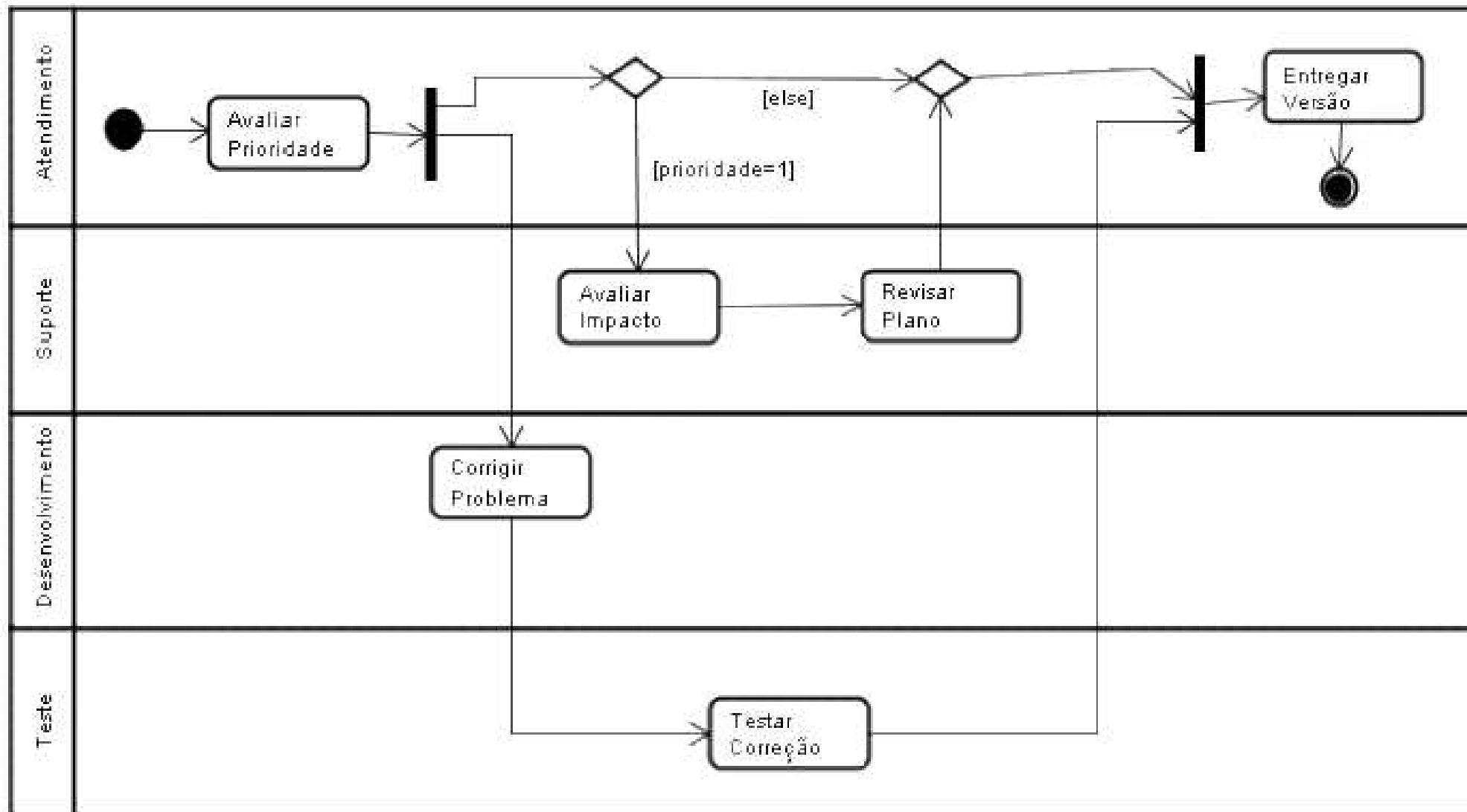
# Partição Multi-Dimensional

	Vendas	Compras
Rio		
São Paulo		

# Partição Hierárquica

Empresa		Externo
Vendas	Produção	Cliente

# Exemplo com Raias



# Envio de Sinais

SignalSendAction

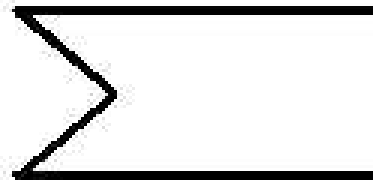
- ➡ Especialização de Ação
- ➡ Uma ação com dois parâmetros:
  - Sinal
  - Nome do Receptor
- ➡ Não representa o sinal, mas o envio do sinal

# Recepção de Evento

## ➡ Especialização de Ação



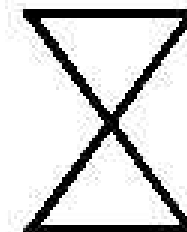
UML 1.5



UML 2.0

## ➡ Indicam que a ação recebe um evento

## ➡ Se é um evento temporal, notação diferenciada (ampulheta)



# Recepção de Evento

## ➡ Especialização de Ação



UML 1.5

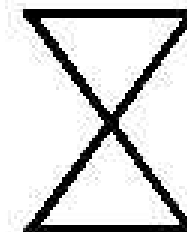


UML 2.0

Ferramentas  
utilizadas  
ainda não  
suportam esses  
símbolos

## ➡ Indicam que a ação recebe um evento

## ➡ Se é um evento temporal, notação diferenciada (ampulheta)





# Recepção de Evento

## ➡ Especialização de Ação



UML 1.5

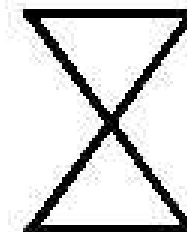


UML 2.0

Ferramentas  
utilizadas  
ainda não  
suportam esses  
símbolos

## ➡ Indicam que a ação recebe um evento

## ➡ Se é um evento temporal, notação diferenciada (ampulheta)

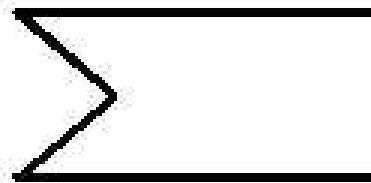


# Recepção de Evento

## ➡ Especialização de Ação



UML 1.5

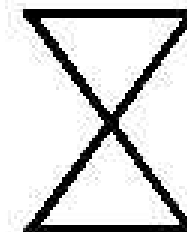


UML 2.0

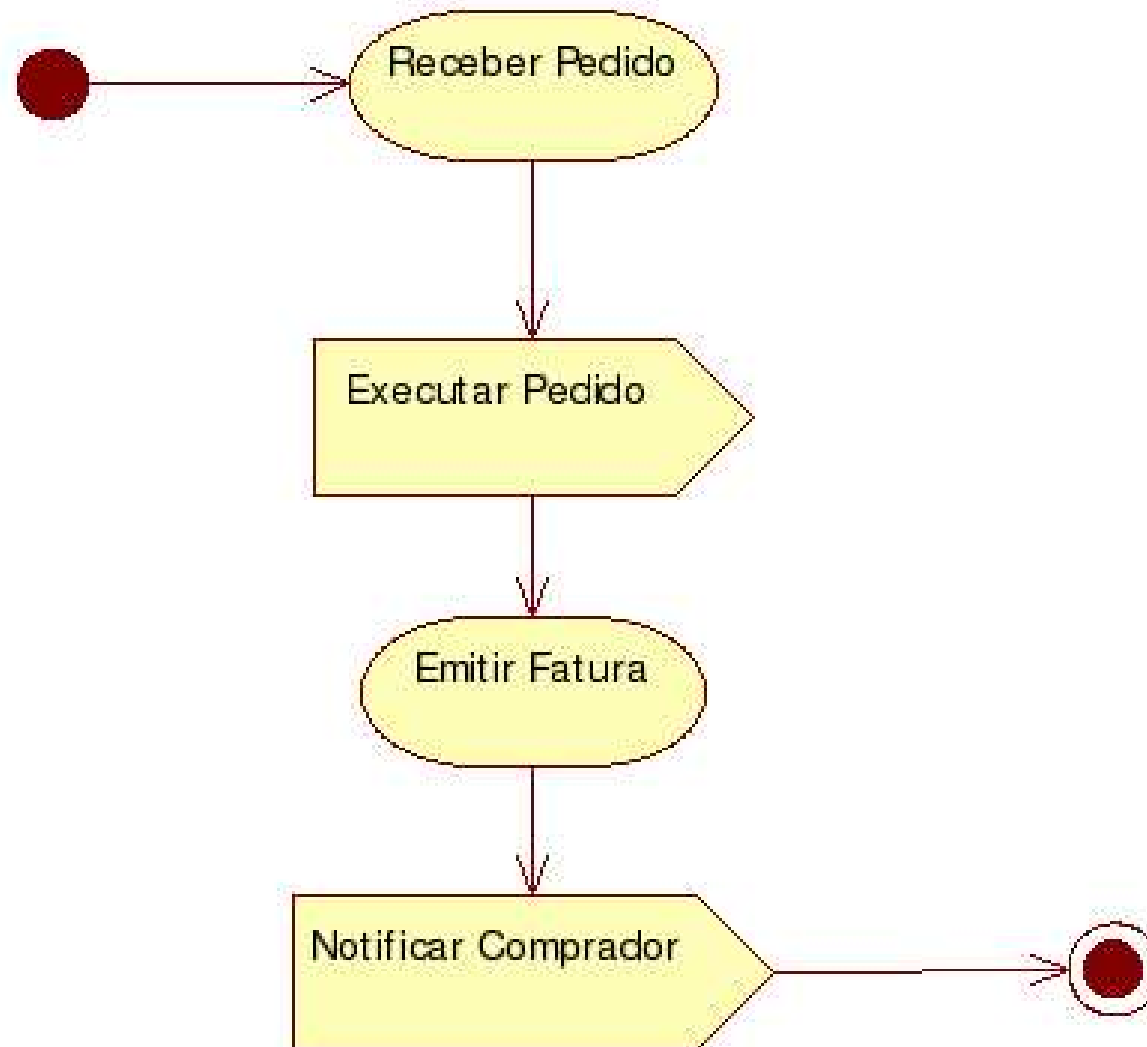
Ferramentas  
utilizadas  
ainda não  
suportam esses  
símbolos

## ➡ Indicam que a ação recebe um evento

## ➡ Se é um evento temporal, notação diferenciada (ampulheta)

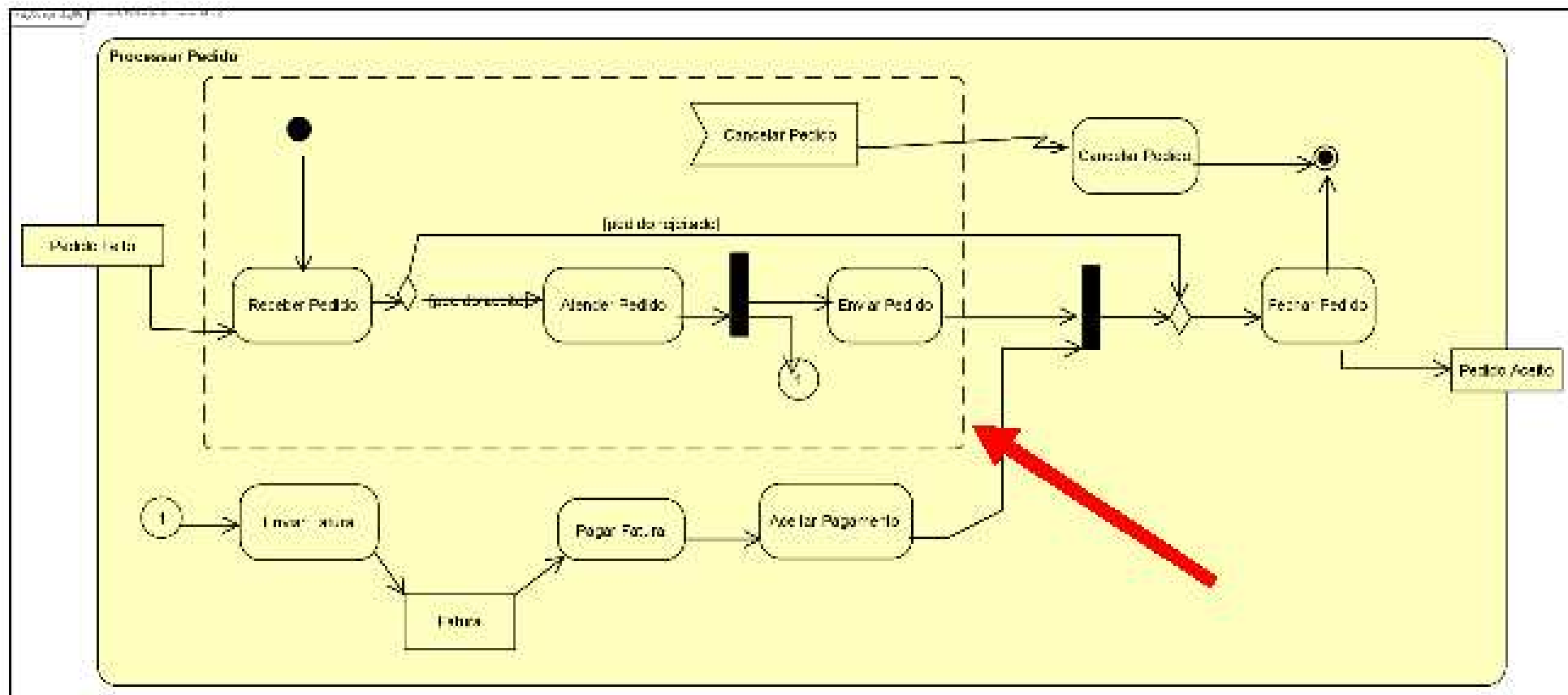


# Exemplo Enviando Sinais



# Área Interrompível (1/2)

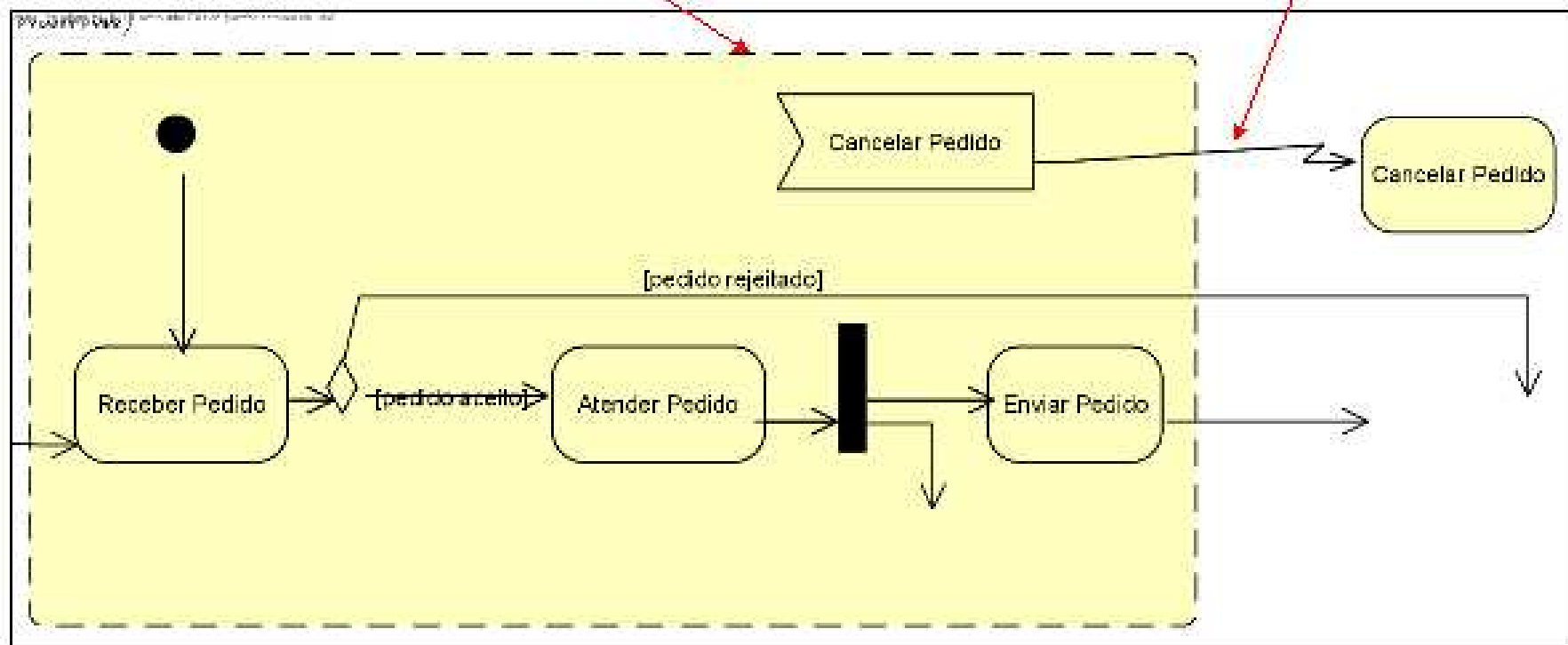
➡ Uma região que pode sofrer uma interrupção  
(por exemplo, exceção)



# Área Interrompível (2/2)

Recepção de Sinal

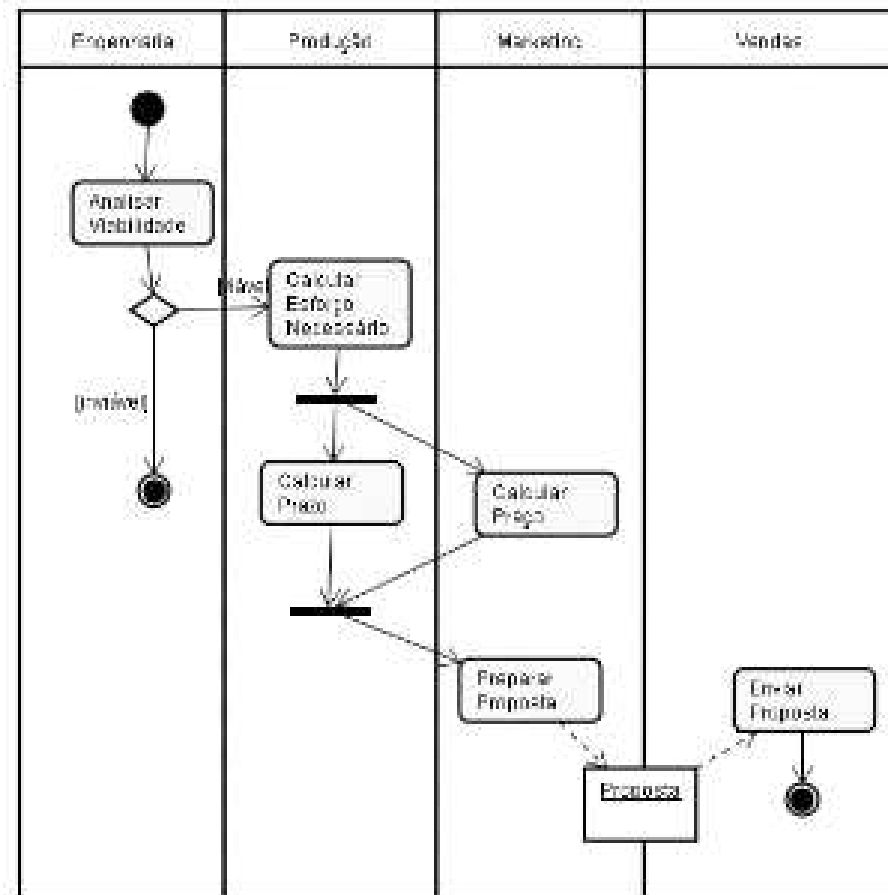
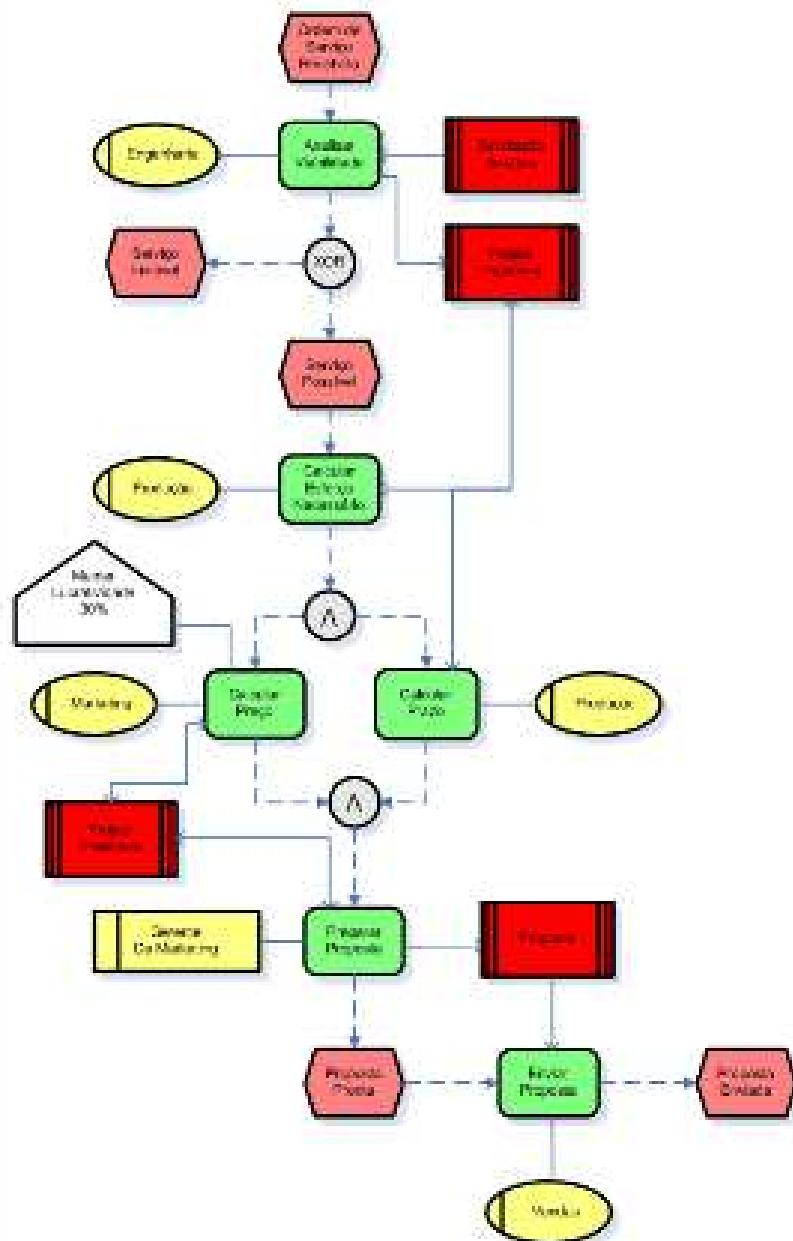
Tratamento de Exceção



# Modelando o Negócio com DA

- ➡ O Diagrama de Atividades pode ser usado em qualquer modelo que descreve um fluxo de ações.
  - ▢ No lugar do EPC
  - ▢ Em um nível de abstração mais detalhado do que o EPC, usando um DA para cada Processo do EPC
- ➡ EPCs também seguem uma semântica de fluxo de tokens! (Rede de Petri).

# EPC e DA



Esses dois diagramas mostram algumas semelhanças. As informações que faltam ao DA podem ser colocadas como objetos e comentários, porém sem alcançar a mesma qualidade do EPC. Para modelar negócios a OMG propõe uma linguagem própria (BPMN)

# Nível de abstração do DA

- ➡ O DA pode trabalhar em vários níveis de abstração, desde níveis ligados ao negócio até a descrição de um comportamento específico de um método.
- ➡ A nova versão do DA foi criada especialmente para suportar a criação automática de programas a partir de modelos (MDA - Model Driven Architecture)



## DA 2.0



Poucas ferramentas estão totalmente adaptadas, e as que estão adaptadas ainda apresentam algumas pequenas não conformidades

- Em especial, as ferramentas livres ainda estão fortemente ligadas a versão 1.5, que utilizava a semântica de máquina de estados

# DA 2.0 é mais poderosa?

➡ Na verdade, os diagramas de atividade da versão 2.0 trazem maior diversidade a UML, pois a notação de máquina de estado já existia.

➡ Novas características aumentam o poder como ferramenta de modelagem

▢ Mas não aumentam o poder computacional "de facto".

## Aula 18

### Professor:

Geraldo Xexéo  
DCC/IM/UFRJ  
PESC/COPPE/UFRJ

### Conteúdo:

**FIM: Diagramas de  
Atividade: UML 2.0**