Diagramas de Actividade

Diagramas de actividade

- Modelam aspectos dinâmicos dum sistema.
- Também modelam o fluxo de controlo de uma operação, classe, sistema, subsistema.
- Podem ser utilizados para descrever cenários de *use cases*.
- Enfatizam o fluxo de controlo através das mensagens entre objectos.
- Descrevem um processo consistindo em:
  - acções e actividades;
  - fluxo de controlo;
  - objectos de entrada e saída;
  - decisões;
  - Concorrência,
  -

Conceitos

- Uma actividade corresponde à execução de um conjunto de acções.
- Uma acção é uma computação atómica.
- Um diagrama de actividade contém:
  - Estados de actividade
  - Estados de acção
  - Transições
  - Objectos
  - Decisões (*branching*)
  - Disjunção (*fork*) e junção (*join*)
  - Pistas (*swimlanes*)

Estados de acção

- São estados do sistema representando a execução de uma acção:
  - e.g., criar ou destruir um objecto, executar uma operação num objecto
- Não podem ser decompostos, são atómicos e não podem ser interrompidos.
- O tempo de execução de um estado de acção é insignificante.

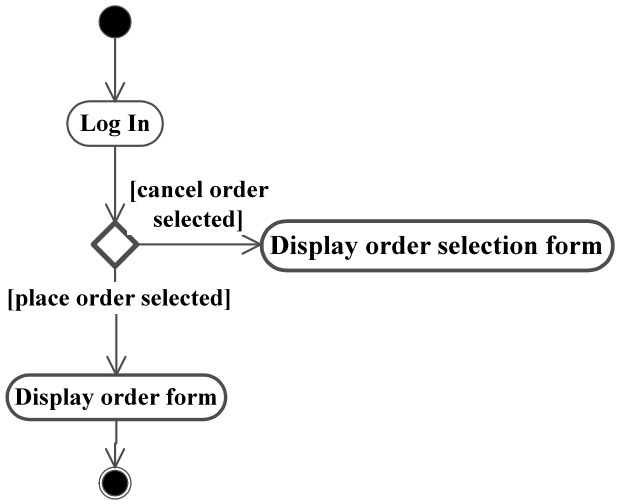
Estados de actividade

- Podem ser decompostos; onde se usa actividade pode usar-se outro diagrama de actividades.
- Não sendo atómicos, podem ser interrompidos (têm tempo (significativo) de execução).
- (Um estado de acção é um estado de actividade que não pode ser decomposto.)
- A notação é a mesma do estado de acção

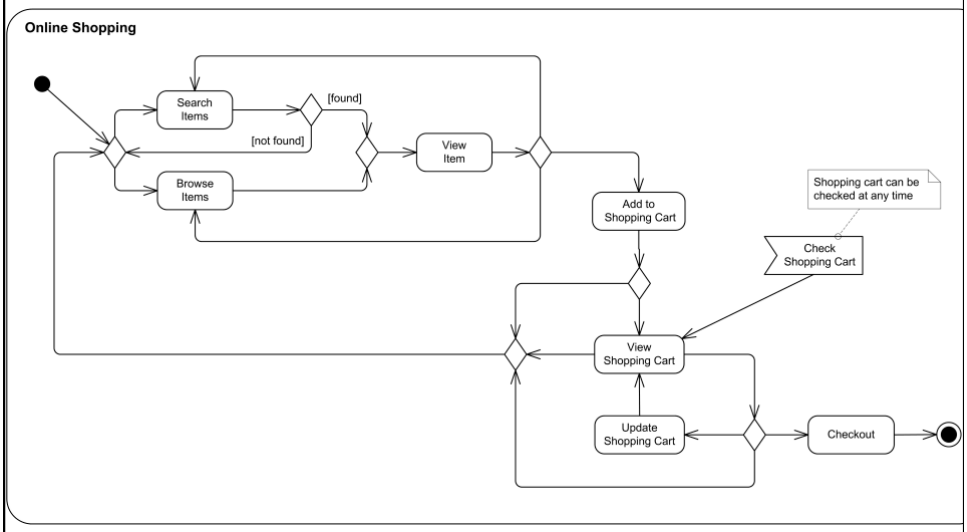
Transições e decisões

- **Transições**: quando a acção ou actividade de um estado se completa, o fluxo de controlo passa imediatamente p/ o próximo estado de acção ou actividade.
- **Decisões** (*branching*): expressões *booleanas* que especificam passos alternativos.
- Um decisão consiste numa transição de entrada e duas ou mais de saída.
- **Guardas**: condições.
- Nas transições de saída as *guardas* não se devem sobrepor, mas devem cobrir todas as possibilidades.

Transições e decisões



Decision e *merge*

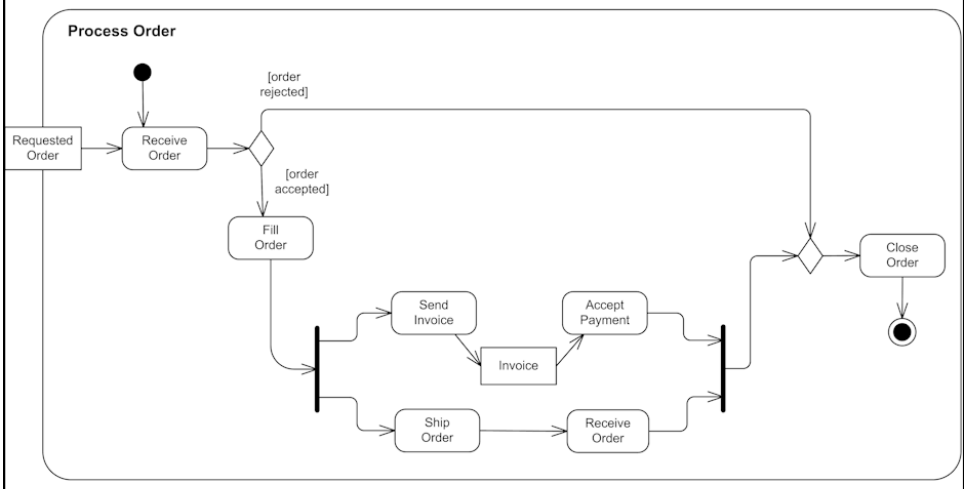


[http://www.uml-diagrams.org/notation/activity-edge-connector.png]

Disjunção (*fork*) e junção (*join*)

- Uma **disjunção/difusão** representa a separação de um fluxo de controlo em dois ou mais fluxos de controlo.
  - Pode ter uma transição de entrada e duas ou mais transições de saída.
- Uma **junção** representa a sincronização de dois ou mais fluxos de controlo.
  - Pode ter duas ou mais transições de entrada e uma de saída;
  - Os fluxos concorrentes sincronizam-se assim: espera-se que todos os fluxos de entrada cheguem ao ponto de junção prosseguindo com apenas um fluxo depois da junção.

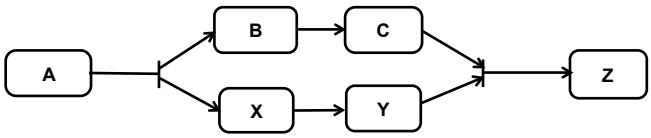
Exemplo de *fork* e *join*



[http://www.uml-diagrams.org/notation/activity-edge-connector.png]

Modelo de concorrência

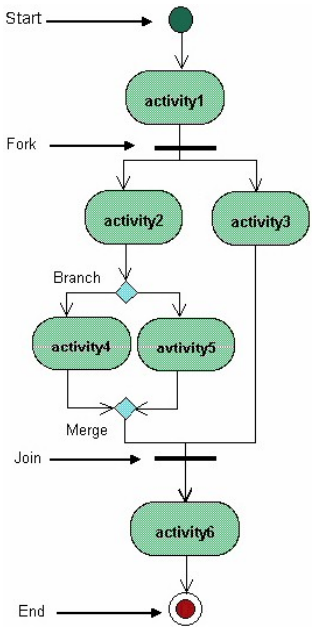
- *Fully independent concurrent streams*



Trace: A, {(B,C) || (X,Y)}, Z

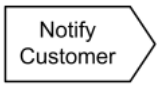
- Onde "||" é o operador de concorrência (usado nas linguagens de especificação algébricas, por exemplo)

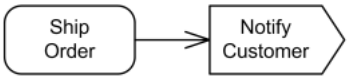
Até aqui…



Sinais (1)

send signal: acção de invocação que cria um sinal e o envia ao objecto destino, onde pode originar a execução de uma actividade ou a transição de um estado

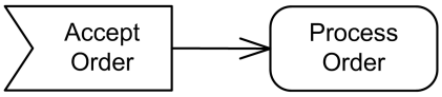




Depois de **Ship Order** cria-se o sinal **Notify Customer**

Sinais (2)

accept signal: é uma acção que espera que o evento especificado ocorra (normalmente assíncrono)



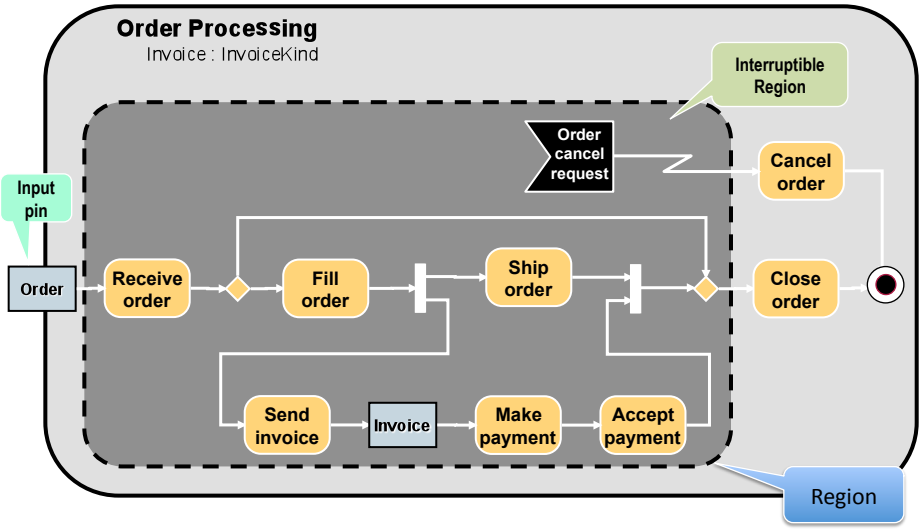
A aceitação do evento **Accept Order** causa a invocação da acção **Process Order**

Sinais (3)

send/accept signal

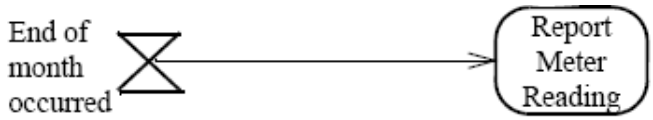
A actividade **Process Order** origina o envio do sinal **Request Payment**. A actividade **Ship Order** espera para receber o sinal **Payment Confirmed**

Exemplo, com regiões



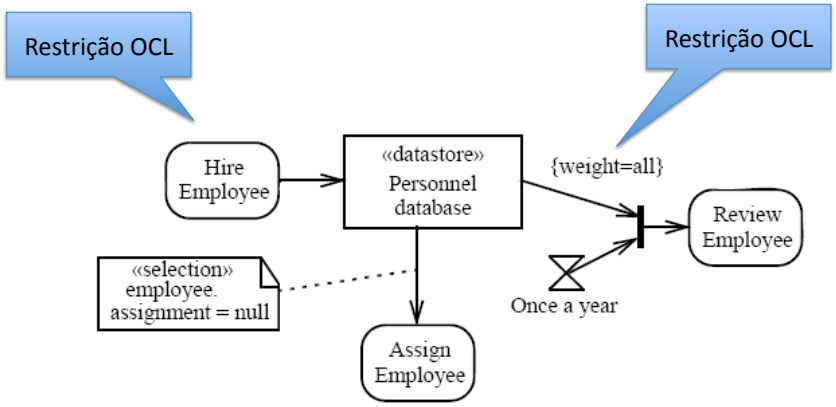
Eventos temporais (repetitivos)

wait time action



Ao fim do mês (**End of month occurred**) activa-se a actividade **Report Meter Reading**

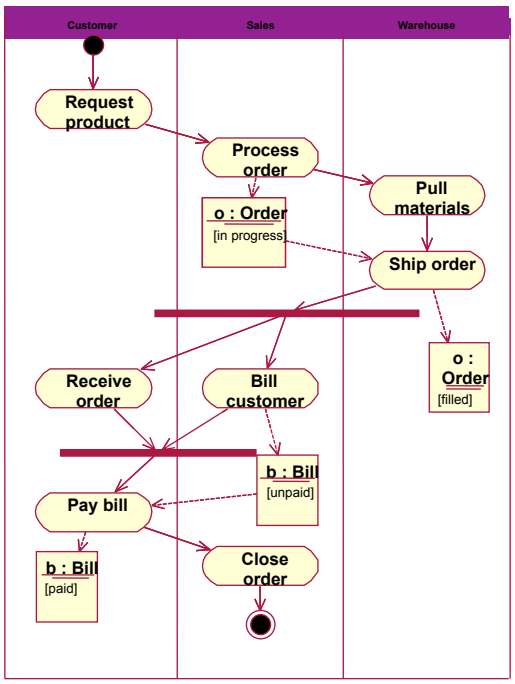
Exemplo

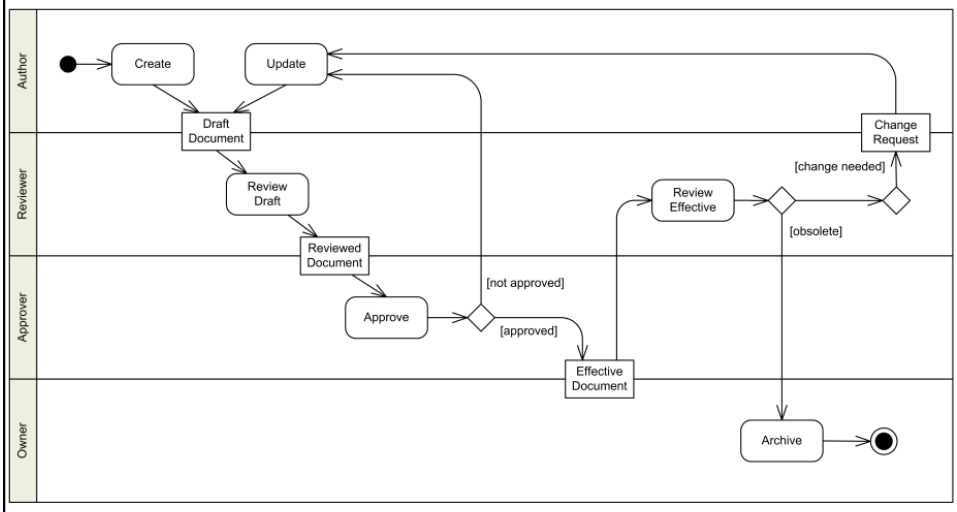


Anualmente (**Once a year**) activa-se a actividade **Review Employee**, para os empregados contratados

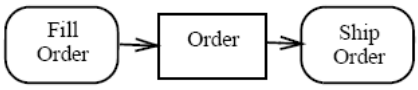
Pistas *(swimlanes)*

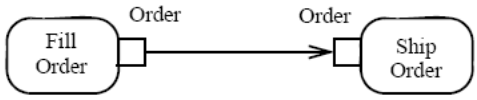
- Utilizado para particionar os estados de actividade em grupos quando se modela *workflows*, *business process*, processos de *software;*
- Cada *swimlane* pode ser implementada por uma ou mais classes;
- Transições podem partir de um *swimlane* para outro mas uma actividade pertence a um só *swimlane.*





Object flow

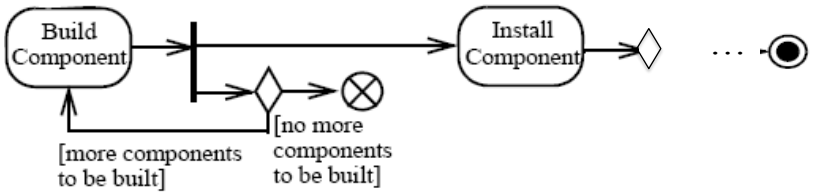




- Objectos podem estar envolvidos no fluxo de controlo associado com um diagrama de actividade
  - Os fluxos de objecto são dependências que criam, removem ou alteram um objecto;
  - Pode-se mostrar o estado ou os valores dos atributos do objecto.

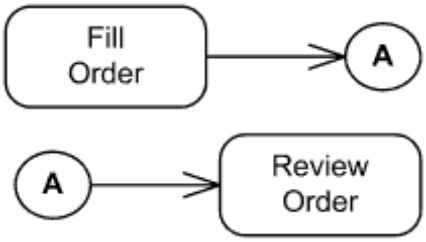
Flow final + activity final

- Activity final: é um nó de controlo que pára **todos** os fluxos no diagrama. (Pode haver mais que um num diagrama.)
- Final flow: é um nó de controlo que termina **um** fluxo. Não tem qualquer efeito noutros fluxos no diagrama

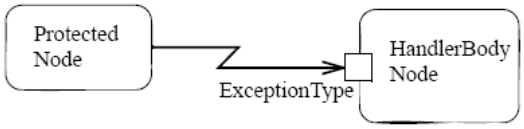


Conector

- Usado para evitar uma transição comprida
- Não afecta o modelo



Conceitos avançados: exception handler



- *If an exception occurs during the execution of an action, the execution of the action is abandoned and no regular output is generated by this action. If the action has an exception handler, it receives the exception object as a token. If the action has no exception handler, the exception propagates to the enclosing node and so on until it is caught by one of them. If an exception propagates out of a nested node (action, structured activity node, or activity), all tokens in the nested node are terminated. The data describing an exception is represented as an object of any class.*

[http://www.uml-diagrams.org/activity-diagrams.html#partition]

E quando definimos diagramas de actividade separados para cenários alternativos?

- *Extension points* no diagrama de casos de uso
- *Pattern Specifications* para diagramas de actividade?

Pattern Specifications (1)

- Permitem a formalização da reutilização de modelos
- Baseado em UML
- Um PS descreve um padrão de estrutura ou comportamento
  - Definido sobre os papéis (*roles*) desempenhados peloas participantes nesse papel
  - *Roles* são precedidos por uma barra vertical "|"

Pattern Specifications (2)

- Um PS pode ser instanciado atribuindo elementos de modelo concretos para desempenhar os papéis (*roles*) identificados
- Um *role* é uma especialização de uma metaclasse UML restringida por propriedades adicionais que qualquer elemento a desempenhar o *role* deve possuir
  - Metaclasse: uma classe cujas instâncias são classes