



Desenvolvimento de Sistemas Software

Ponto de situação



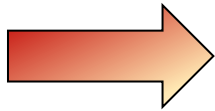
Fases do ciclo de vida do desenvolvimento de sistema

Planeamento

- Decisão de avançar com o projecto
- Gestão do projecto

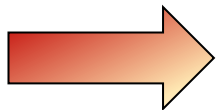
Análise

- Análise do domínio do problema
- Análise de requisitos



Concepção

- Concepção da Arquitectura
- Concepção do Comportamento

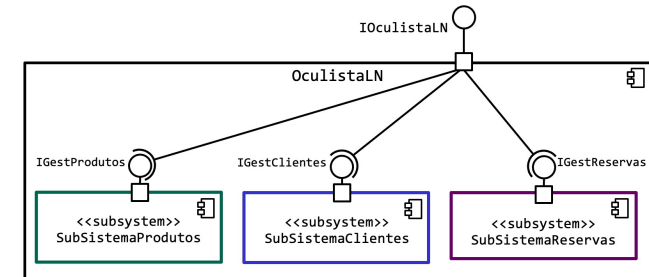
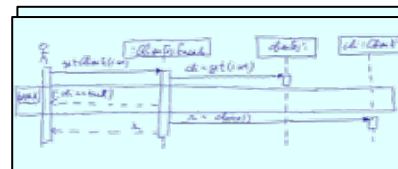
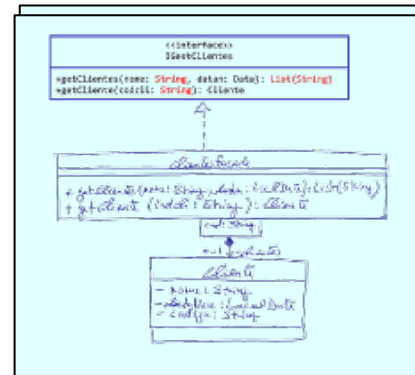
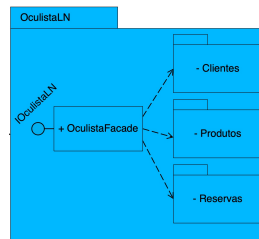
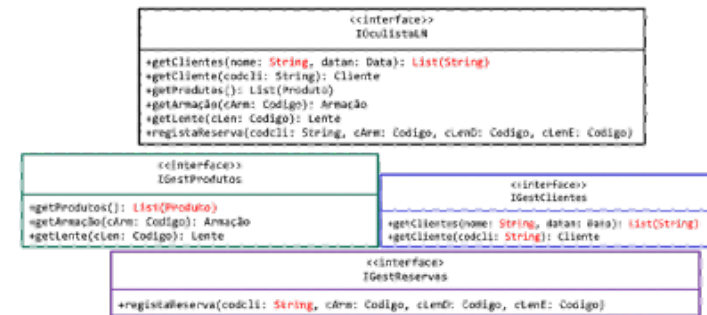
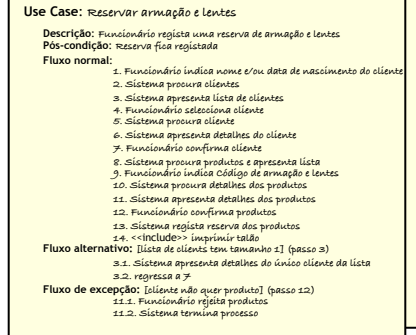
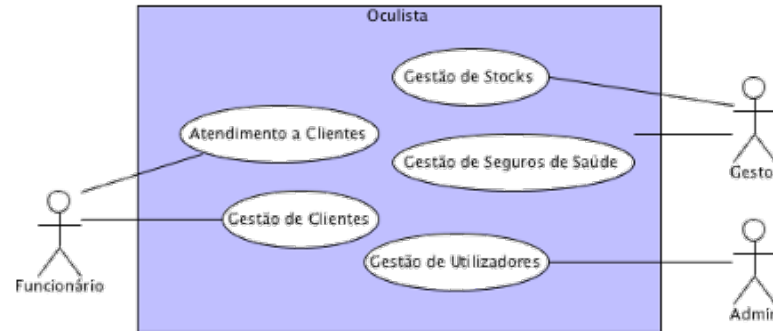
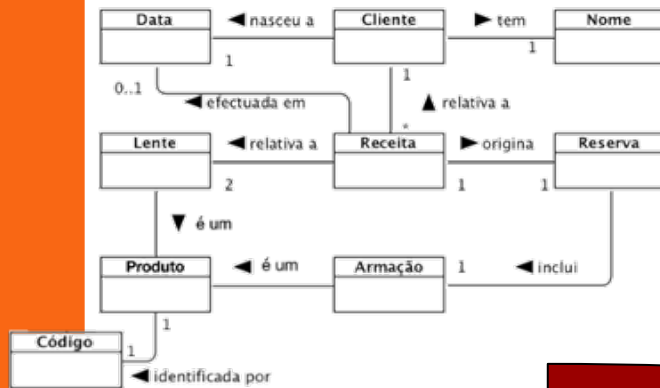


Implementação

- Construção
- Teste
- Instalação
- Manutenção



Resumindo o exemplo...



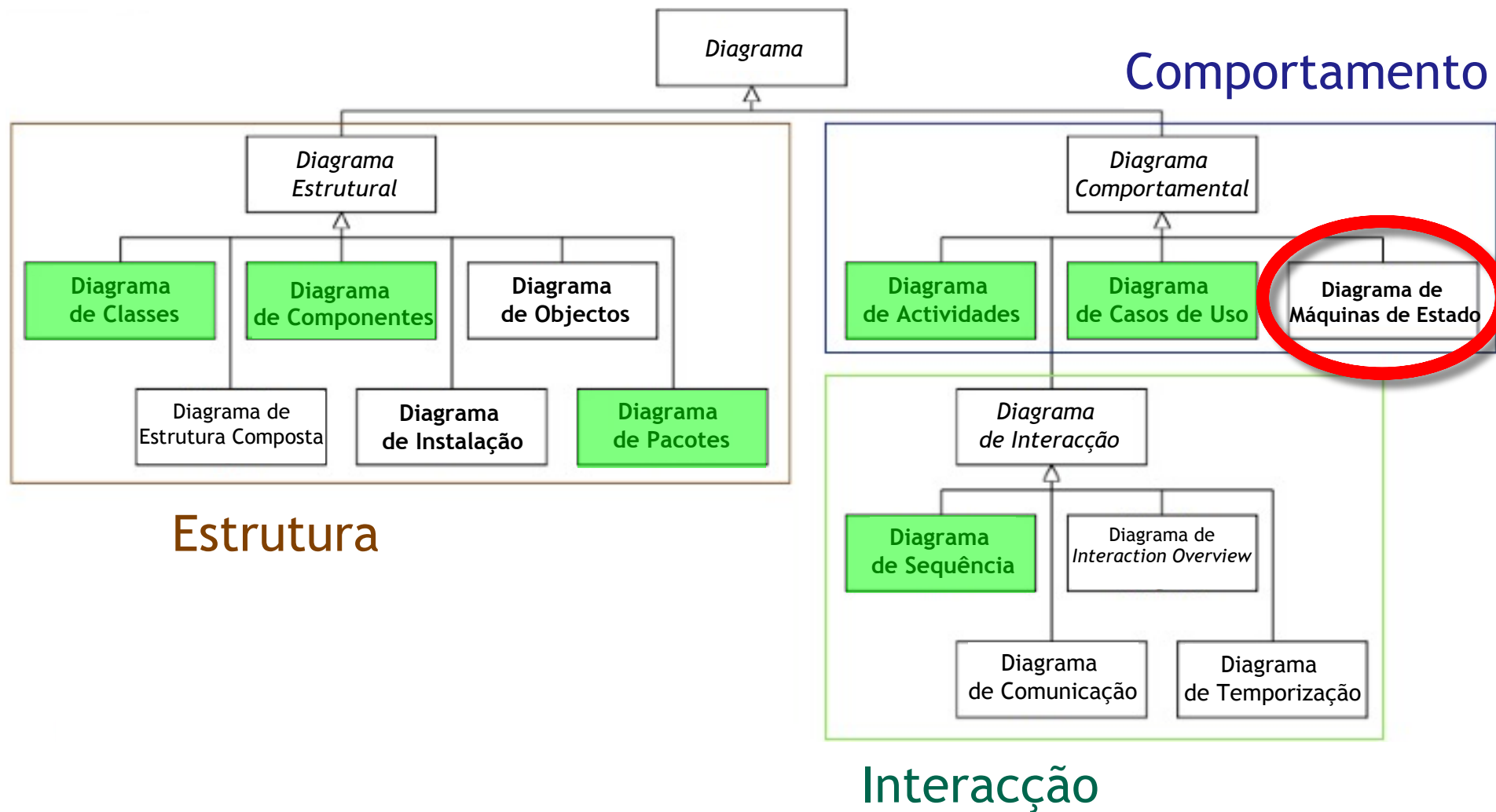


Desenvolvimento de Sistemas Software

Modelação Comportamental (Máquinas de Estado)



Diagramas da UML 2.x





Introdução aos Diagramas de Estado – Aplicação

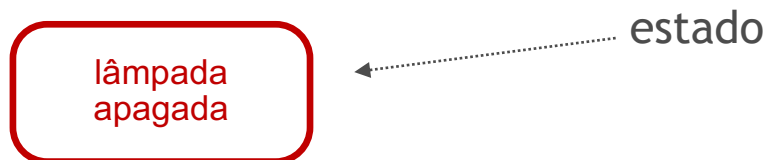
- Os Diagramas de Estado permitem modelar o comportamento de um dado objecto/sistema de forma global.
- A ênfase é colocada no estado do objecto/sistema — modelam-se todos os estados possíveis que o objecto/sistema atravessa em resposta aos eventos que podem ocorrer.
- Úteis para modelar:
 - O comportamento de um objecto de forma transversal aos use case do Sistema
 - O Sistema como um todo
- Devem utilizar-se para entidades/classes em que se torne necessário compreender o comportamento do objecto de forma global ao sistema.
 - Nem todas as entidades/classes vão necessitar de diagramas de estado.



Diagramas de Estado

Notação base

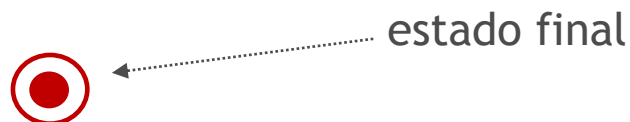
- Estado — define uma possível estado do objecto (normalmente traduz-se em valores específicos dos seus atributos)



- Estado inicial — estado do objecto quando é criado



- Estado final — destruição do objecto

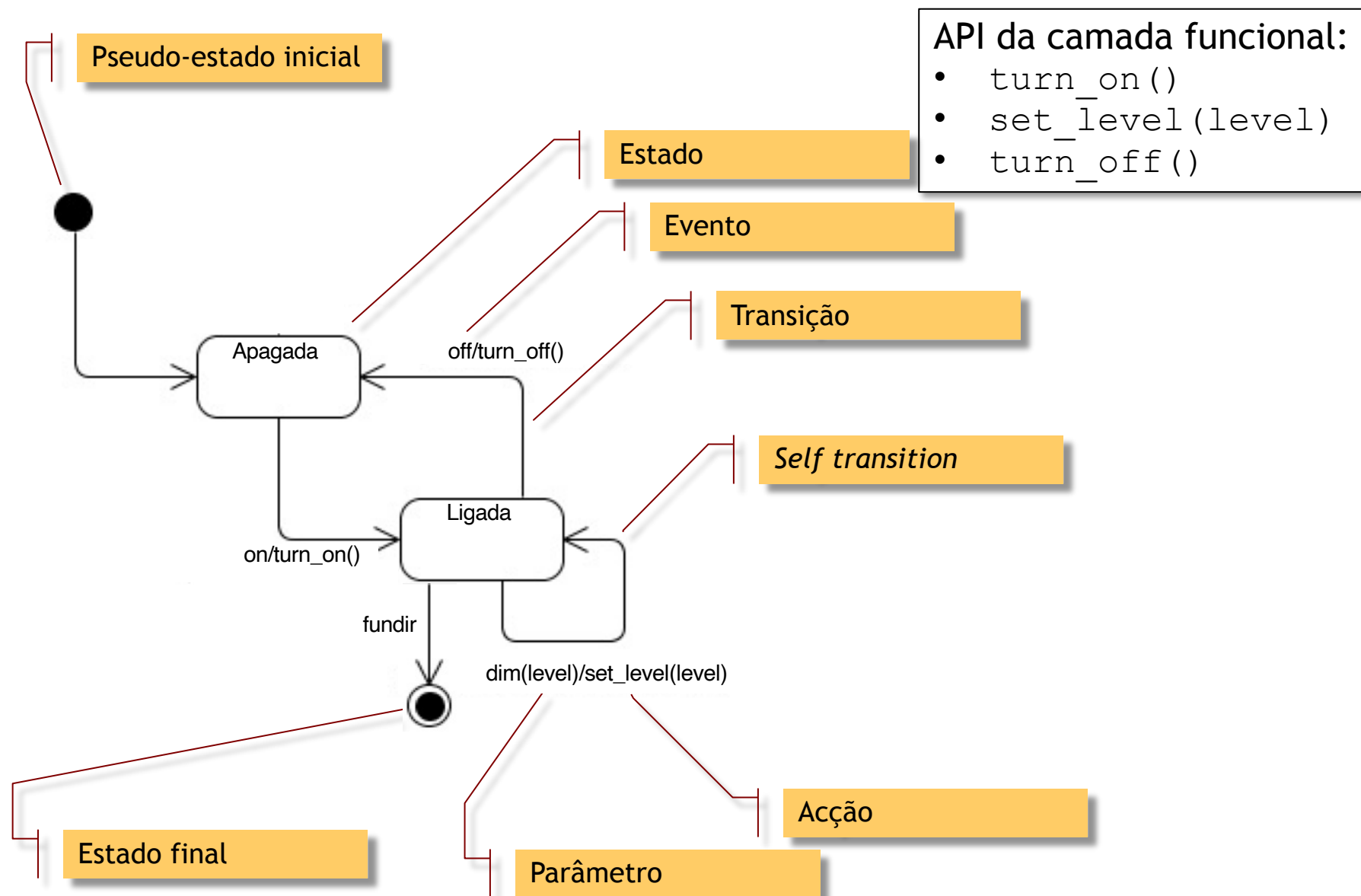


- Transições — evento[guarda]/acção (todos são opcionais!)





Maquina de Estados básica

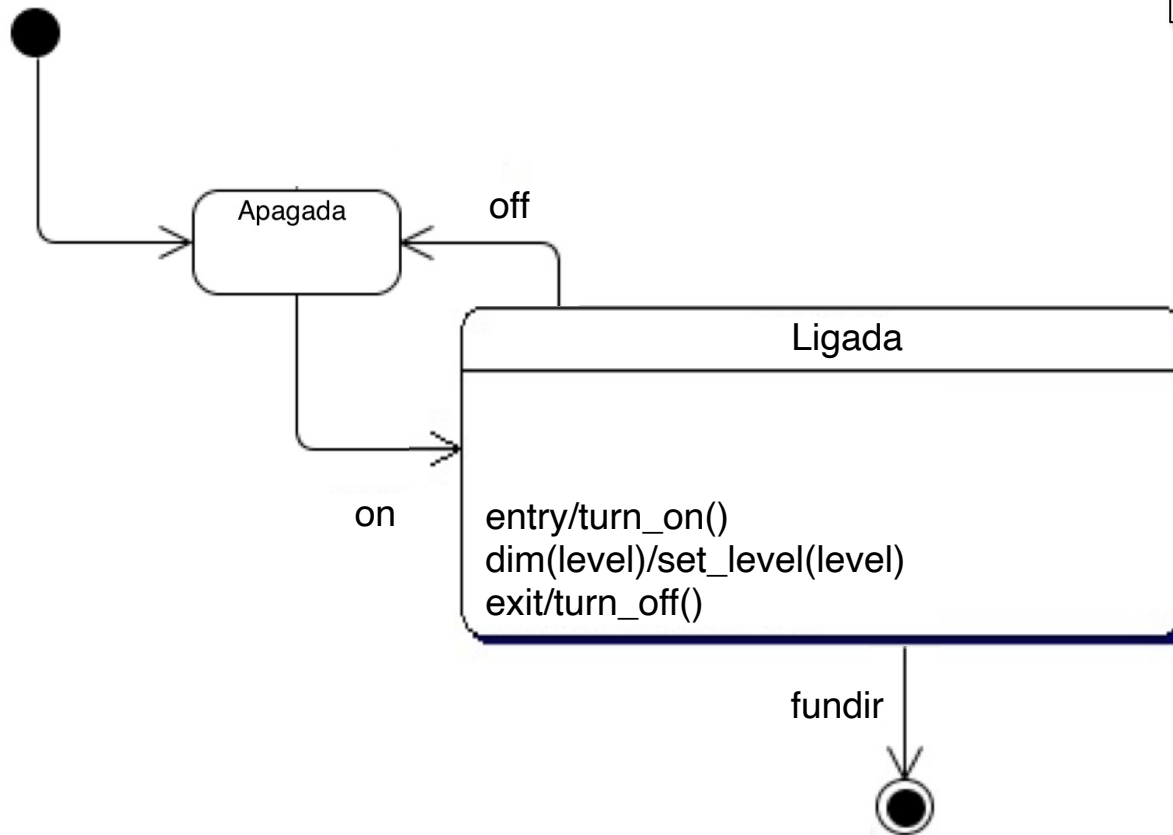




Actividades internas

API da camada funcional:

- `turn_on()`
- `set_level(level)`
- `turn_off()`





Actividades internas

- Actividades que não provocam transições de estado...

entry/acção

- “acção” é automaticamente executada quando o objecto entra no estado;

evento/acção

- “acção” é automaticamente executada se “evento” ocorrer (transição interna);

do/acção

- “acção” é continuamente executada enquanto o objecto estiver no estado;

evento/defer

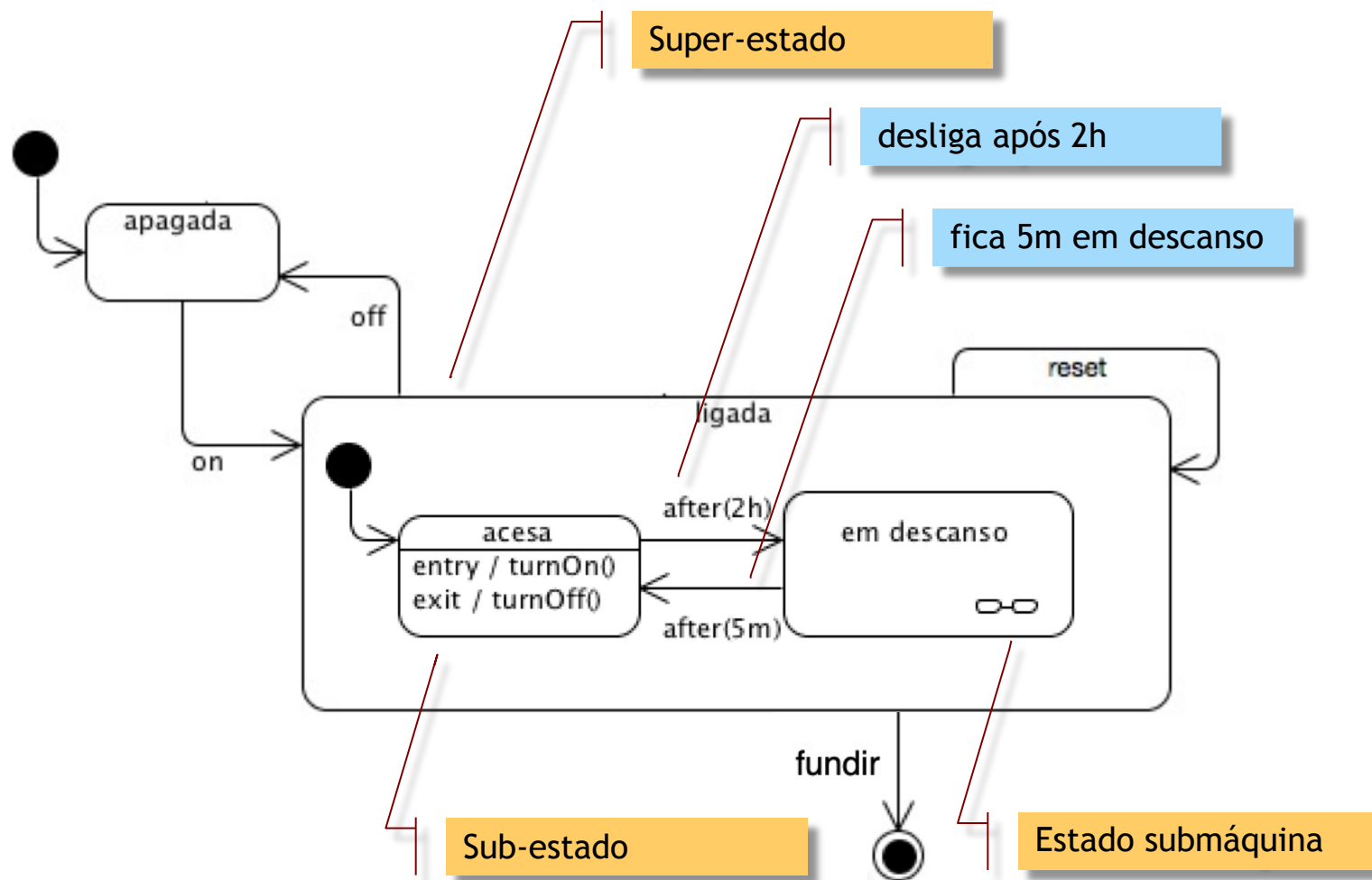
- “evento” é deferido até o estado actual ser abandonado;

exit/acção

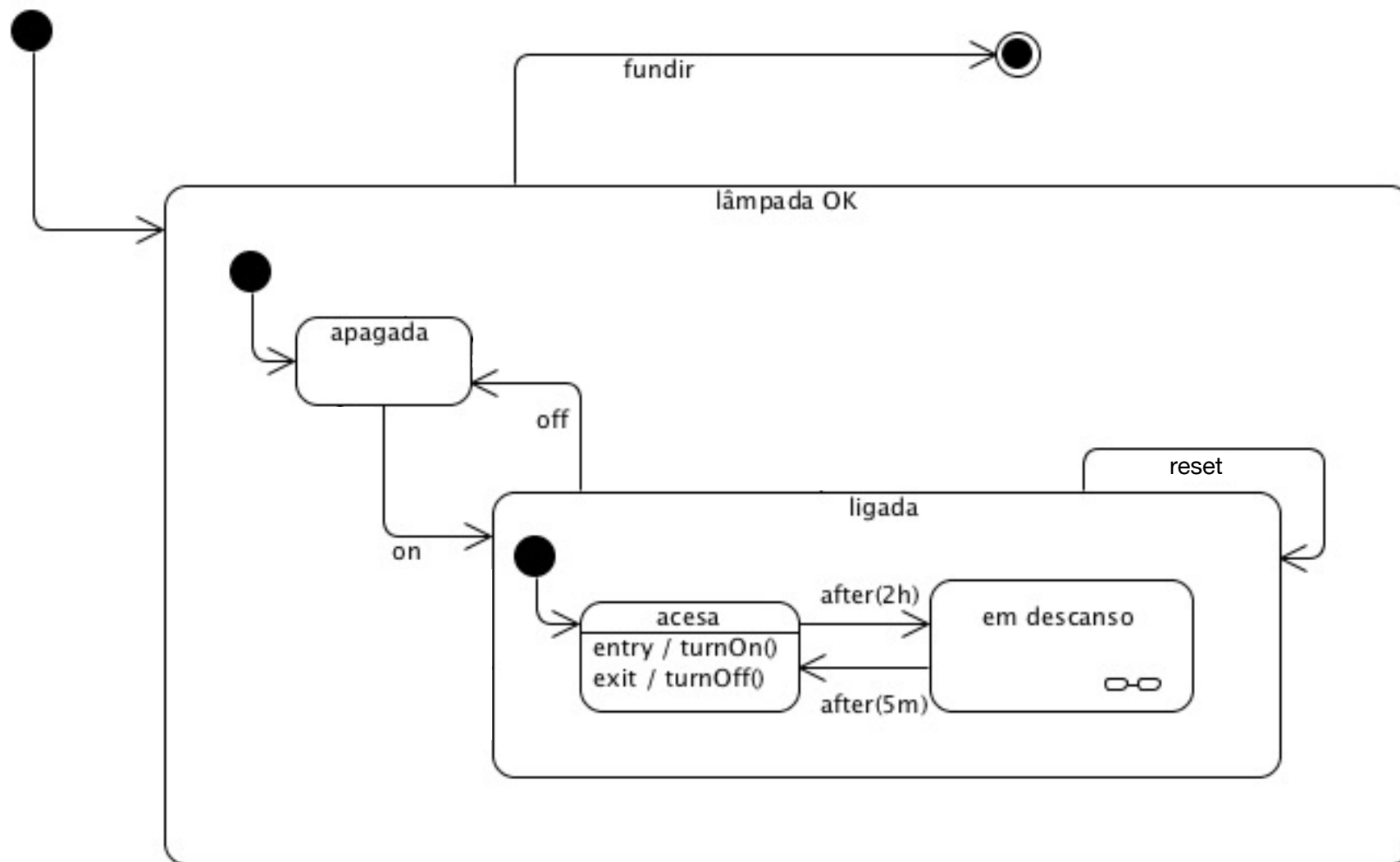
- “acção” é automaticamente executada quando o objecto sai do estado.



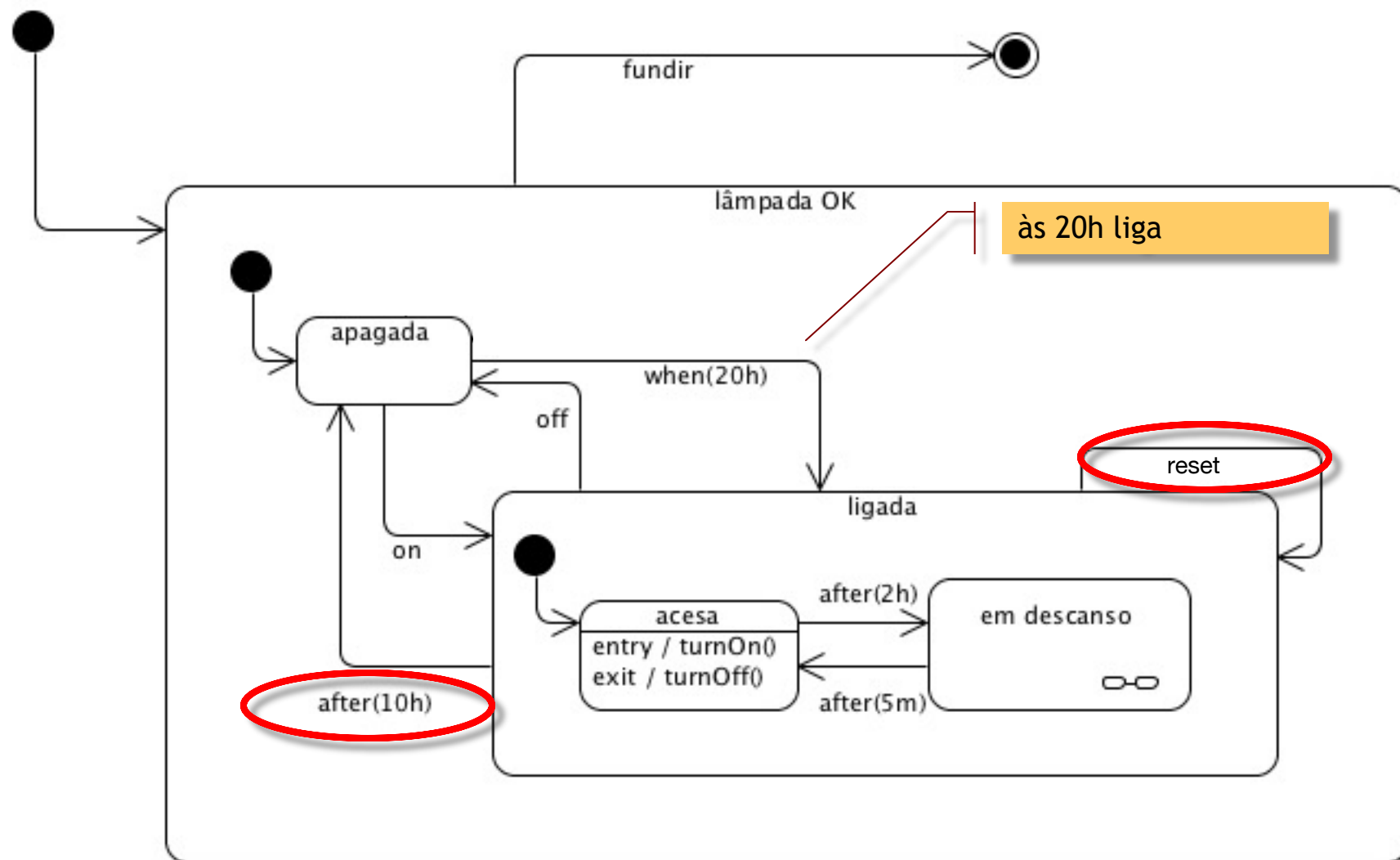
Estados e estados compostos (super-estados)



Eventos *when* / *after*

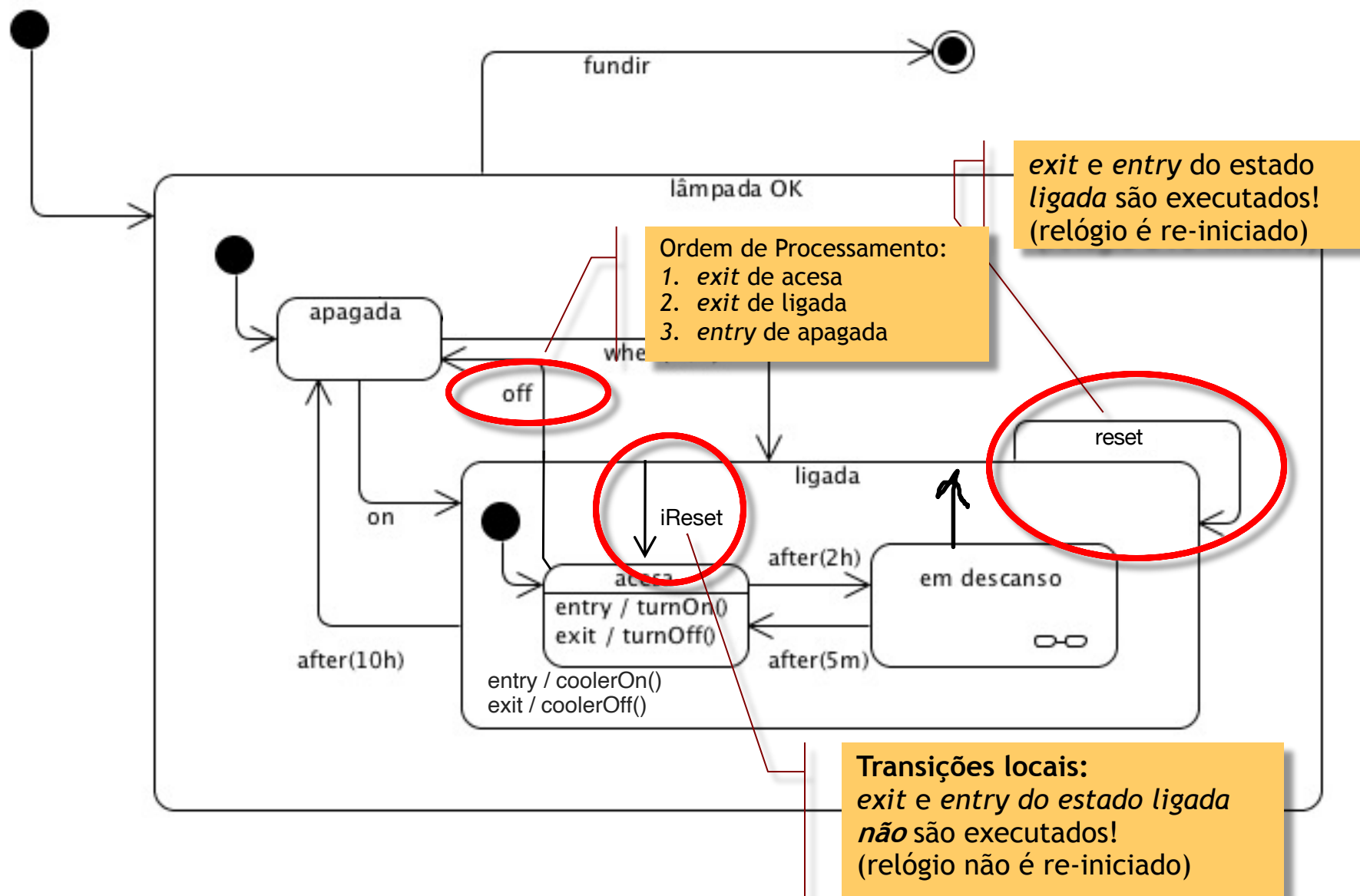


Eventos *when* / *after*





Transições vs. actividades internas

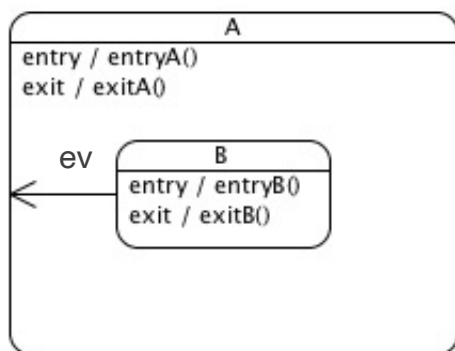




Transições locais vs. transições externas

Em resposta ao evento **ev**, o modelo...

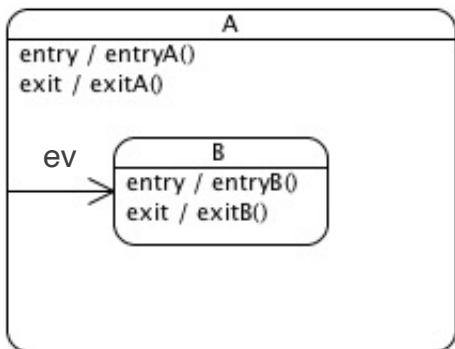
Transições locais



Executa:

- exitB()
- ...

(sub-estado para super-estado)

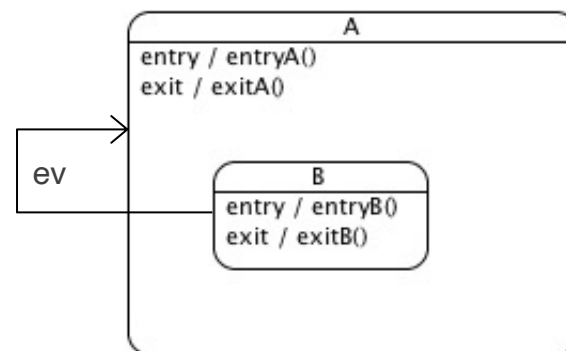


Executa:

- ...
- entryB()

(super-estado para sub-estado)

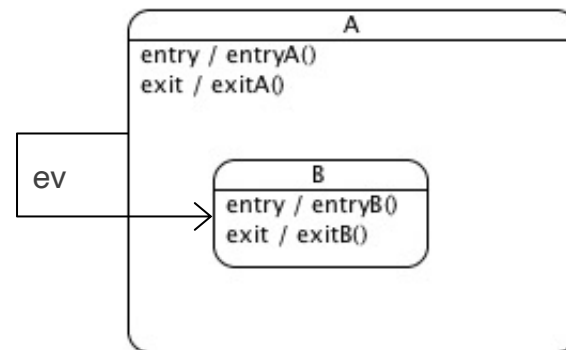
Transições externas



Executa:

1. exitB()
2. exitA()
3. entryA()

(sub-estado para super-estado)



Executa:

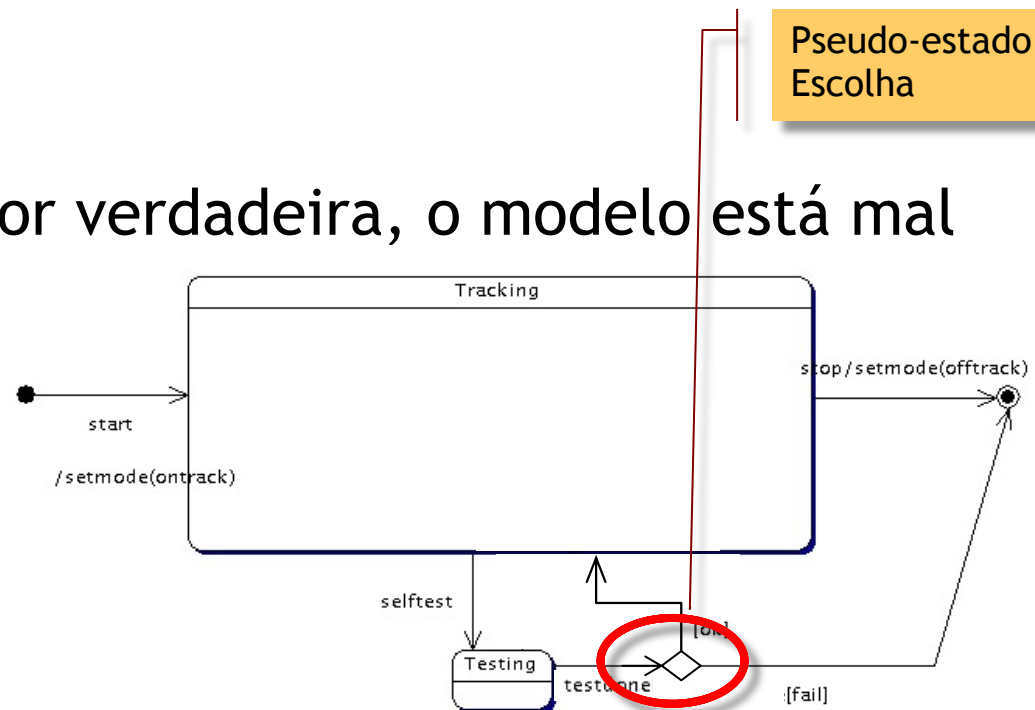
1. exitA()
2. entryA()
3. entryB()

(super-estado para sub-estado)



Pseudo-estado de Escolha

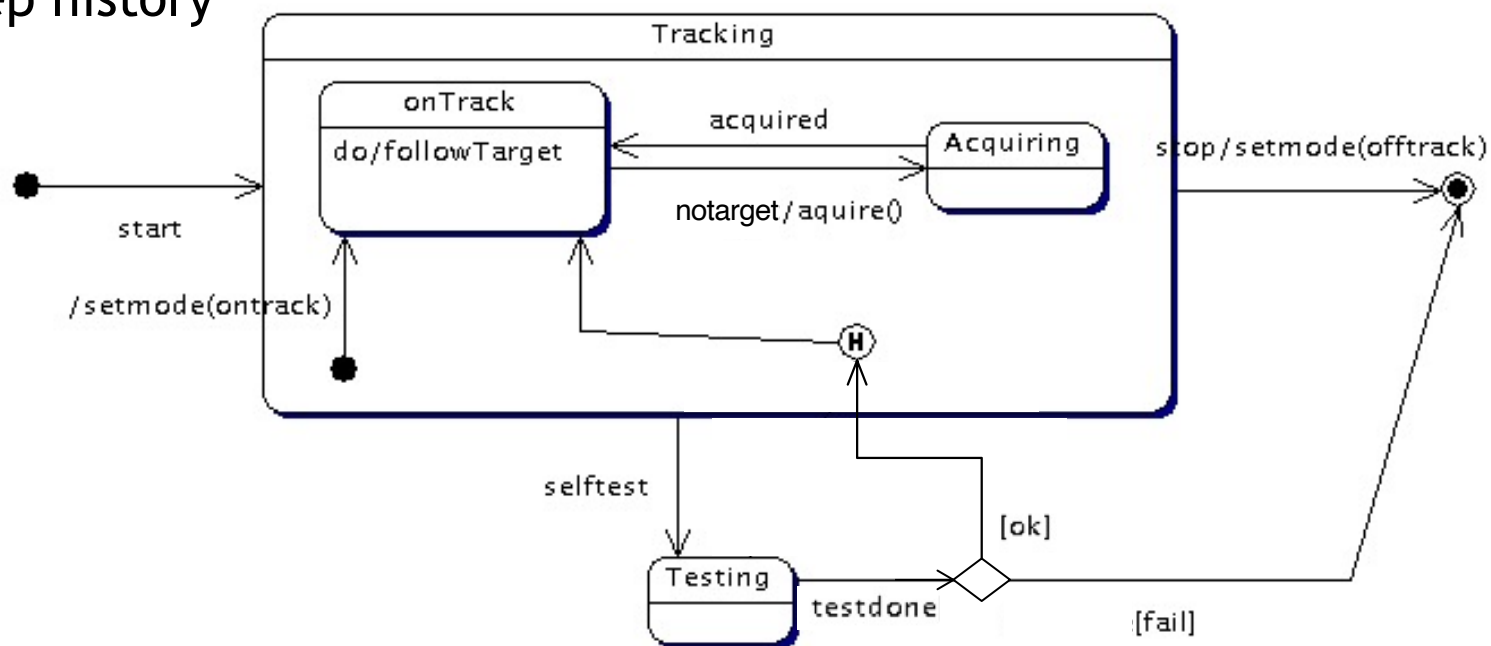
- Ramificação condicional (dinâmica!) em função do valor de uma expressão.
- Decisão pode ser uma função de acções anteriores.
- Caso mais que uma guarda verdadeira, a escolha é não determinística.
- Se nenhuma guarda for verdadeira, o modelo está mal formado ([else]!)





Pseudo-estados de História

- Permitem modelar interrupções – actividade da máquina é retomada no estado em que se encontrava aquando da última saída
- \textcircled{H} shallow history
- $\textcircled{H^*}$ deep history





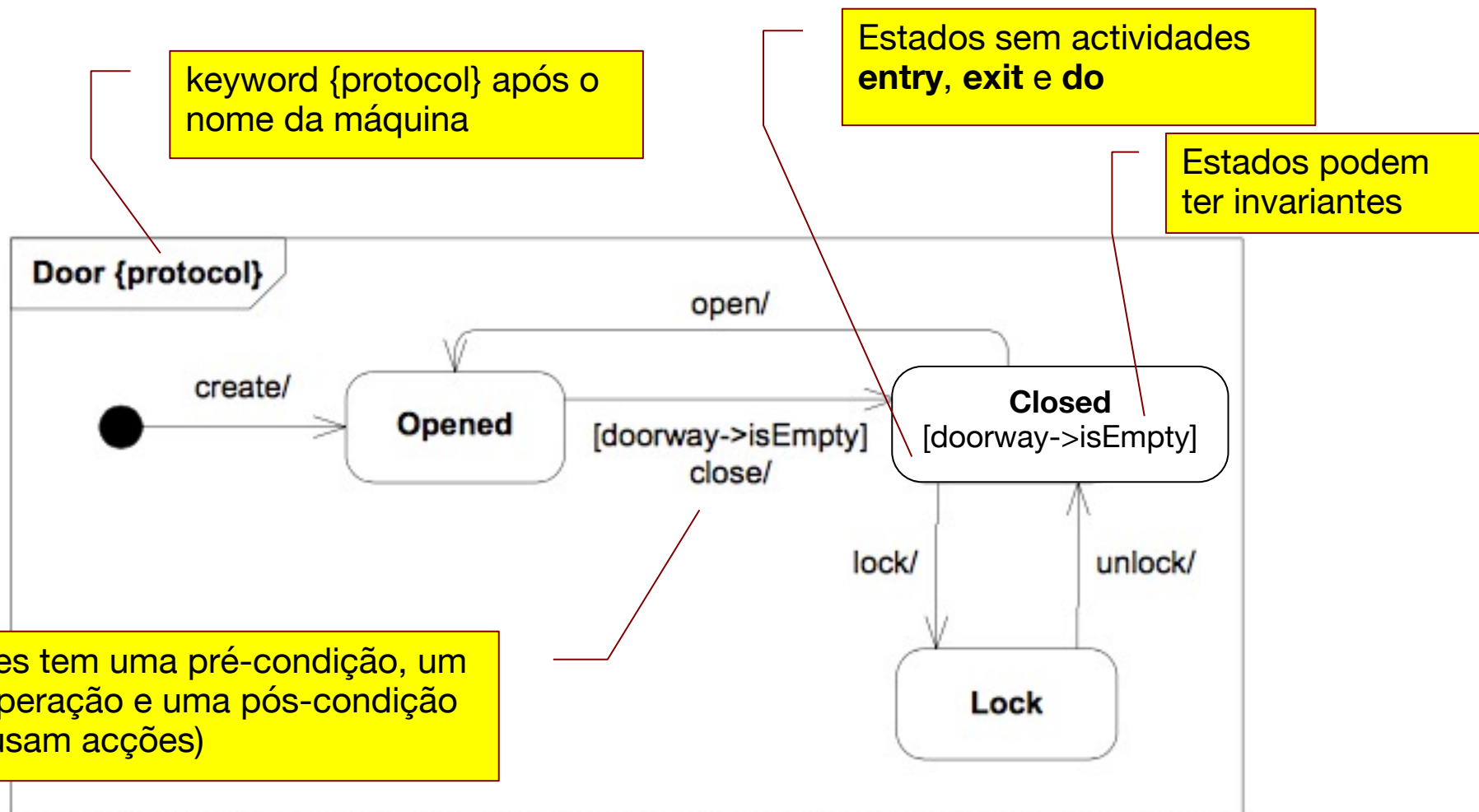
Resumo da notação (até agora)

	Estado
	Estado composto
	Estado submáquina
	Pseudo-estado inicial
	Estado final
	Transição (evento [condição] / acção) (entre estados vs. para o próprio estado vs. locais)
	Pseudo-estado de escolha
	Pseudo-estados de história (shallow/deep)



Protocol State Machines

- Especificam que operações podem ser invocadas em cada estado e em que condições - a sequências válidas de invocação das operações.





Modelação do controlo de diálogo da interface

Use Case: Reservar armação e lentes

Descrição: Funcionário regista uma reserva de armação e lentes

Pós-condição: Reserva fica registada

Fluxo normal:

1. Funcionário indica nome e/ou data de nascimento do cliente
2. Sistema procura clientes
3. Sistema apresenta lista de clientes
4. Funcionário selecciona cliente
5. Sistema procura cliente
6. Sistema apresenta detalhes do cliente
7. Funcionário confirma cliente
8. Sistema procura produtos e apresenta lista
9. Funcionário indica Código de armação e lentes
10. Sistema procura detalhes dos produtos
11. Sistema apresenta detalhes dos produtos
12. Funcionário confirma produtos
13. Sistema regista reserva dos produtos
14. <<include>> imprimir talão

Fluxo de excepção: [cliente não quer produto] (passo 12)

- 11.1. Funcionário rejeita produtos
- 11.2. Sistema termina processo

Registo de receitas

Nome Cliente: Data Nasc.:

202 x 68

Armações... Lentes Esq./Dir.

175 x 124

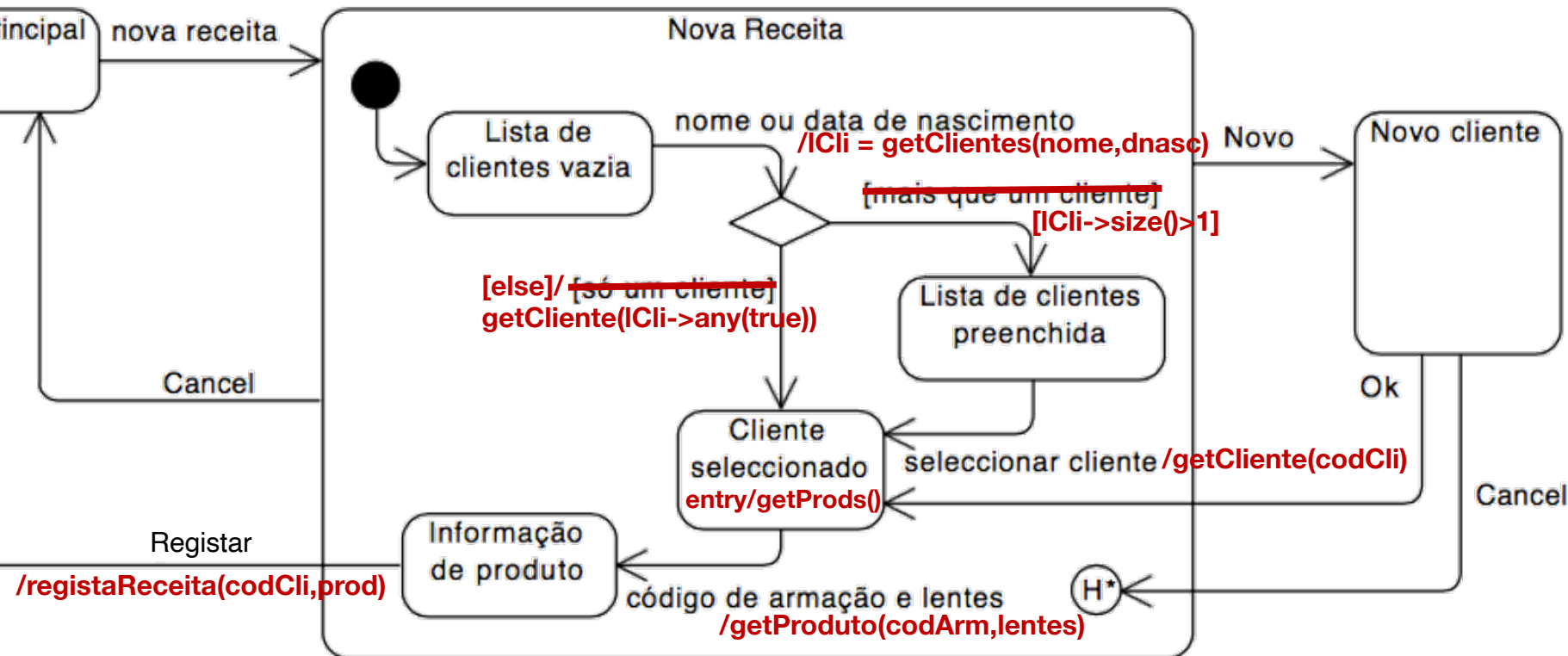
Modelação do controlo de diá

Registo de receitas

Nome Cliente: Data Nasc.:

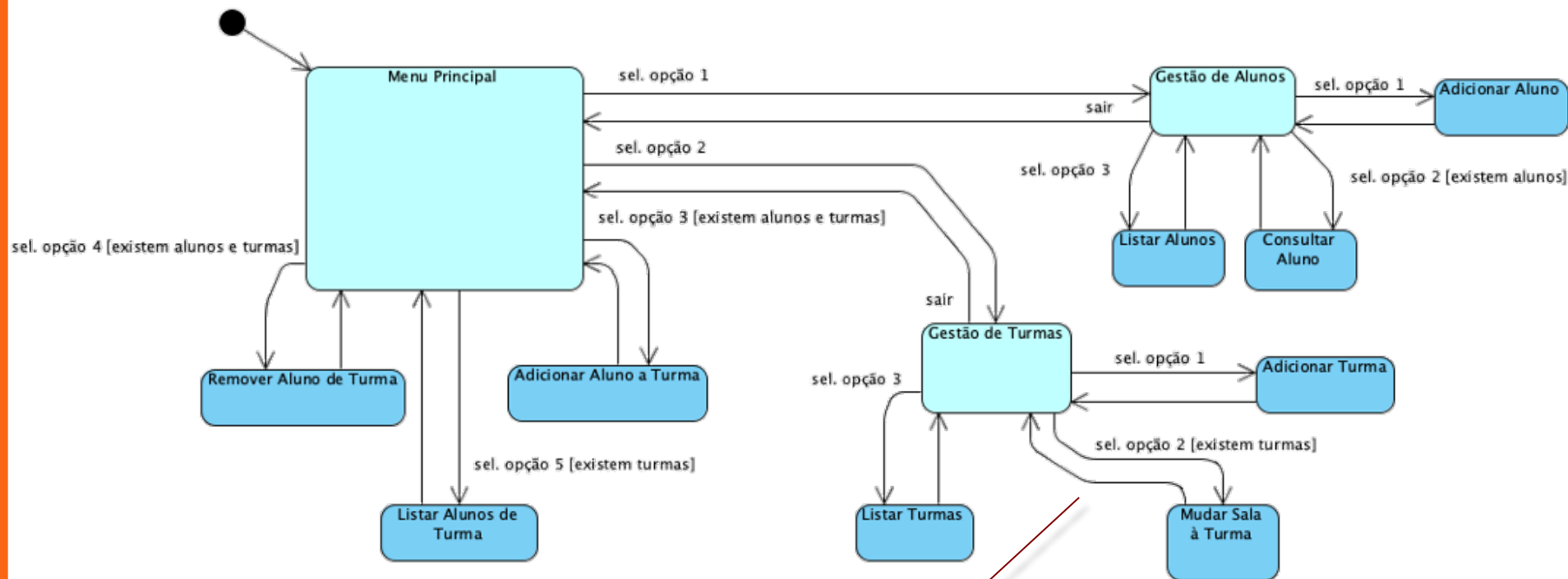
Armações...

Lentes Esq./Dir.





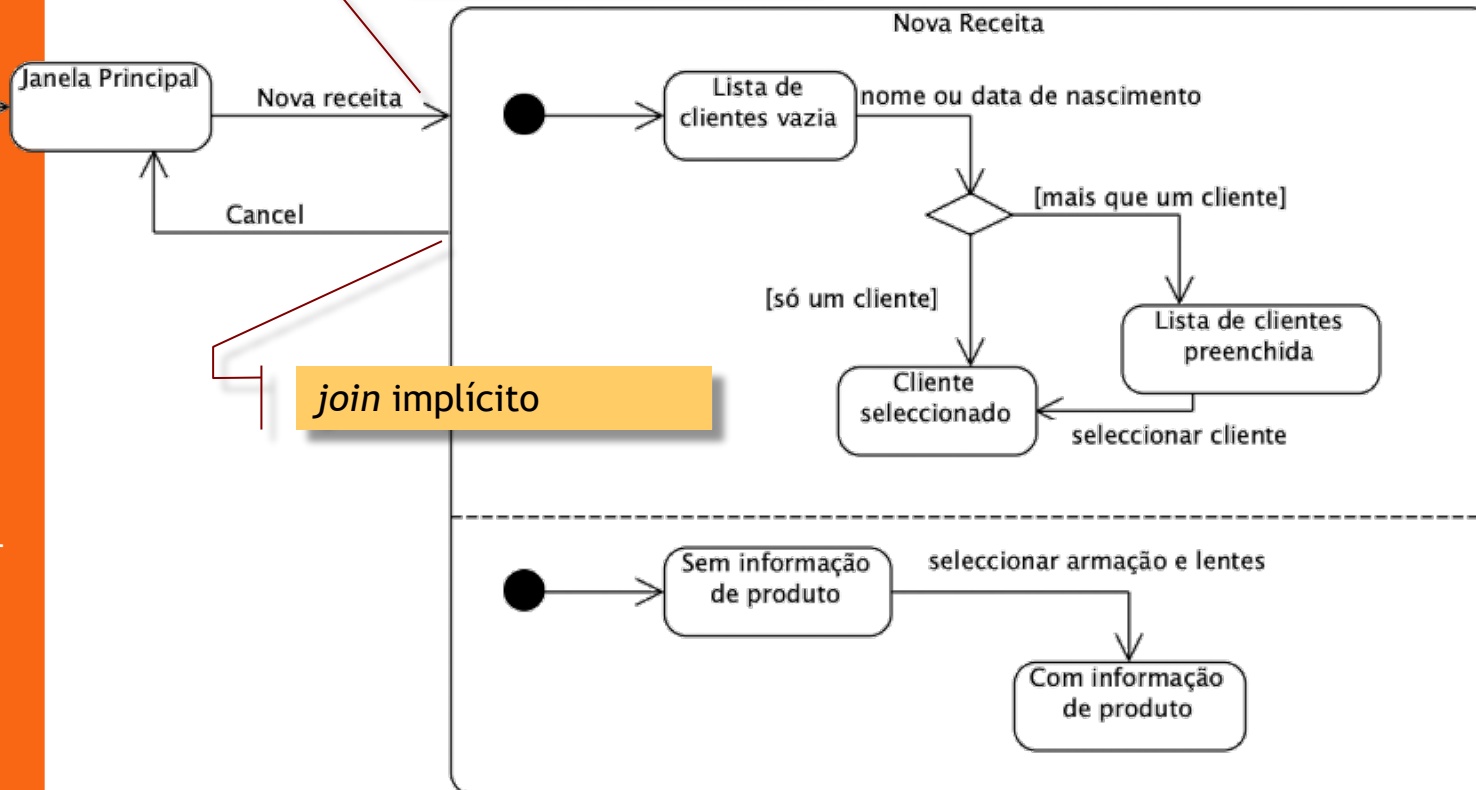
Mesmo em modo texto...



Completion transition
 Não tem evento, dispara automaticamente logo que possível.

Estados com concorrência...

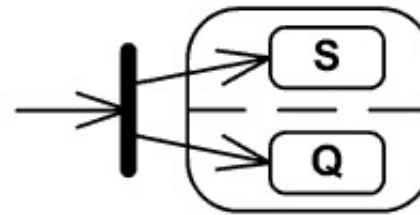
- Um estado pode ser dividido em “regiões” ortogonais
- Cada região contém um sub-diagrama
- Os diagramas das regiões são executados de forma concorrente



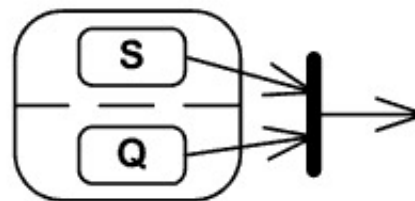


Pseudoestados *fork* e *join*

- Permitem gerir concorrência.
- Fork - divide uma transição de entrada em duas ou mais transições
 - Transições de saída têm que terminar em regiões ortogonais distintas



- Join - funde duas ou mais transições de entrada numa só transição de saída
 - Transições de entrada têm que originar em regiões ortogonais distinta

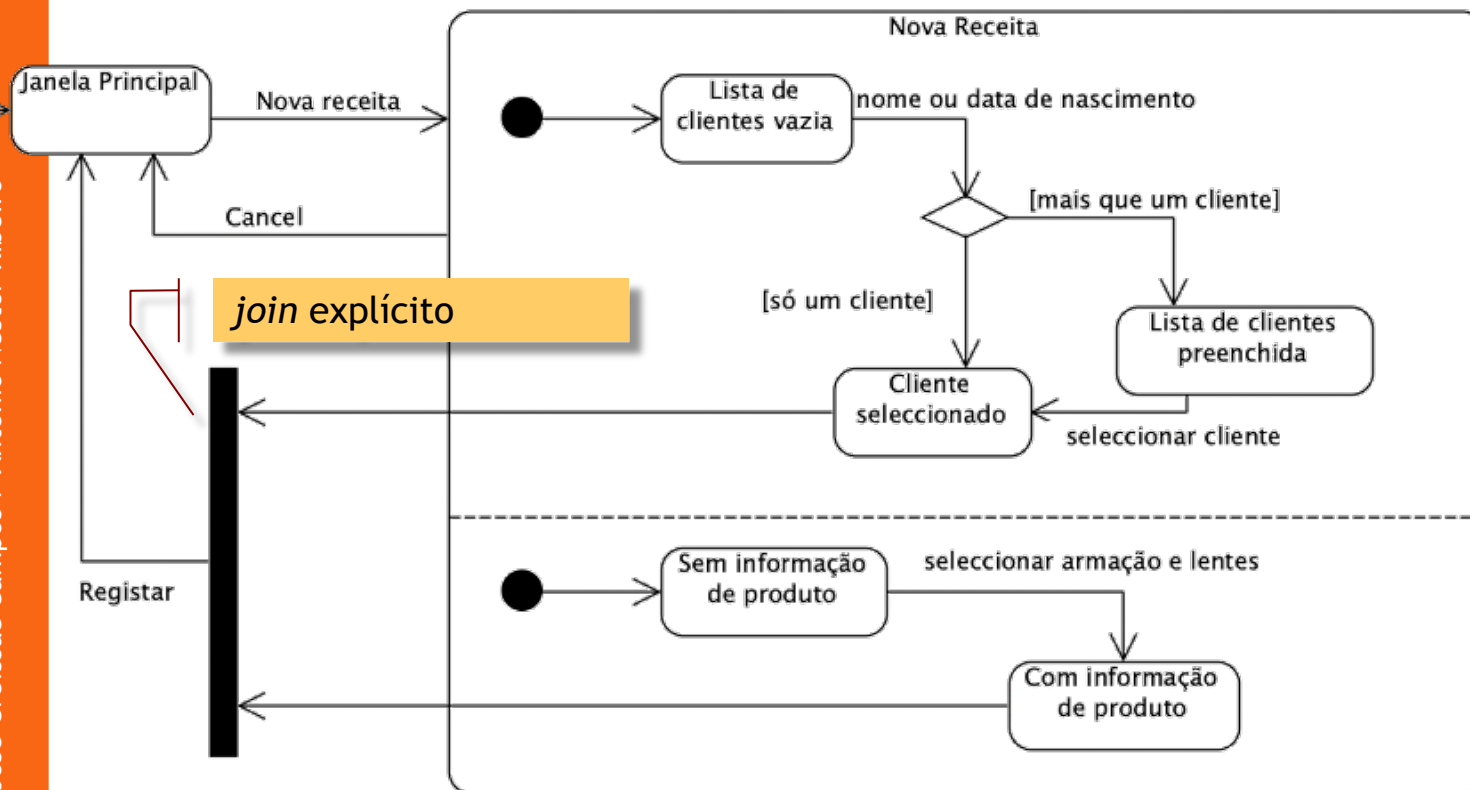


Estados com concorrência...

Registo de receitas

Nome Cliente: Data Nasc.:

Armações... Lentes Esq./Dir.



Estados com concorrência...

Registo de receitas

Nome Cliente: Data Nasc.:

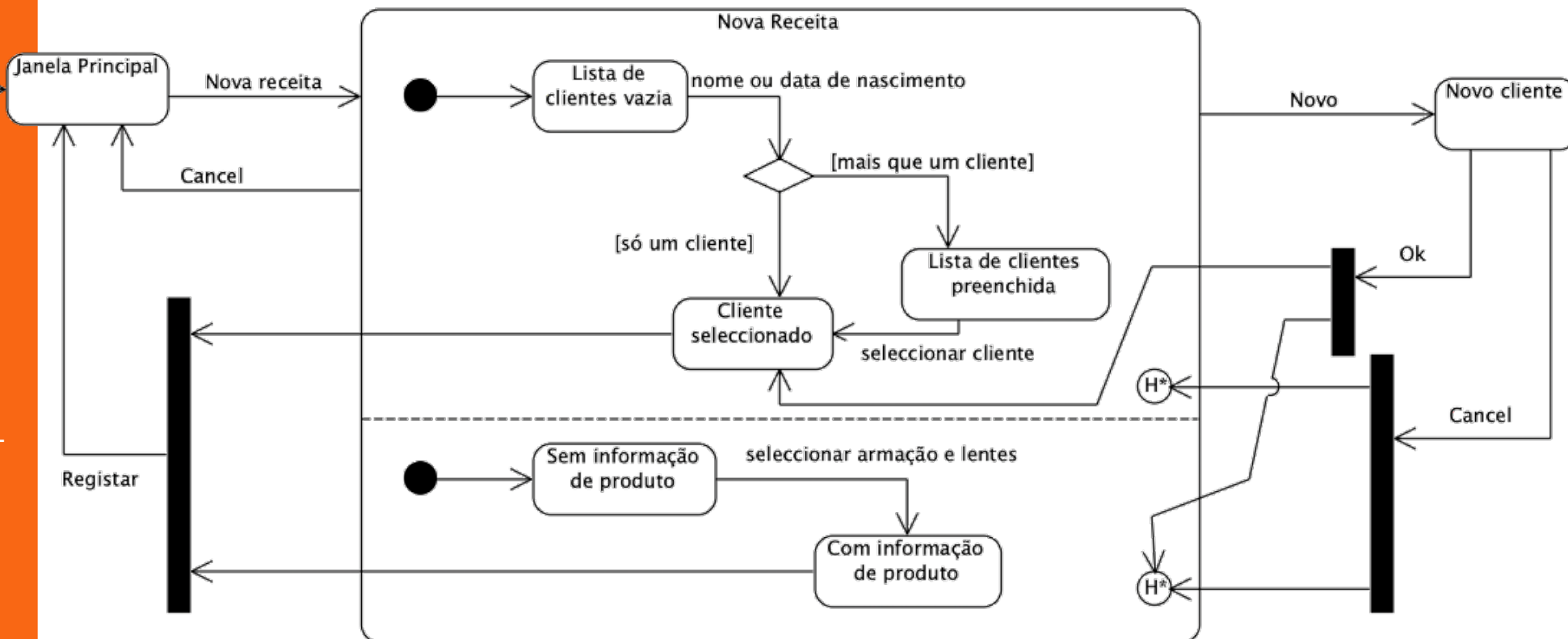
Armações...

Armação 1
Armação 2
Armação 3

Lentes Esq./Dir.



Lente
Lente

Lente
Lente

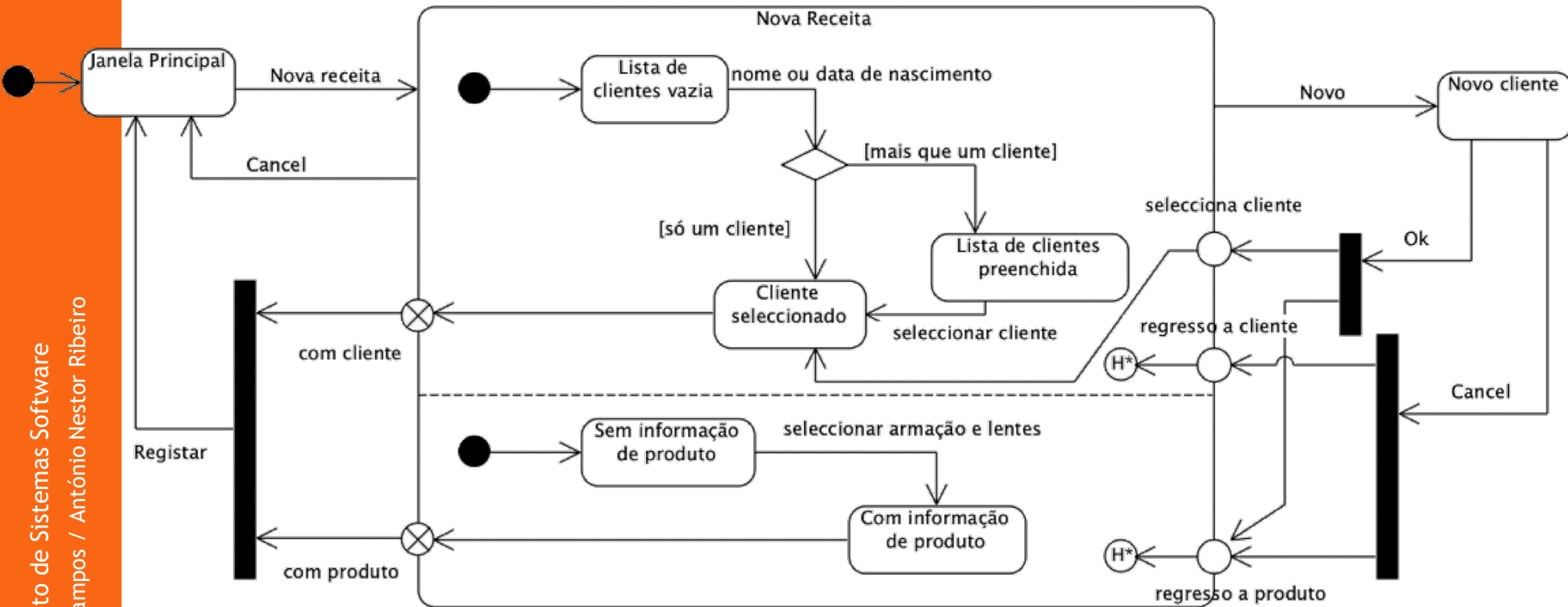




Pseudoestados Ponto de entrada e Ponto de saída

- Ponto de entrada 
 - Permite definir um ponto de entrada numa máquina de estados ou num estado composto
 - O ponto de entrada é identificado por nome
 - O ponto de entrada transita para um estado interno que poderá ser diferente do definido pelo estado inicial
- Ponto de saída 
 - Permite definir um ponto de saída alternativo ao estado final
 - O ponto de saída é identificado por nome

Desenvolvimento de Sistemas Software
José Creissac Campos / António Nestor Ribeiro



Desenvolvimento de Sistemas Software
José Creissac Campos / António Nestor Ribeiro

