UML – Diagramas de Comportamento e Testes

Classes / Objectos de Desenho – descrição dos estados

- Objectos de Desenho
  - são instâncias das Classes de Desenho
- O ciclo de vida de alguns Objectos de Desenho
  - é governado por estados – *state controlled*, ou seja,
    - o objecto passa, ao longo da vida, por um conjunto de estados
    - ... em cada instante o objecto está num desses estados
    - ao receber uma mensagem, o objecto tem um comportamento que é
    - ... determinado pelo estado em que o objecto está nesse instante
- O ciclo de vida dos Objectos *state controlled*, pode ser descrito,
  - através de um "diagrama de transição de estados"
  - ... também designado por,
    - "diagrama de estados" – *statechart*
    - "máquina de estados"

Diagramas de Estados

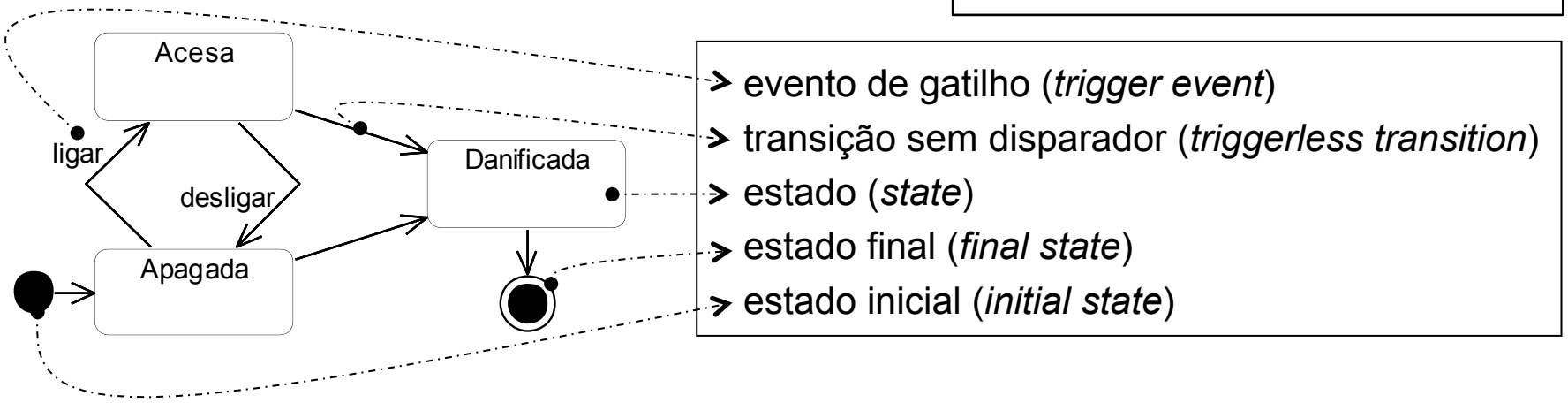
- Os Diagramas de Estados, tal como os Diagramas de Interacção
  - capturam a comportamento dinâmico do sistema
- Os Diagramas de Interacção, usam-se para modelar
  - uma sociedade de objectos que opera em conjunto
- Os Diagramas de Estados, usam-se para modelar
  - a evolução de um objecto individual, através de um conjunto de estados,
  - como resposta a eventos e à passagem do tempo
- Um Diagrama de Estados
  - pode caracterizar o ciclo de vida, das instâncias de,
    - uma classe
    - um caso de utilização (para modelar os seus cenários)
    - todo um sistema / subsistema

Diagramas de Estados (cont.)

- Um Diagrama de Estados, representa,
  - os possíveis estados de um objecto
  - as correspondentes transições entre estados
  - os eventos que fazem desencadear as transições
  - as operações (acções e actividades)
    - executadas dentro de um estado
    - ou durante uma transição

Diagrama de Estados de uma lâmpada:

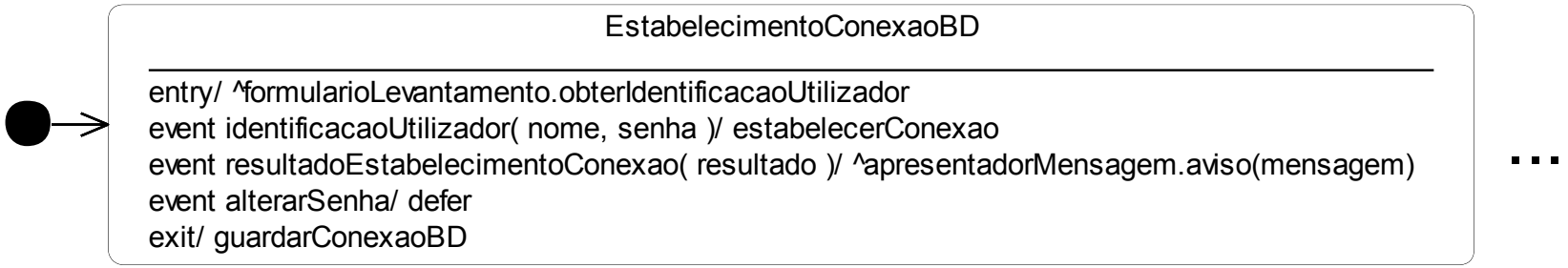
- Evolui entre os estados "acesa" e "apagada" conforme se liga e desliga um interruptor.
- Quando um dos estados "acesa" ou "apagada" termina a sua actividade normalmente (sem que essa terminação seja disparada pelos eventos "ligar" ou "desligar") a lâmpada passa ao estado "danificada"



Estado

- Condição ou Situação do ciclo de vida, durante a qual o objecto,
  - satisfaz alguma condição, e / ou
  - executa alguma actividade, e / ou
  - espera por algum evento
- Os constituintes de um estado são
  - nome – um texto que o distingue dos restantes estados
    - um estado sem nome designa-se por "estado anónimo"
  - acções de entrada / saída – executadas ao entrar / abandonar o estado
  - transições internas – são tratadas sem originar mudança no estado
  - eventos diferidos – lista de eventos que não é tratada no estado actual
    - empilhados e deixados para tratamento pelo objecto noutro estado
  - sub-estados – estados hierarquicamente contidos noutro estado
    - sub-estados disjuntos – execução sequencial
    - sub-estados concorrentes – execução paralela

Estado (cont.)



`^o.e` ≡ evento `e` enviado ao objecto `o`
`e/a` ≡ evento `e` acciona acção `a`

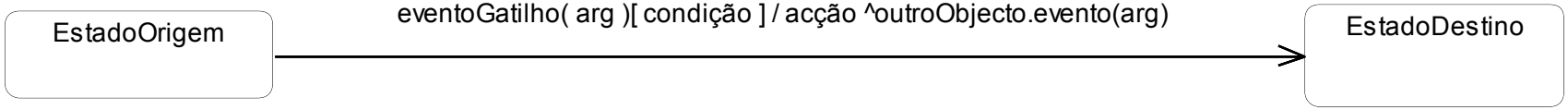
- acções de entrada / saída (*entry* / *exit* no diagrama)
  - acção independente da transição que conduziu / abandonou o estado
- transições internas (*event* no diagrama)
  - não acciona as acções de entrada / saída do estado
    - diferente de uma transição para o próprio estado
- eventos diferidos (acção *defer* no diagrama)
  - até que algum estado os trate
  - implementar implica ter uma pilha interna de eventos pendentes

Transição

- Relação entre dois Estados (origem e destino), indicando que,
  - um objecto no Estado origem, irá executar determinadas acções
  - e passar ao Estado destino, assim que,
    - o evento especificado ocorrer
    - e as condições especificadas forem satisfeitas
  - "transição disparou" – diz-se da mudança do Estado origem para destino
    - até que a transição dispare o objecto está no Estado origem
    - quando a transição dispara o objecto passa ao Estado destino
- Uma transição pode ter
  - múltiplos estados origem
    - representa a junção (*join*) de múltiplos caminhos concorrentes
  - múltiplos estados destino
    - representa a difusão (*fork*) para múltiplos caminhos concorrentes
  - ... no entanto este tipo de utilização não é muito comum

Transição (cont.)

- Uma transição pode ser descrita deste modo,



- Os constituintes de uma transição são
  - estado origem e destino que a transição interliga
  - evento de gatilho (*trigger event*)
    - quando recebido pelo objecto, no estado origem,
    - torna a transição "pronta a disparar", caso a condição seja satisfeita
  - condição (*guard condition*) – avaliada na recepção do evento de gatilho
    - condição satisfeita – transição é eleita para disparar
    - condição não satisfeita – transição não dispara; não existindo mais nenhuma transição que possa ser disparada o evento é perdido
  - acção (*action*) – computação atómica (não pode ser interrompida)
    - actua directamente sobre o objecto dono da máquina de estados
    - actua indirectamente sobre outros objectos que lhe estejam visíveis

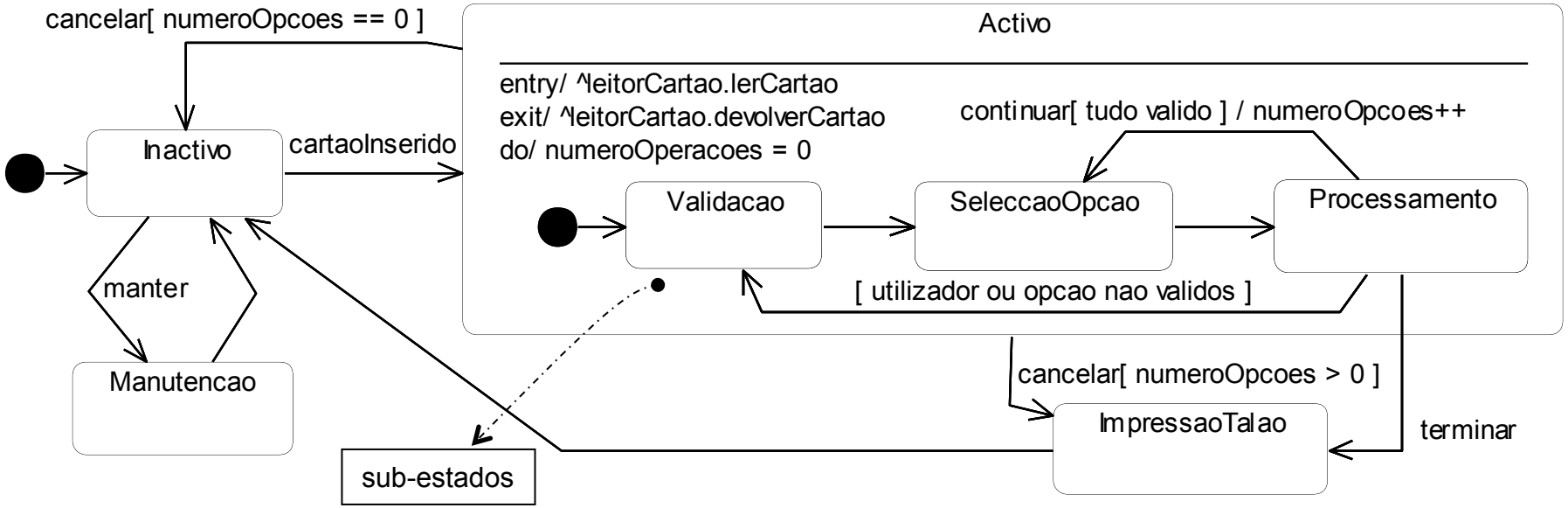
Estados e Transições – exemplo ATM

Um ATM pode estar num de três estado: inactivo (à espera de utilizador); activo (a tratar operações de um utilizador); manutenção (p.e. carregamento de dinheiro)

Quando está activo o ATM tem essencialmente o seguinte comportamento: valida o utilizador; este escolhe uma opção; a opção é processada; é impresso um talão.

É também útil que o ATM permita ao utilizador escolher uma nova opção depois de processada a anterior e antes da impressão do talão.

Em qualquer instante o cliente pode decidir cancelar a operação corrente – isso origina a impressão do talão com as operações até aí efectuadas e passagem do ATM a inactivo.

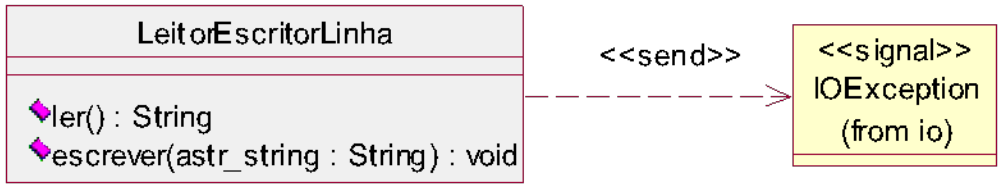


Evento

- Especificação de uma ocorrência localizada no tempo e no espaço
  - no contexto dos Diagramas de Estados, um evento representa,
    - a ocorrência de um estímulo que pode disparar uma transição
- Os eventos podem ser externos ou internos
  - externos – os que fluem entre o sistema e os seus actores
    - p.e. premir botão; sensor de impacto enviando interrupção
  - internos – os que fluem entre os objectos que residem no sistema
    - p.e. excepção resultante de divisão por zero
- Existem quatro tipos de eventos,
  - sinal (*signal*)
  - invocação (*call event*)
  - passagem de tempo (*time event*)
  - mudança de estado (*change in state event*)

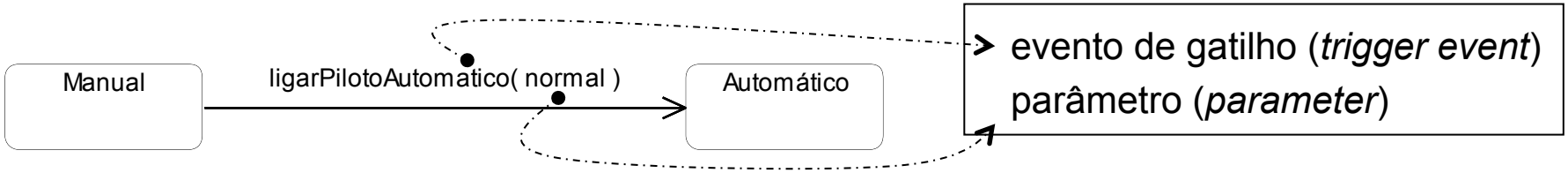
Evento – Sinal

- Sinal (*signal*) – representa um objecto
  - lançado/enviado (*thrown*) de modo assíncrono por um objecto
  - apanhado/recebido (*caught*) por outro
    - exemplo mais comum é o do mecanismo de excepções
- Um sinal pode ser lançado, como,
  - acção de uma transição num Diagrama de Estados
  - mensagem num Diagrama de Interacção
- A especificação comportamento de uma Classe ou Interface,
  - deve incorporar os sinais (e excepções) que cada operação pode enviar
  - através de uma Relação de Dependência estereotipada com <<send>>



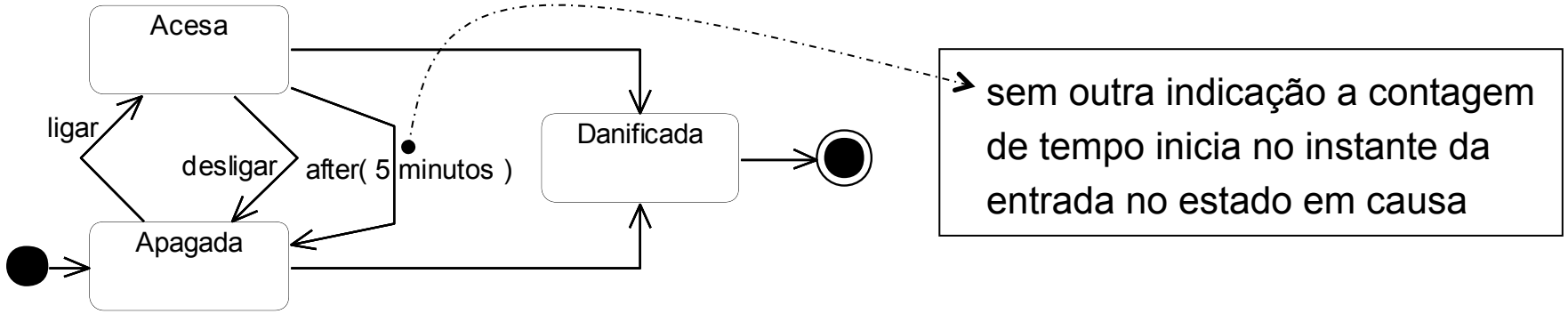
Evento de Invocação

- Evento de Invocação (*call event*) – lançamento de uma operação
  - é, em geral, síncrono (diferente do *signal* que é sempre assíncrono)
- Lançar operação – o objecto receptor tem uma máquina de estados
  - o controlo passa para o objecto receptor
  - no receptor é disparada uma transição
  - o receptor transita para um novo estado
  - o controlo regressa ao objecto que lançou a operação
- A modelação de um *signal* é igual à de um *call event*
  - a distinção está implícita no modelo
  - *signal* – tratado pela próprio Diagrama de Estados
  - *call event* – tratado por um método da classe



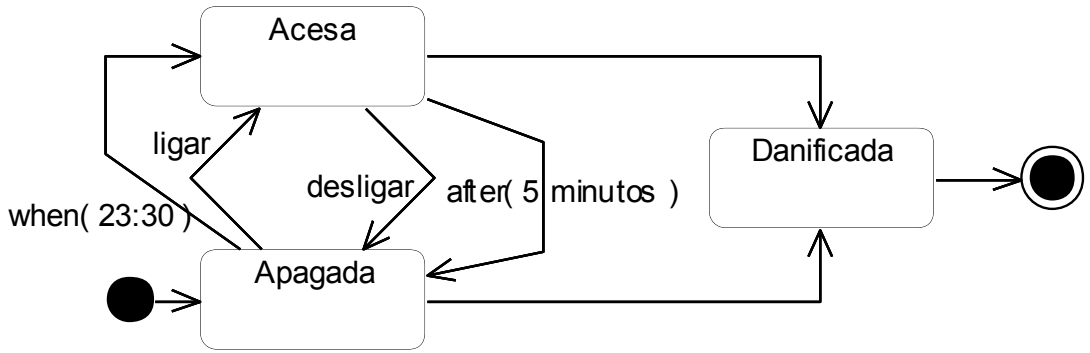
Evento de Passagem do Tempo

- Evento de Passagem de Tempo (*time event*)
  - o conhecido "fenómeno" da passagem do tempo !
- Especificação deste evento
  - normalmente não é feita explicitamente
  - no entanto, se for relevante explicitar e caracterizar este evento
    - pode-se usar a palavra chave `after`
    - seguida por uma expressão que avalia o período de tempo
    - `after( 1 dia )`
    - `after( 1 segundo após saída no estado Acesa ),...`



Evento de Mudança de Estado

- Evento de Mudança de Estado (*change in state event*)
  - mudança de estado, ou
  - satisfação de uma determinada condição
- Especificação deste evento
  - é feita usando a palavra chave `when`
    - seguida por uma expressão lógica
    - `when( time >= 12:30 )`
    - `when( altitude >= 1000 and altitude <= 1500 ),...`



Envio de Eventos

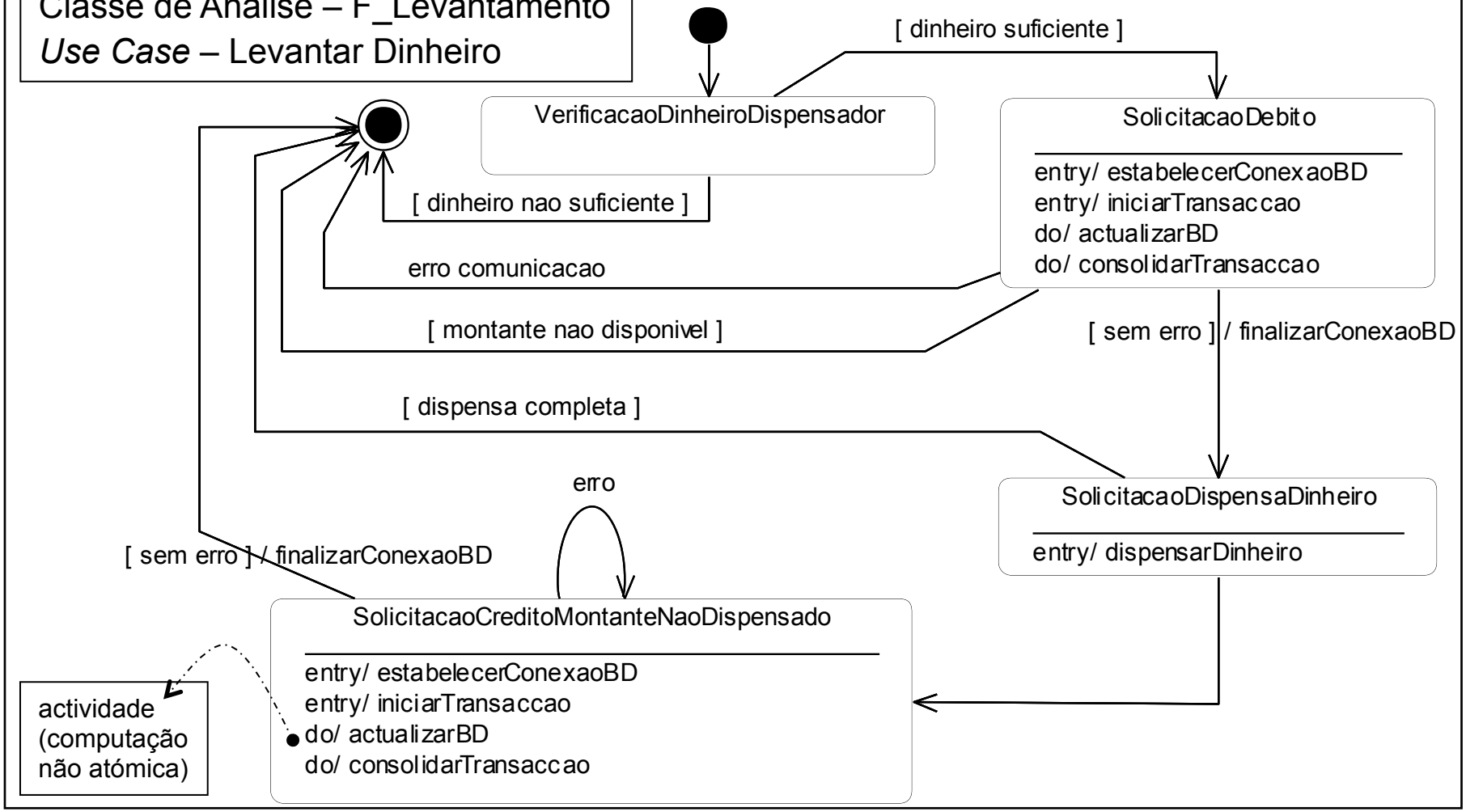
- Envio de um Sinal (*signal*)
  - o objecto que envia um sinal, despacha-o e continua o seu fluxo
  - não espera por qualquer retorno de um receptor
- Envio de um Evento de Invocação (*call event*)
  - o objecto de que invoca uma operação
  - espera que o receptor termine a execução da operação
- *multicast* e *broadcast*
  - *multicast* – envio de um sinal a um conjunto de objectos
    - para modelar colocar o objecto, a enviar o sinal, a uma colecção que contem os objectos receptores
  - *broadcast* – envio de um sinal a qualquer objecto que esteja à escuta
    - para modelar colocar o objecto, a enviar o sinal, a um outro objecto que representa todo o sistema

Recepção de Eventos

- Recepção de um Evento de Invocação (*call event*)
  - emissor e receptor "encontram-se num ponto" (estão em *rendezvous*)
  - durante o período de duração da operação
  - ou seja, o fluxo de execução do emissor é suspenso
    - até que a actividade da operação esteja concluída
- Recepção de um Sinal (*signal*)
  - não é necessário existir *rendezvous*
- Em qualquer dos casos, o evento,
  - pode ser perdido
    - se não estiver especificada uma resposta ao evento
  - pode accionar a máquina de estados do receptor
    - se tal existir
  - ou pode simplesmente desencadear a chamada normal de um método

Classes / Objectos de Desenho – Diagrama de Estados

Diagrama de Estados
Classe de Desenho – Levantamento
Classe de Análise – F_Levantamento
Use Case – Levantar Dinheiro



... do Diagrama de Estados à Implementação – exemplo

Considerando que uma frase em determinada linguagem (p.e. XML) segue a sintaxe:

```
frase ::= '<' etiqueta '>' corpo ';'
etiqueta ::= string
corpo ::= string
```

pretende-se construir um analisador lexicográfico que separe as "etiquetas" dos "corpos"

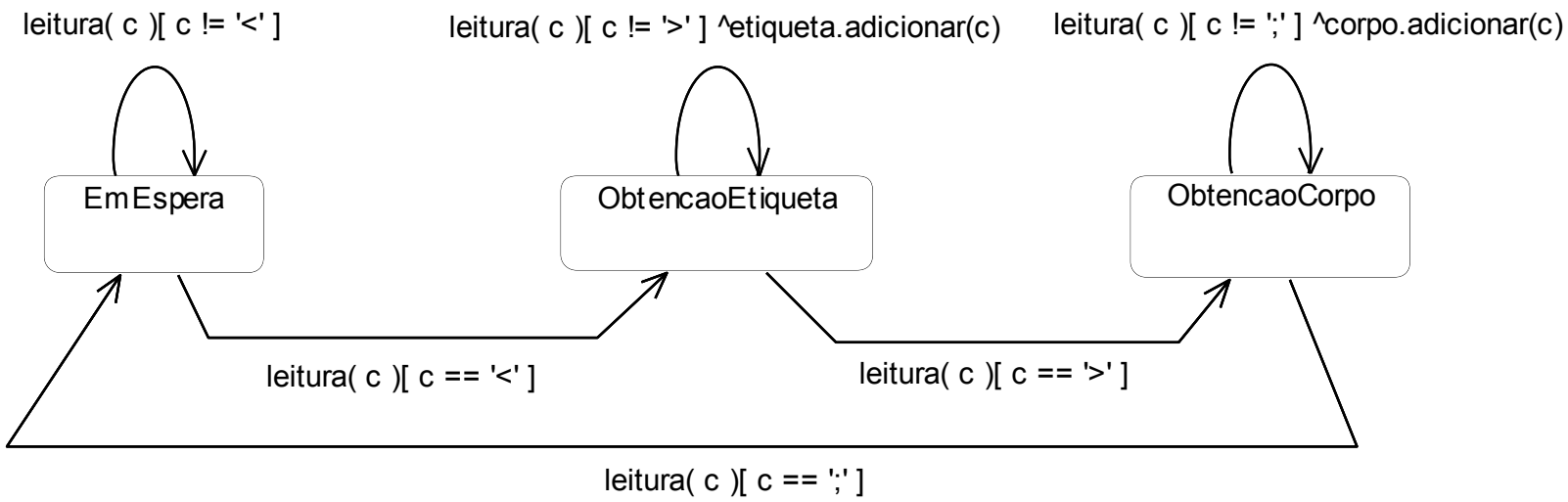
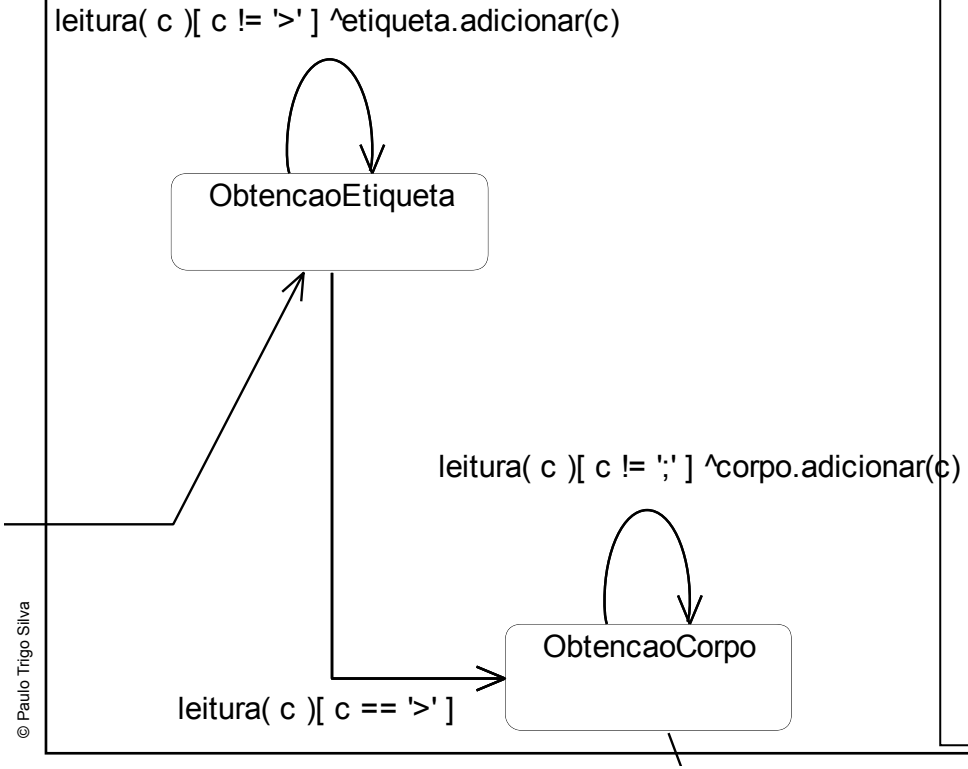


Diagrama de Estados e Implementação – exemplo




```
class Analisador {
final static int emEspera = 0;
final static int obtencaoEtiqueta = 1;
final static int obtencaoCorpo = 2;
int estado = emEspera;
StringBuffer etiqueta, corpo;

public boolean leitura( char c ) {
   switch( estado ) {
      case emEspera:
           if( c == '<' ) {
              estado = obtencaoEtiqueta;
              etiqueta = new StringBuffer();
              corpo = new StringBuffer(); }
              break;
      case obtencaoEtiqueta:
           if( c == '>' )
              estado = obtencaoCorpo;
           else
              etiqueta.adicionar( c );
           break;
      case obtencaoCorpo:
           if( c == ';' )
              estado = emEspera;
           else
              corpo.adicionar( c );
           break; }
}
```

Diagrama de Estados – quando usar ?

- Comum Diagrama de Estados modelar comportamento de
  - uma classe
  - um caso de utilização (para modelar os seus cenários)
  - todo um sistema / subsistema
- Comum usar Diagrama de Estados para modelar objectos reactivos
  - objecto reactivo (ou conduzido por eventos – *event driven*) – aquele cujo comportamento melhor se caracteriza pela sua resposta a eventos
    - está normalmente inactivo até receber um evento
    - é usual a resposta a um evento depender dos eventos anteriores
    - após resposta a um evento o objecto fica novamente inactivo
- Objectos reactivos – ênfase da modelação está na identificação,
  - dos seus estados estáveis
  - dos eventos que funcionam como gatilhos para transições
  - das acções que ocorrem em cada mudança de estado

Diagrama de Estados / Actividades

- Comum usar Diagrama de Estados para modelar objectos reactivos
  - *model the discrete stages of an object's lifetime – state centric*
- Comum usar Diagrama de Actividades para modelar
  - fluxos de trabalho ou operações
  - ... de modo idêntico ao dos *flowchart* (fluxograma)
  - *model the sequence of activities in a process – activity centric*
- Comum Diagrama de Actividades modelar comportamento de
  - uma operação (método), de uma classe
  - um caso de utilização (a sequência de actividades dos seus cenários)
  - todo o fluxo de um qualquer processo de negócio (*workflow*)
- Diagrama de Actividades é idêntico ao de Estados, no entanto,
  - maioria dos Estados são Actividades
  - maioria das transições são disparadas pela conclusão da actividade

Actividade

- Pode representar a execução de,
  - uma tarefa dentro de um fluxo de trabalho (*workfow*)
  - uma directiva dentro de um procedimento (*statement in a procedure*)
- Uma Actividade é idêntica a um Estado, mas expressa a intenção,
  - da não existência de ênfase na espera pela ocorrência de eventos
- Ligações Actividade-Actividade e Actividade-Objecto
  - Actividade-Actividade
    - através de transições (tal como com os Estados)
  - Actividade-Objecto
    - através de um "fluxo de objecto" (*object flow*)

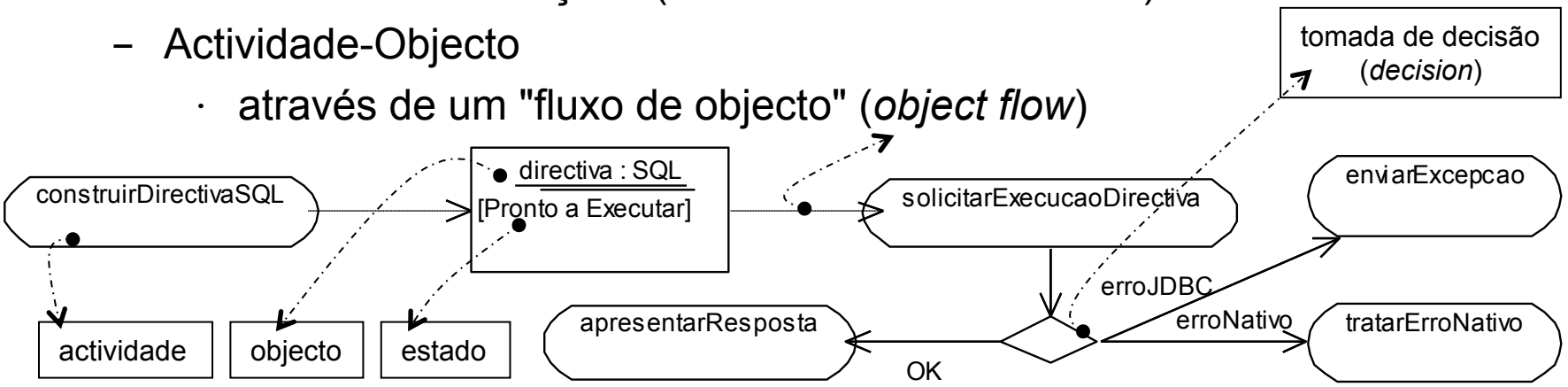


Diagrama de Actividades – *workflow*

- ... uma definição de *workflow* – fluxo de trabalho
  - *... a well-defined sequence of activities that produces an observable value or objective to an individual or entity when performed*
  - por *Rational Software Corporation*
- Objectivos da modelação de um *workflow*
  - compreender a estrutura e dinâmica de uma organização
  - garantir que clientes, utilizadores e implementadores têm um entendimento comum da organização
  - derivar os requisitos dos sistemas que suportam a organização
  - ... cada *use case* pode também ser visto como um *workflow*
- *In business and in other industries,*
  - *there are many manual and automated systems*
  - *... each of these systems contains one or more workflows*

Construção de um *workflow*

- Quando um Diagrama de Actividades especifica um *workflow* devem-se encontrar nele respostas às seguintes questões,
  - que actividades são necessárias para se atingir o objectivo ?
    - apenas contemplar as actividades mais importantes para o *workflow*
    - ... não é necessário indicar todas as actividades ou estados
  - que actividades criam ou modificam objectos ?
    - as actividades e objectos podem ser ligados por *object flows*
    - ... o objecto pode ainda ter a indicação sobre o estado em que está
  - em que momento cada actividade ou estado ocorre ?
    - a colocação de cada actividade no diagrama, dá essa indicação
  - quem tem responsabilidade no *workflow* ?
    - o diagrama de actividade pode pertencer a um *use case* ou classe
  - quem é responsável por executar as diversas actividades ou estados ?
    - definir cada actividade numa pista (*swimlane*)
    - ... cada pista é responsável pelas actividades que contem

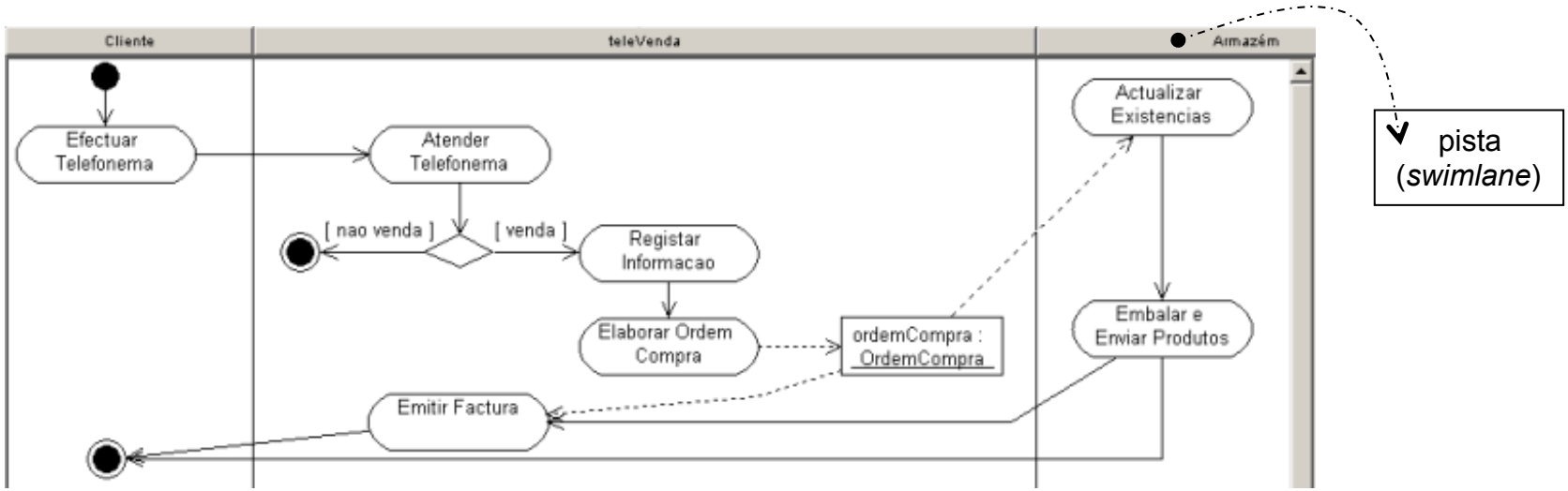
Construção de um *workflow* – exemplo

Numa empresa existe um departamento – teleVenda, responsável pela venda de produtos via telefone.

Toda a informação necessária à venda é registada durante o telefonema do cliente. Em seguida o teleVenda coloca uma ordem de compra e transmite-a a um dos armazéns da empresa.

No armazém a ordem de compra é analisada e os produtos embalados. Em seguida o armazém envia os os produtos ao cliente.

O armazém indica ao teleVenda que os produtos já seguiram. Nesse momento, o teleVenda emite uma factura e envia-a ao cliente.



Estado, Actividade, Acção

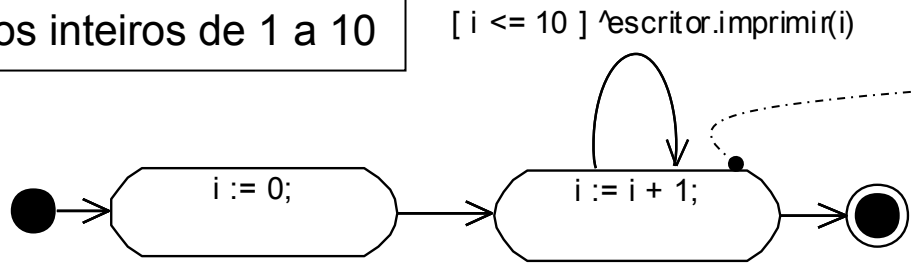
- ... uma definição de estado – *state*
  - *... represents a condition or situation during the life of an object during which it satisfies some condition or waits for some event*.
  - *... each state represents the cumulative history of its behaviour*.
- ... uma definição de actividade – *activity* (caso particular de Estado)
  - *... the performance of task or duty in a workflow.*
  - *... may also represent the execution of a statement in a procedure.*
  - *... is similar to a state, but expresses the intent that there is no significant waiting (for events) in an activity*.
- ... uma definição de acção – *action*
  - *best described as a "task" that takes place while inside a state or activity*
  - *each state and activity on a statechart or activity diagram may contain any number of internal actions*
  - *there are four possible actions – on entry; on exit; do; on event*

... o Estado Acção e o Estado Actividade

- o Estado que é uma Acção
  - no fluxo modelado por um Diagrama de Actividades acontecem coisas ...
    - são atribuídos valores a variáveis,
    - são criados e destruídos objectos,
    - são invocadas operações sobre objectos,
    - são enviados sinais a objectos, ...
  - quando uma coisa acontece no contexto de uma computação atómica
    - designa-se por "estado acção" (*action state*)
    - computação atómica – eventos ocorrem mas acção não interrompe
    - ... estes estados não podem ser decompostos

Diagrama de Actividades

- impressão dos inteiros de 1 a 10



acção

expressão – UML não define sintaxe – usar p.e. a de uma linguagem de programação

... o Estado Acção e o Estado Actividade (cont.)

- o Estado que é uma Actividade
  - a computação que ocorre não é atómica
    - eventos ocorrem e podem interromper a actividade
  - ... estes estados podem ser decompostos
- Um Estado Acção pode ser visto como
  - um caso particular de um Estado Actividade
  - ... é um Estado Actividade que já não pode ser decomposto
- Um Estado Actividade pode ser visto como uma composição de
  - outros Estados Actividade e de
  - Estados Acção

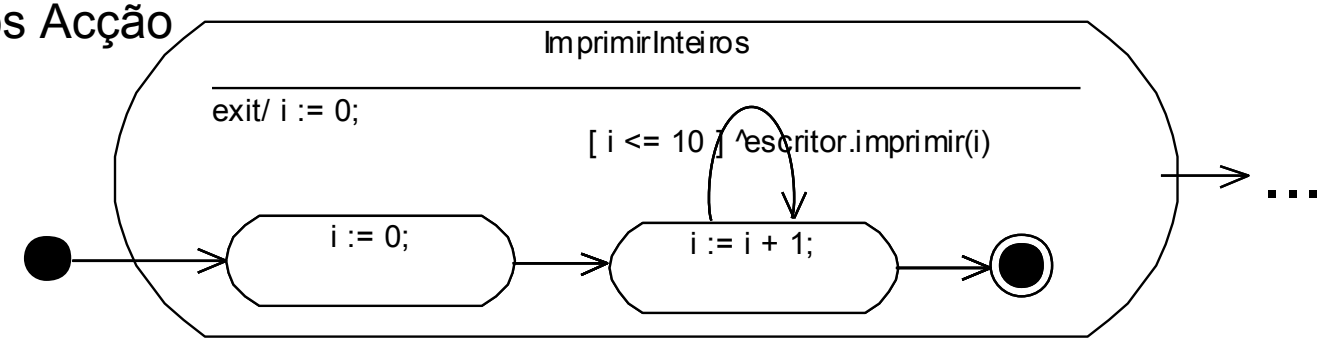
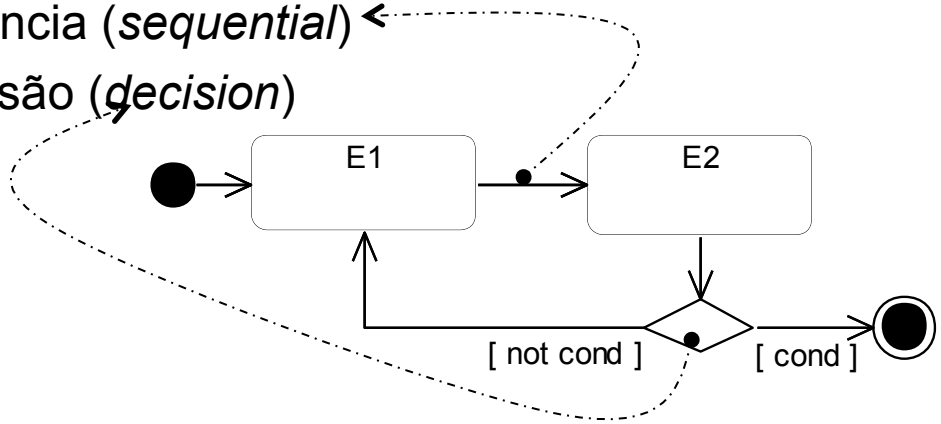


Diagrama de Estados / Actividades – difusão e junção

- As transições mais usuais são,
  - as de sequência (*sequential*)
  - e as de decisão (*decision*)



- No entanto, podem-se modelar fluxos concorrentes,
  - utilizando uma barra de sincronização (*synchronization bar*)

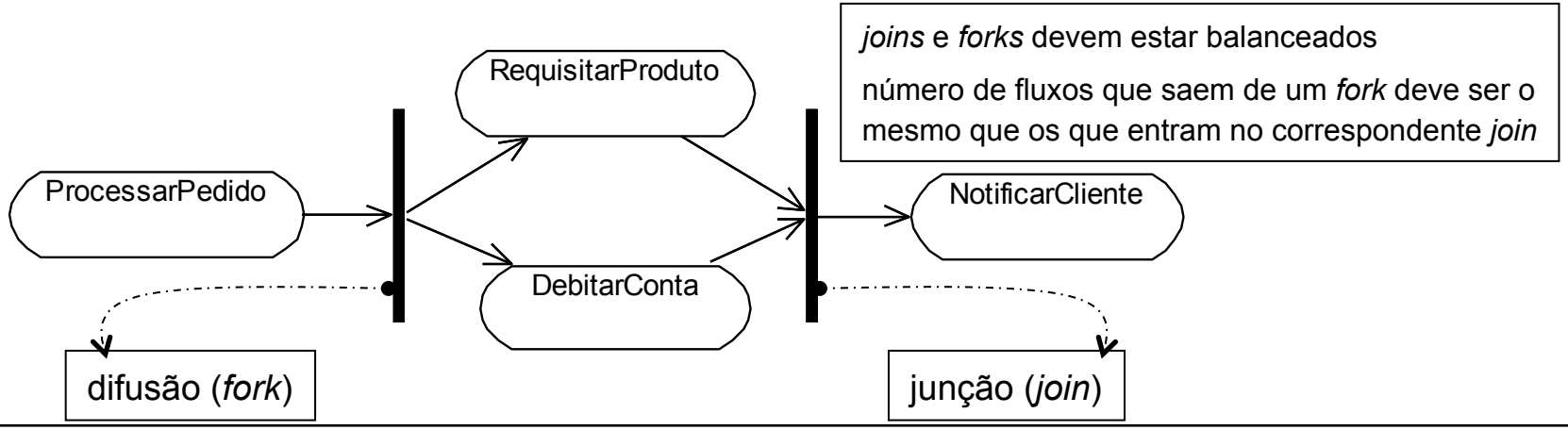


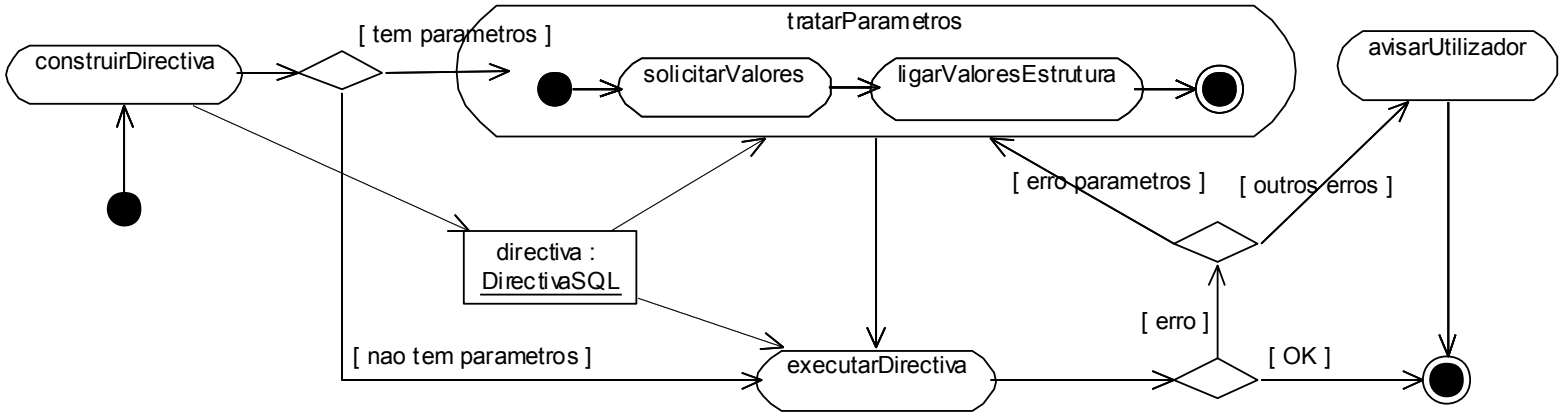
Diagrama de Actividades – especificação de um método

Para se efectuar um levantamento no ATM é necessário executar uma directiva SQL de actualização da base de dados das contas dos clientes.

A construção de uma directiva devolve um objecto da classe "DirectivaSQL", que irá ser analisado, pois a sua definição pode não estar ainda completa.

Uma directiva SQL pode ou não ter parâmetros – caso não tenha parâmetros, a directiva pode ser executada; se tiver parâmetros eles têm que ser lidos para variáveis e essas variáveis têm que ser ligadas aos elementos da estrutura que representa a directiva; depois dessa leitura e ligação a directiva pode então ser executada.

A execução da directiva pode devolver uma indicação de erro – se o erro tiver que ver com a leitura dos parâmetros a leitura deve ser repetida; caso contrário o utilizador deve ser informado do erro ocorrido.



Modelo de Implementação

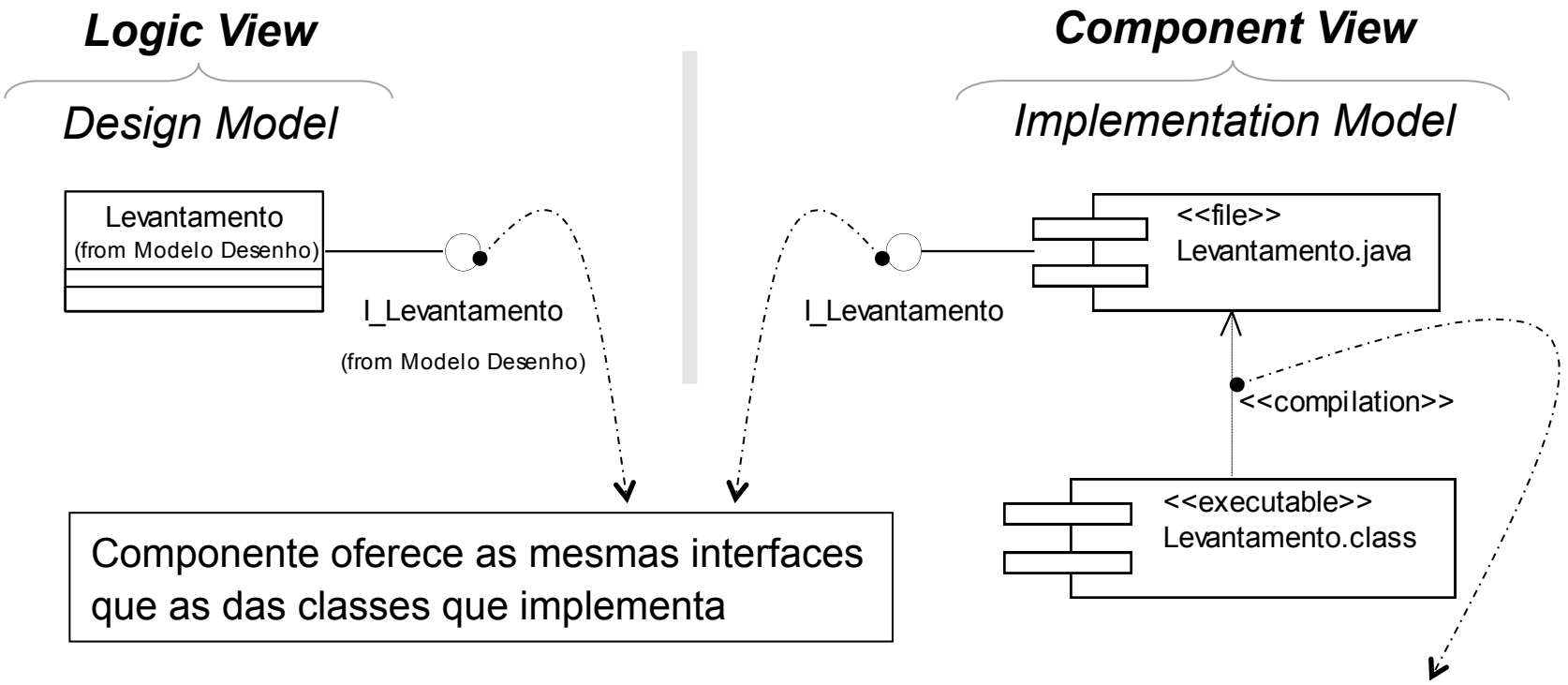
- Tem como pressuposto (*input*)
  - as Classes de Desenho
  - os Subsistemas identificados
- Tem como objectivo (*output*)
  - transformar Classes de Desenho em Componentes com código fonte
  - compilar e ligar (*compile and link*) os Componentes de cada Subsistema
    - construir executáveis e bibliotecas
  - implantar (*deploy*) Componentes executáveis em nós
- Quando construir um Modelo de Implementação ?
  - numa fase inicial para definir, criar e validar a arquitectura do sistema
  - na fase de construção do sistema
  - numa fase posterior de transição para novas versões
  - ... representa a efectiva concretização do sistema pelo que deve ser mantido ao longo de todo o processo de desenvolvimento

Modelo de Implementação – Componentes

- Componentes correspondem no Modelo de Implementação
  - ao "empacotamento" físico de Classes de Desenho
- Alguns estereótipos de Componentes
  - <<executable>> programa que pode executar num nó
  - <<file>> um ficheiro que contém algum código fonte ou dados
  - <<library>> uma biblioteca estática ou dinâmica
  - <<document>> um documento
  - <<table>> uma tabela numa base de dados
- Algumas características dos Componentes
  - comum um componente implementar diversas Classes de Desenho
    - isso pode depender das capacidades da linguagem de programação
  - comum terem relações de *trace* para os elementos que implementam

Componentes, Interfaces e Classes de Desenho

No sistema ATM, a classe de desenho "`Levantamento`" é implementada no componente de código fonte "`Levantamento.java`" – é comum, em *Java*, criar um ficheiro "`.java`" para cada classe (embora isso não seja obrigatório).



Componentes BD, Tabelas e Classes de Desenho

No sistema ATM, a classe de desenho "`ContaPersistente`" é implementada através de uma tabela gerida pelo sistema de gestão de base de dados (SGBD) do banco.

O banco utiliza o DB2 como SGBD e a base de dados que contém os dados das contas é designada por "`ContasClientes`".

Logic View

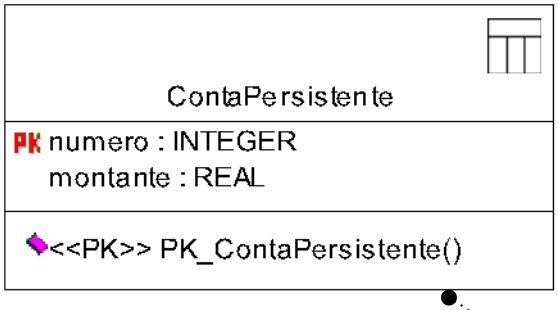
Design Model

ContaPersistente

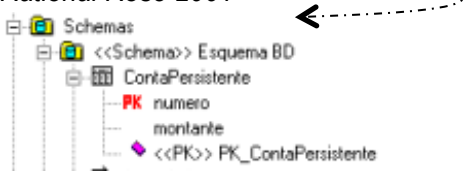
numero : int
montante : float

Implementation Model

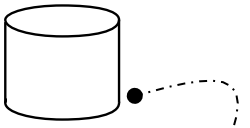
Logic View



Rational Rose 2001

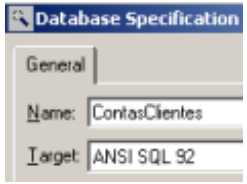


Component View



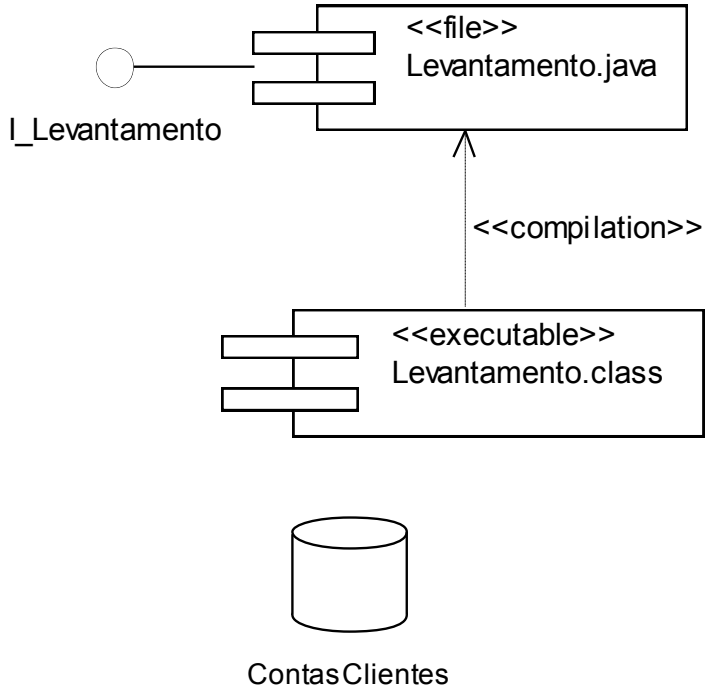
ContasClientes

Rational Rose 2001



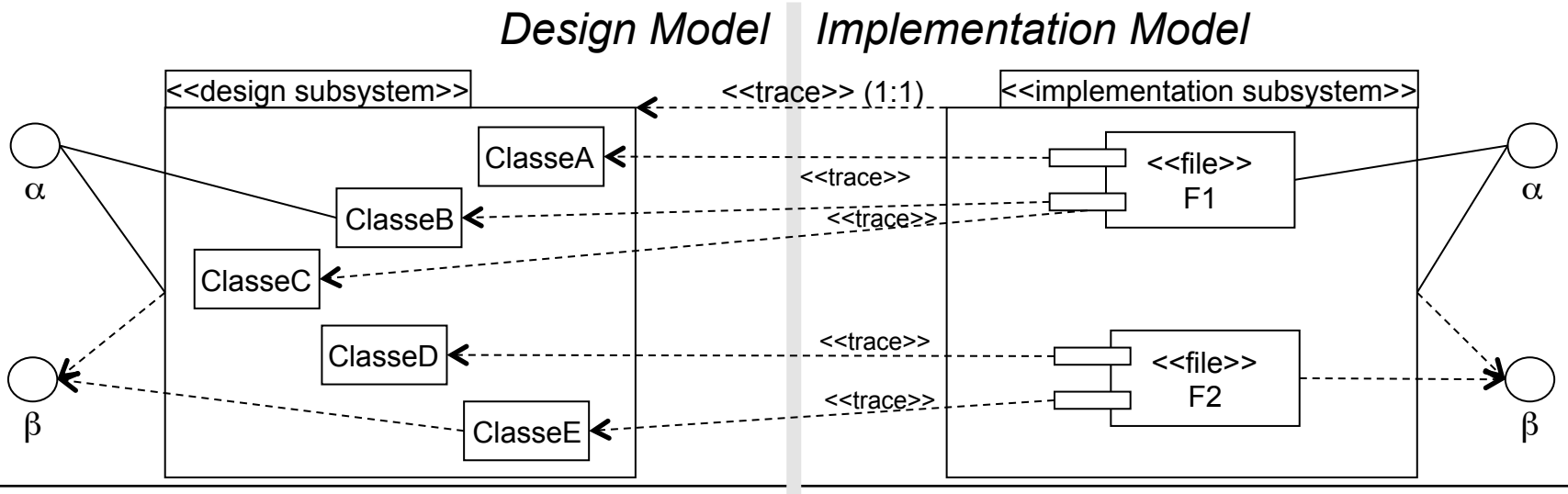
Modelo de Implementação / Diagramas de Componentes

- Capturam a estrutura física
  - de implementação
- São construídos
  - como elementos da especificação da arquitectura do sistema
- Têm como objectivo
  - organizar o código fonte
  - construir um suporte executável
  - especificar uma base de dados física
- São desenvolvidos por
  - arquitectos de sistemas (*architects*)
  - programadores (*programmers*)



Modelo de Implementação / Subsistemas

- Subsistema do Modelo de implementação (*implementation subsystem*)
  - *package* – em *Java*
  - *project* – em *Visual Basic*
  - *directory of files* – em *C++*
  - *component view package* – no *Rational Rose*
- Estreita ligação com os Subsistemas do Modelo de Desenho
  - devem ser isomorfos – relação 1:1



Modelo de Implantação (*Deployment*)

- Através de Diagramas de Implantação
  - capturam a topologia de *hardware* do sistema
- Os Diagramas de Implantação são também construídos
  - como elementos da especificação da arquitectura do sistema
- Têm como objectivo
  - especificar a distribuição dos componentes
  - identificar estrangulamentos (*bottlenecks*) no desempenho
- São desenvolvidos por
  - arquitectos de sistemas (*architects*)
  - especialistas em redes (*networking engineers*)

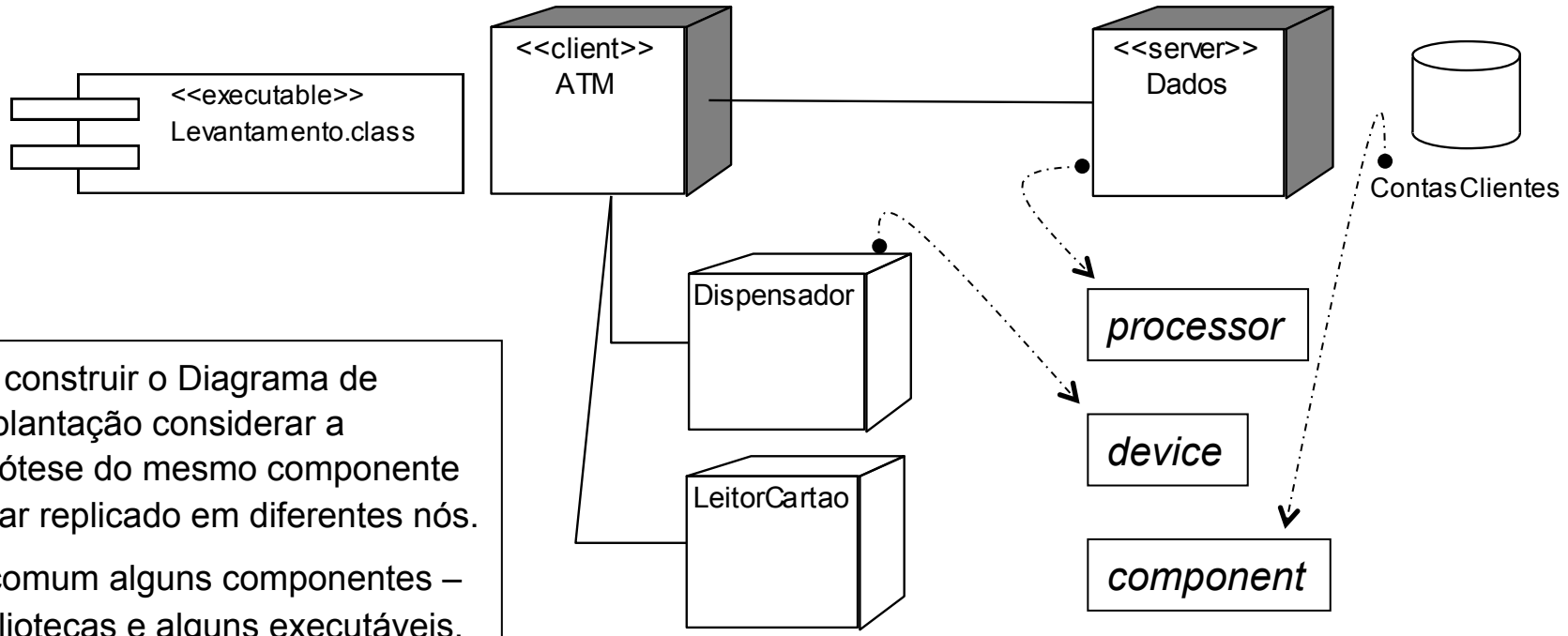
Modelo de Implantação / Diagramas de Implantação

- Diagrama de Implantação contem
  - nós (*nodes*)
  - relações de associação entre nós
- Dois tipos de Nó
  - dispositivo (*device*)
    - não tem capacidade de processamento (que seja modelada a este nível de abstracção)
    - normalmente representa algo que funciona como uma interface para o "mundo real", tal como *modem*, terminal, sensor, ...
  - processador (*processor*)
    - tem capacidade de processamento
    - pode executar um componente
    - p.e. *motherboard*, computador pessoal, servidor, telemóvel, ...

Diagrama de Implantação – exemplo

O banco existe um computador (servidor) onde reside o SGBD.

Cada ATM (cliente) dialoga directamente com esse computador para obter e actualizar os dados que manipula ao responder às solicitações de cada utilizador.



Ao construir o Diagrama de Implantação considerar a hipótese do mesmo componente estar replicado em diferentes nós.

É comum alguns componentes – bibliotecas e alguns executáveis, residirem simultaneamente em múltiplos nós.

Modelo de Teste

- É uma colecção de
  - casos de teste
  - procedimentos de teste
  - componentes de teste
- Casos de Teste
  - baseado num *use case*
    - verificar resultado da interacção entre actores e sistema
    - verificar que as pré e pós condições dos *use case* são satisfeitas
    - verificar cumprimento das sequências de acções especificadas
    - ... teste caixa preta (*black-box test*) – do comportamento externo
  - baseado numa *use case realization*
    - verificar correcção de cada cenário
    - verificar cumprimento da interacção entre componentes
    - ... teste caixa branca (*white-box test*) – do comportamento interno

Outros Casos de Teste

- Do sistema como um todo
  - teste de instalação (*instalation test*)
    - verificar que o sistema pode ser instalado na plataforma do cliente
    - verificar que o sistema opera correctamente quando instalado
  - teste de configuração (*configuration test*)
    - verificar correcção do sistema em diferentes configurações
    - p.e. diferente topologia de rede; um JDBC de outro fabricante; ...
  - teste negativo ou teste de caso excessivo (*negative test* ou *abuse case*)
    - usar o sistema de formas para as quais não foi desenhado de modo a poder identificar os aspectos de principal fragilidade
    - p.e. capacidade insuficiente de *hardware*; direito acesso errado; ...
  - teste de carga (*stress test*)
    - identificar problemas resultantes de recursos insuficientes
    - identificar problemas quando existe competição pelos recursos

Procedimentos e Componentes de Teste

- Especificam como efectuar cada Caso de Teste
  - procedimento
    - lista de acções a efectuar manualmente
  - componente
    - automatiza um ou mais procedimentos
- Componente de Teste
  - pode ser desenvolvido com uma linguagem de *script*
    - fornece *input* de teste
    - controla e monitoriza execução do componente em teste
    - constrói relatórios com os resultados do teste

Modelo de Teste – exemplo