Engenharia de Software

Modelo de Dinâmica
Parte 2

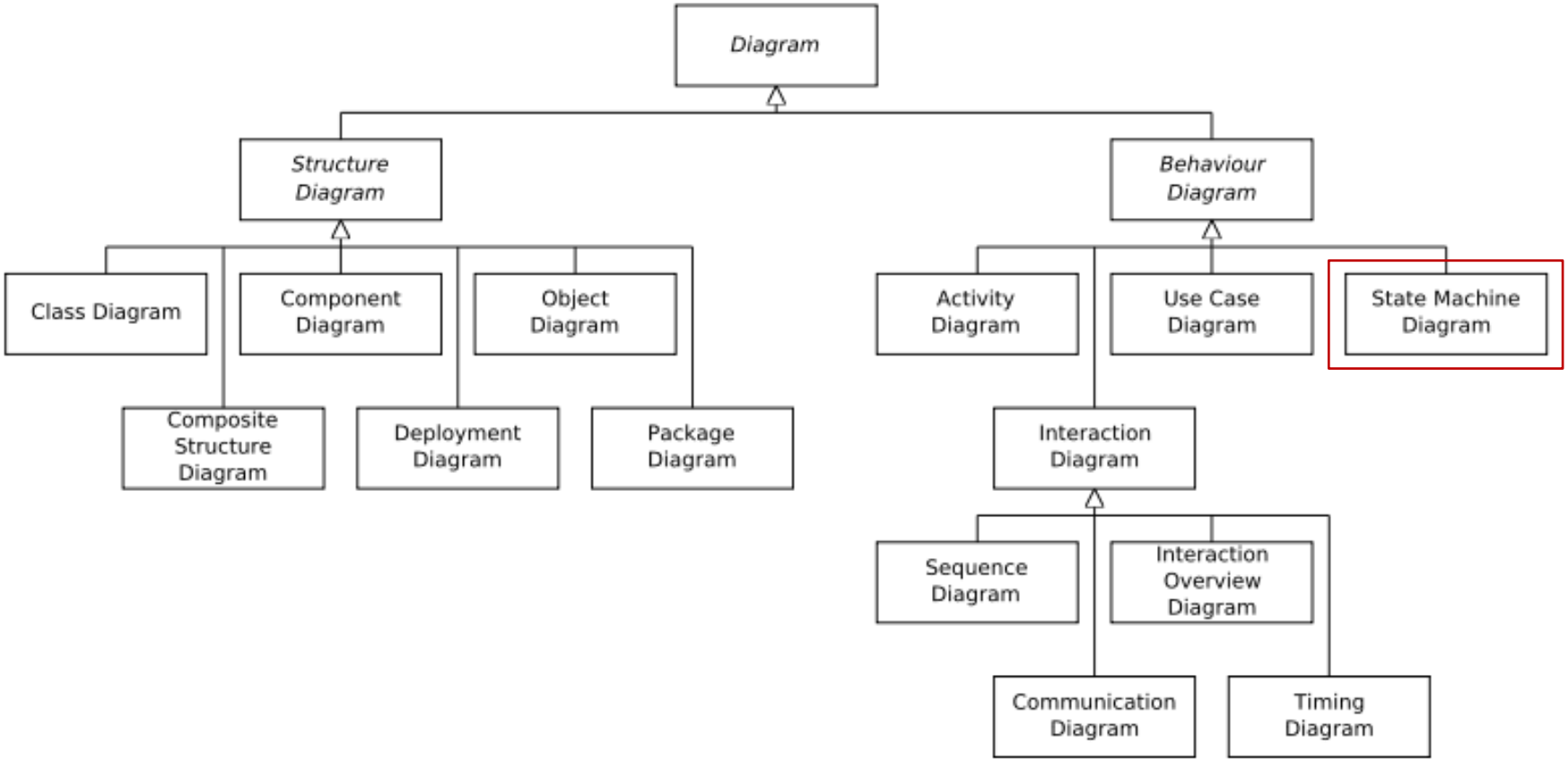
Luís Morgado

Instituto Superior de Engenharia de Lisboa

Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Linguagem UML

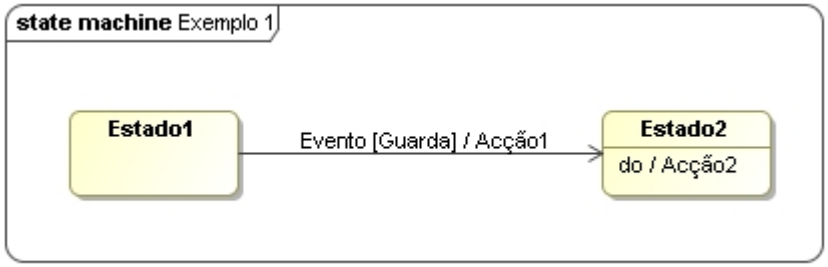
Perspectivas de modelação



[OMG, 2020]

Diagrama de Transição de Estado

- **Estado**
  - Representação de situação de evolução de um sistema ou parte de um sistema
- **Transição**
  - Acontecimento através do qual o sistema evolui do estado actual para um novo estado
  - **Evento**
    - Ocorrência no tempo e no espaço com significado para a evolução de estado
  - **Guarda**
    - Condição que inibe ou permite transições ou acções
      - Exemplo típico de utilização de variáveis de estado
- **Acção**
  - Define comportamento
    - Associado a **transição**
    - Associado a **estado**



Pseudo-estados

Símbolos utilizados com significado específico para definição de semântica adicional - vértices transientes do grafo de transição de estado

Início: Representa a fonte de transição inicial da máquina

Fim: Representa o destino para a transição final da máquina

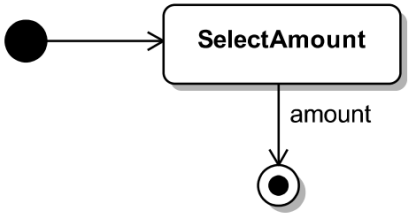
Histórico: Representa o sub-estado (superficial - *shallow*) ou estado (profundo - *deep*) mais recentemente activo de um estado composto

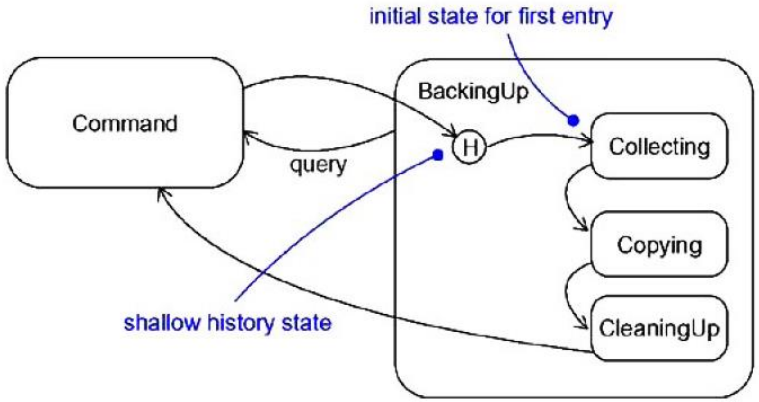
Bifurcação / Reunião: Representam a separação/reunião de transições envolvendo diferentes regiões ortogonais

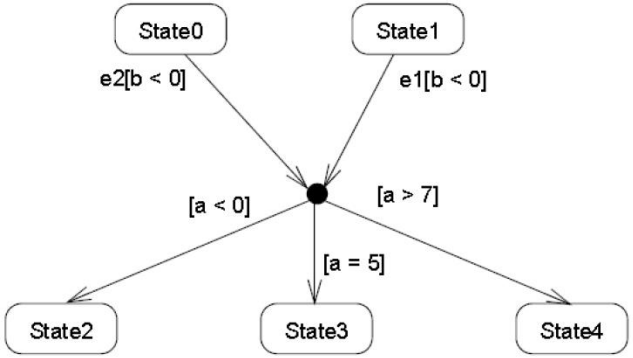
Junção: Suporte geral para relacionar diferentes transições sem introduzir semântica adicional

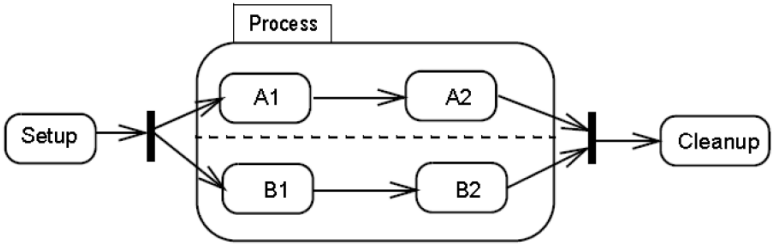
Pseudo-estados

Exemplos



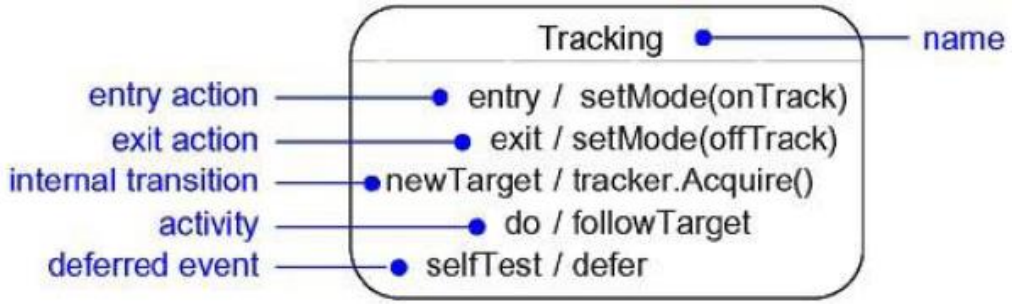






[OMG, 2020; UML Booch *et al.* 1998]

Processamento de Estado



[Booch *et al.* 1998]

- **entry**
  - Acção executada ao entrar no estado
- **exit**
  - Acção executada ao sair do estado
- **do**
  - Acção executada durante o estado (*actividade* de estado)
- **Transições internas**
  - Eventos processadas sem causarem transição de estado
- **Eventos diferidos**
  - Eventos que não são processados no estado, mas são adiados para serem processados noutro estado

Acções de Entrada e de Saída

Acções realizadas ao iniciar e finalizar um estado

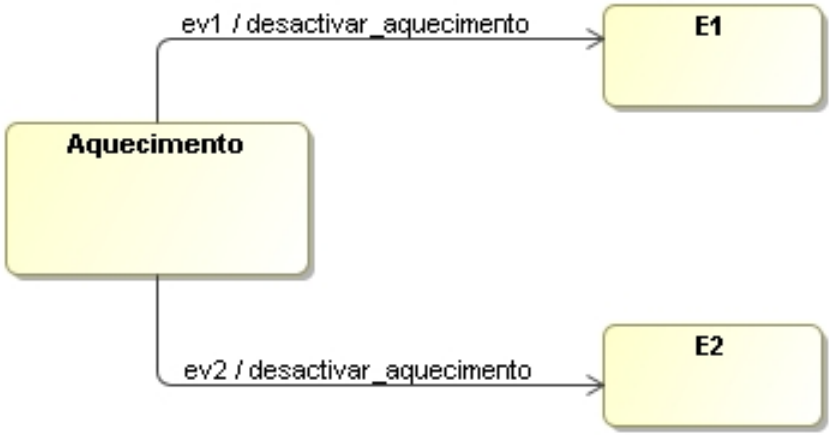
- ***entry***
  - Acções executadas após a entrada no estado
    - Representam comportamento que é executado imediatamente após a transição para o estado
- ***exit***
  - Acções executadas antes da saída do estado
    - Representam o comportamento que é executado imediatamente antes da transição para outro estado

Exemplo

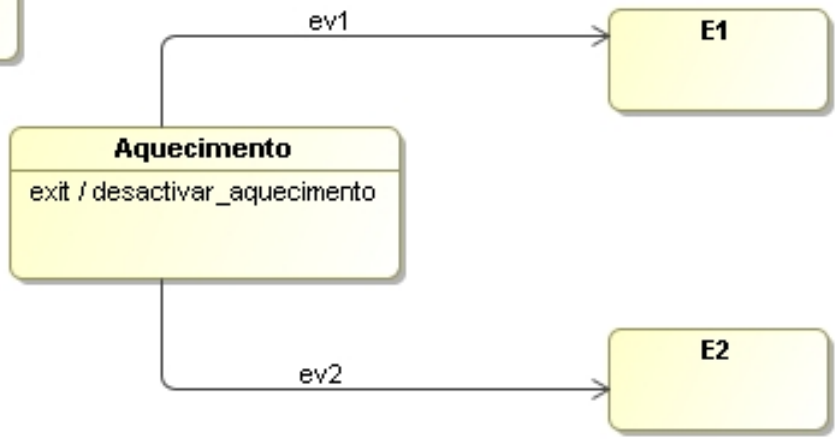
Acções de Entrada e de Saída

Exemplo: Forno de cozinha

- **Requisito crítico de segurança**: desactivar sistema de aquecimento sempre que exista um acesso ao forno



Implementação
- Mais simples
- Mais segura

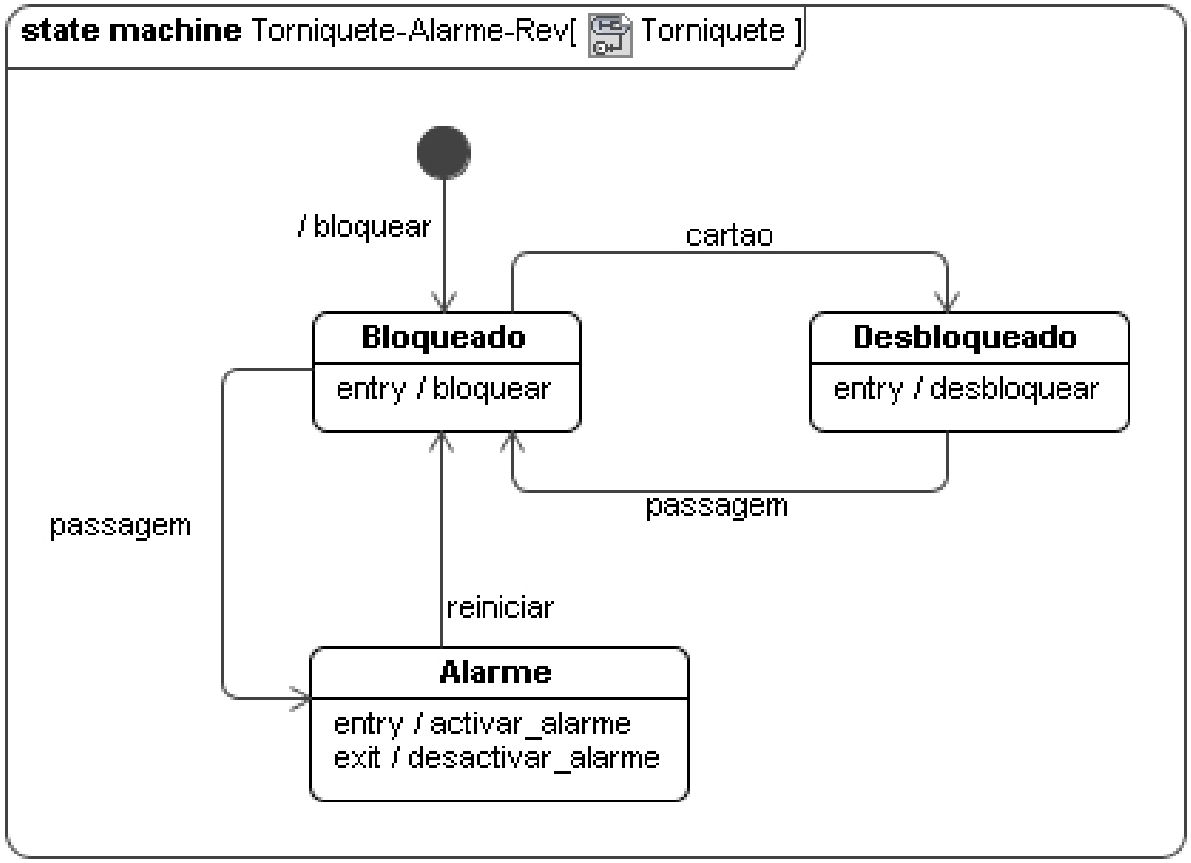


Repetição

- Complexidade de representação
- Potencial de erro por inconsistência de transições (e.g. nova transição sem desactivação de aquecimento)

Caso Prático

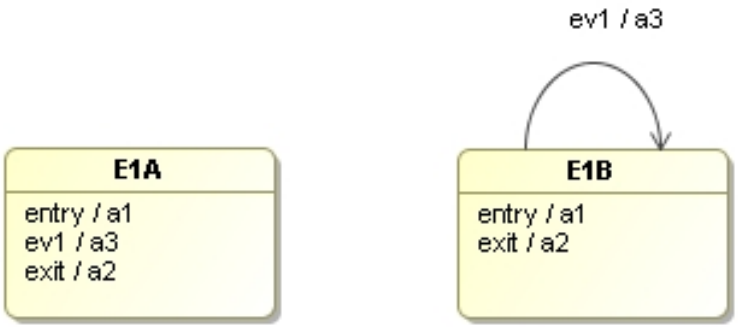
Torniquete de Controlo de Acessos



Transições Internas

- Eventos que produzem execução de acções sem causar transição de estado
  - **Transição interna**
  - Análogo a ***entry*** e ***exit*** mas associadas a eventos específicos
  - Caso não existam acções *entry* e *exit*, comportamento idêntico a auto-transições (estado destino = estado origem)

Exemplo



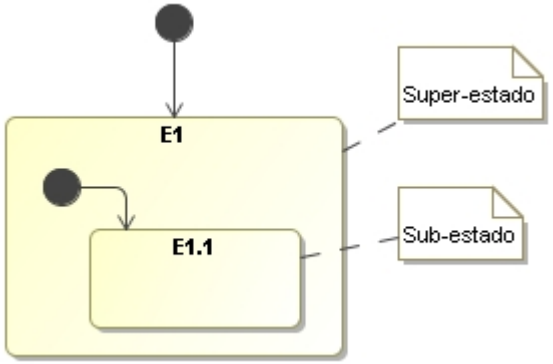
Acções realizadas quando ocorre ev1:	a3	a2, a3, a1

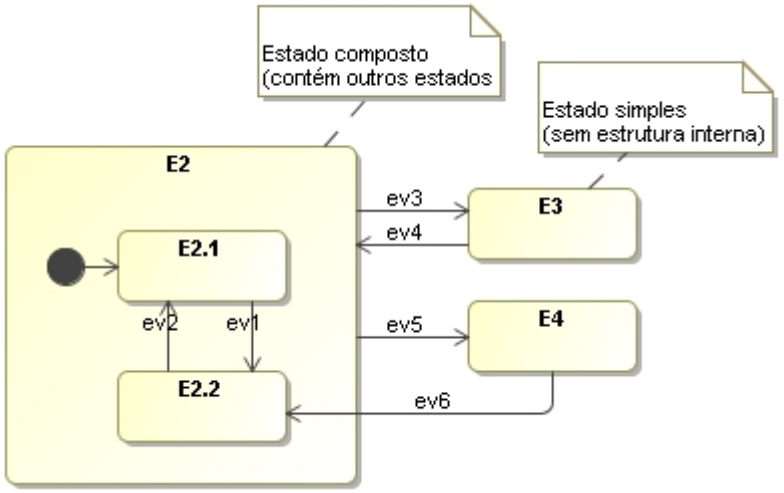
Máquinas de Estados Hierárquicas

- Factorização de comportamento
  - Subestados apenas necessitam de especificar as diferenças de comportamento em relação aos respectivos superestados
  - Reutilização de comportamento
    - Permite **abstrair o que é comum**
      - Automaticamente processado nos níveis superiores
    - Subestados **partilham comportamento** com o superestado
- Abstracção
  - **Controlo de complexidade**
    (redução selectiva de complexidade)
    - *Zoom IN/OUT*
  - Sem abstracção mesmo dinâmicas moderadamente complexas tornam-se difíceis de modelar e implementar

Máquinas de Estados Hierárquicas

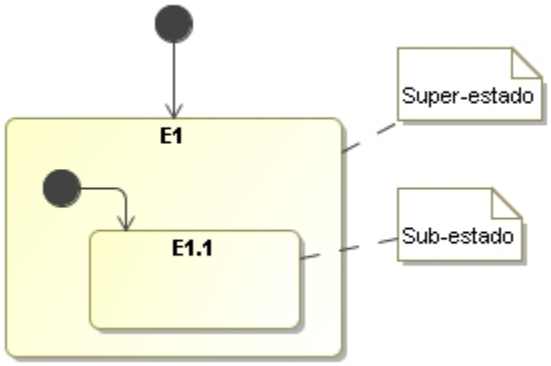
- Estruturação do modelo a diferentes níveis de abstracção





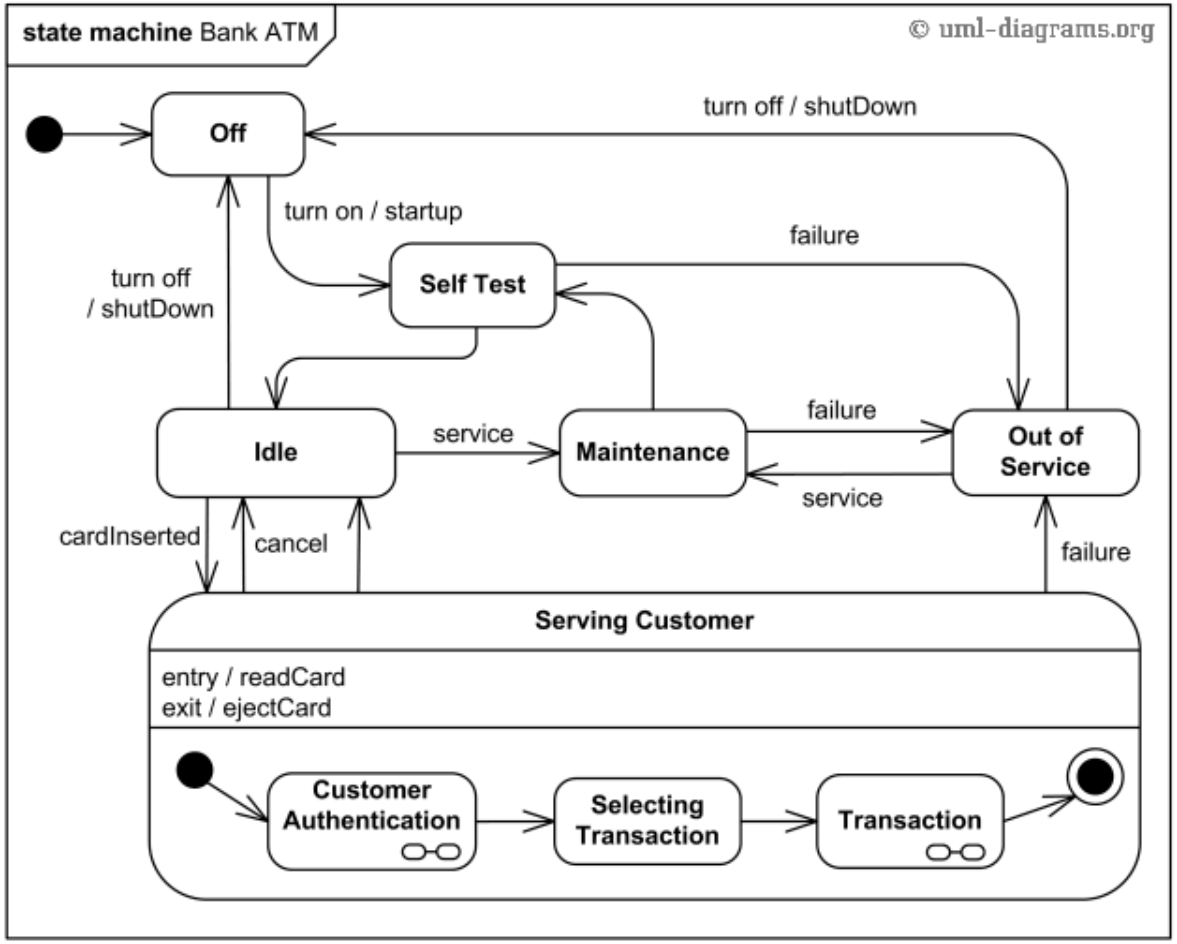
Máquinas de Estados Hierárquicas

- **Semântica**
  - Se um sistema está num subestado E1.1 também está (implicitamente) no respectivo superestado E1
    - **Estado composto**
  - Se o sistema está no subestado E1.1, o processamento dos eventos será feita nesse contexto.
    - Se não for definido um evento para E1.1 ?
      - Será processado no contexto do estado mais geral E1



Máquinas de Estados Hierárquicas

Exemplo

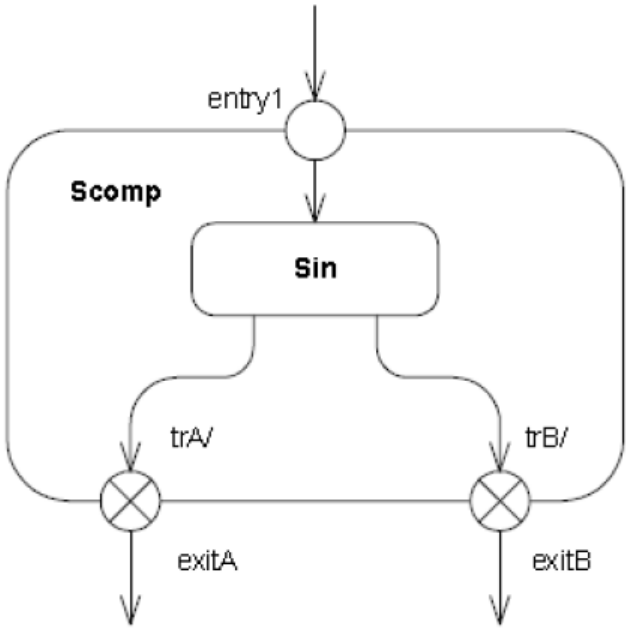


Máquinas de Estados Hierárquicas

Organização de sub-máquinas de estado

Pseudo-estados

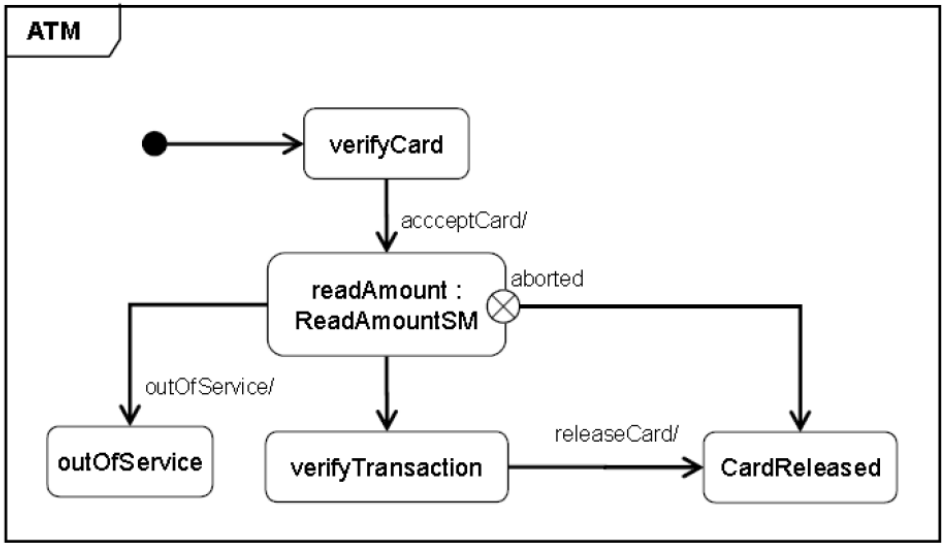
- **Entrada (*entry*)***:* Representa entrada num estado composto
- **Saída (*exit*)***:* Representa saída de um estado composto

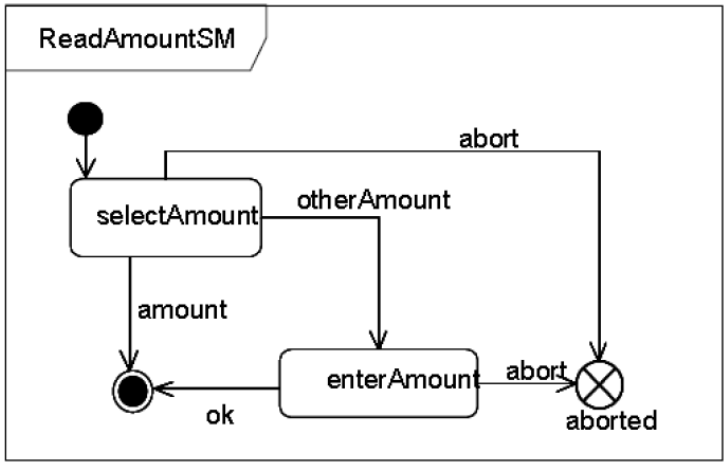


[OMG, 2020]

Máquinas de Estados Hierárquicas

Exemplo





[OMG, 2020]

Caso Prático: Relógio-Agenda

Especificação suplementar

R7	A comutação entre os modos do relógio ocorre de acordo com a seguinte sequência cíclica: Hora ⇨ Data ⇨ Cronómetro	Obrigatório

Caso de Utilização: *Visualizar Modo*

A comutação entre modos ocorre através do botão MODE

Descrição de Caso de Utilização: *Simulador Relógio-Agenda*

Designação: **Visualizar Modo**

Resumo: Este caso de utilização permite ao utilizador visualizar a hora, a data ou o cronómetro.

Actores: Utilizador

Pré-condições:

Cenário principal:

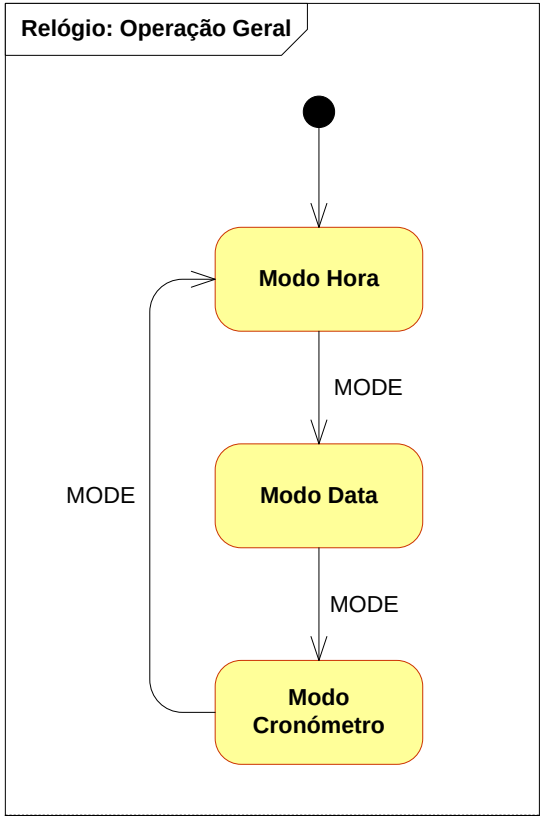
1. O caso de utilização inicia-se quando o utilizador pretende visualizar a hora, a data ou o cronómetro, estando o relógio no modo respectivo.
2. O utilizador observa a informação pretendida e o caso de utilização termina.

Cenário alternativo 1:

1. No passo 1 do cenário principal o sistema está em modo relógio mas não está no modo pretendido.
2. O utilizador pressiona o botão MODE as vezes necessárias para colocar o sistema no modo pretendido (**ver requisito R7**).
3. O utilizador observa a informação e o caso de utilização termina.

Cenário alternativo 2:

1. No passo 1 do cenário principal, o sistema está em modo agenda.
2. O utilizador pressiona o botão MEMO.
3. O sistema retorna ao modo relógio (**ver requisito R8.2**).
4. Aplica-se o cenário anterior adequado.



Caso Prático: Relógio-Agenda

Casos de Utilização: *Acertar Hora, Acertar Campo*

Descrição de Caso de Utilização: *Simulador Relógio-Agenda*

Designação: **Acertar Hora**
Resumo: Este caso de utilização permite ao utilizador acertar a hora do relógio.
Actores: Utilizador
Pré-condições: O sistema encontra-se em modo de visualização de hora.

Cenário principal:

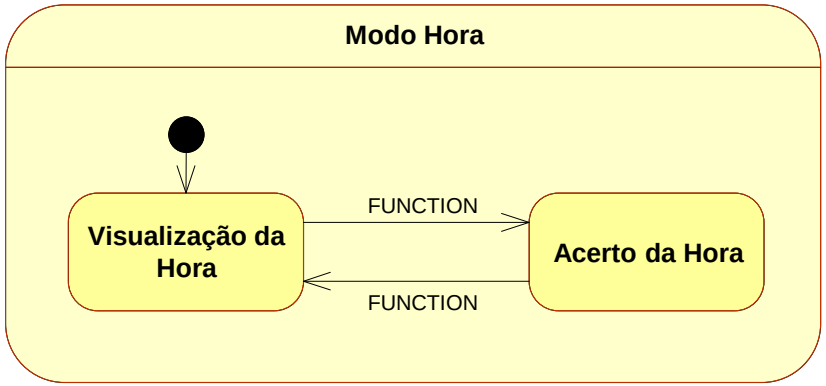
1. O caso de utilização inicia-se quando o utilizador pretende acertar a hora estando o relógio em modo hora.
2. O utilizador pressiona o botão FUNCTION.
3. O campo SEGUNDOS do relógio fica a piscar.
4. **Incluir Acertar Campo.**
5. O utilizador pressiona o botão MODE.
6. O campo SEGUNDOS deixa de piscar e passa a piscar o campo MINUTOS.
7. **Incluir Acertar Campo.**
8. O utilizador pressiona o botão MODE.
9. O campo MINUTOS deixa de piscar e passa a piscar o campo HORAS.
10. **Incluir Acertar Campo.**
11. O utilizador pressiona o botão FUNCTION.
12. O campo HORAS do relógio deixa de piscar e o caso de utilização termina.

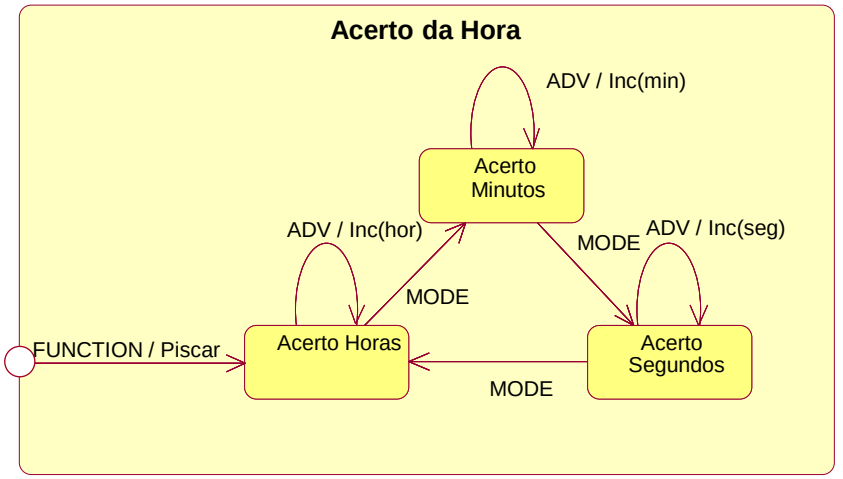
Descrição de Caso de Utilização: *Simulador Relógio-Agenda*

Designação: **Acertar Campo**
Resumo: Este caso de utilização permite ao utilizador acertar um campo de hora ou data.
Actores: Utilizador
Pré-condições: O sistema encontra-se em modo de visualização de hora ou de data com um campo seleccionado (a piscar).

Cenário principal:

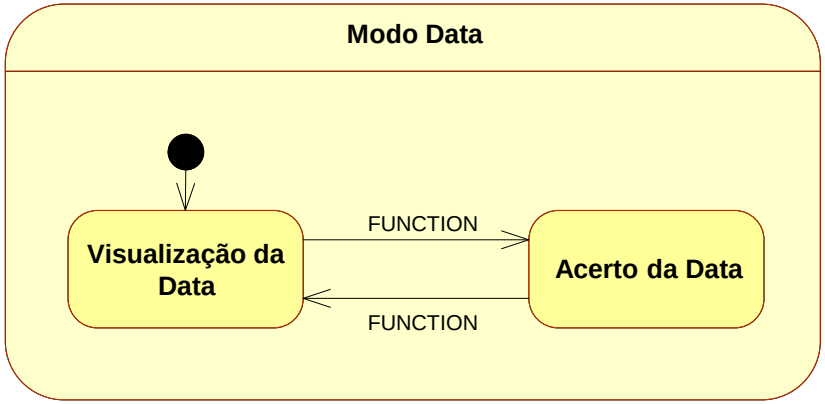
1. O caso de utilização inicia-se quando o utilizador pressiona o botão ADV.
2. Enquanto o campo não apresentar o valor pretendido pelo utilizador:
   a) O sistema incrementa o campo seleccionado (**ver requisitos R2 e R5**).
3. O caso de utilização termina.

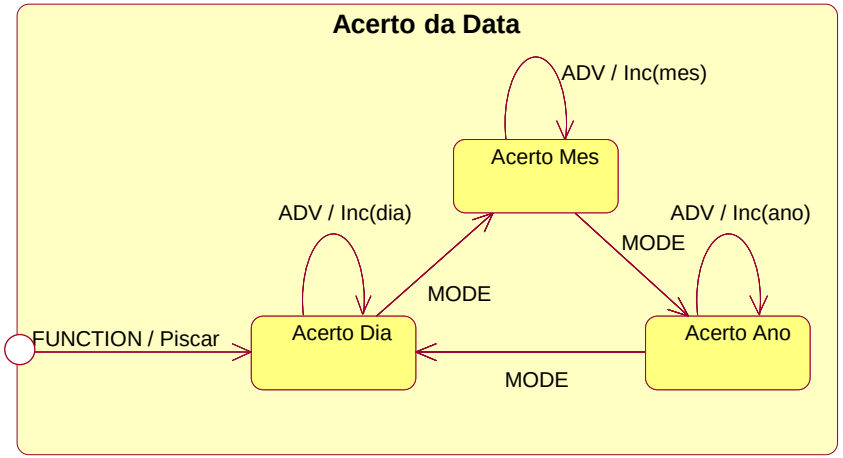




Caso Prático: Relógio-Agenda

Caso de Utilização: *Acertar Data*





Caso Prático: Relógio-Agenda

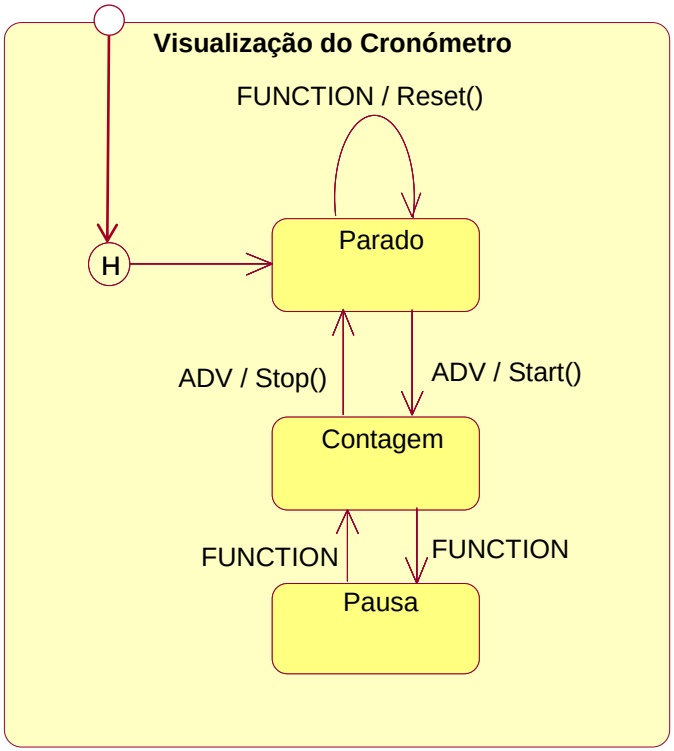
Caso de Utilização: *Utilizar Cronómetro*

Descrição de Caso de Utilização: *Simulador Relógio-Agenda*

Designação:	**Utilizar Cronómetro**
Resumo:	Este caso de utilização permite ao utilizador cronometrar uma actividade.
Actores:	Utilizador
Pré-condições:	O sistema está em modo de cronómetro com o cronómetro parado.

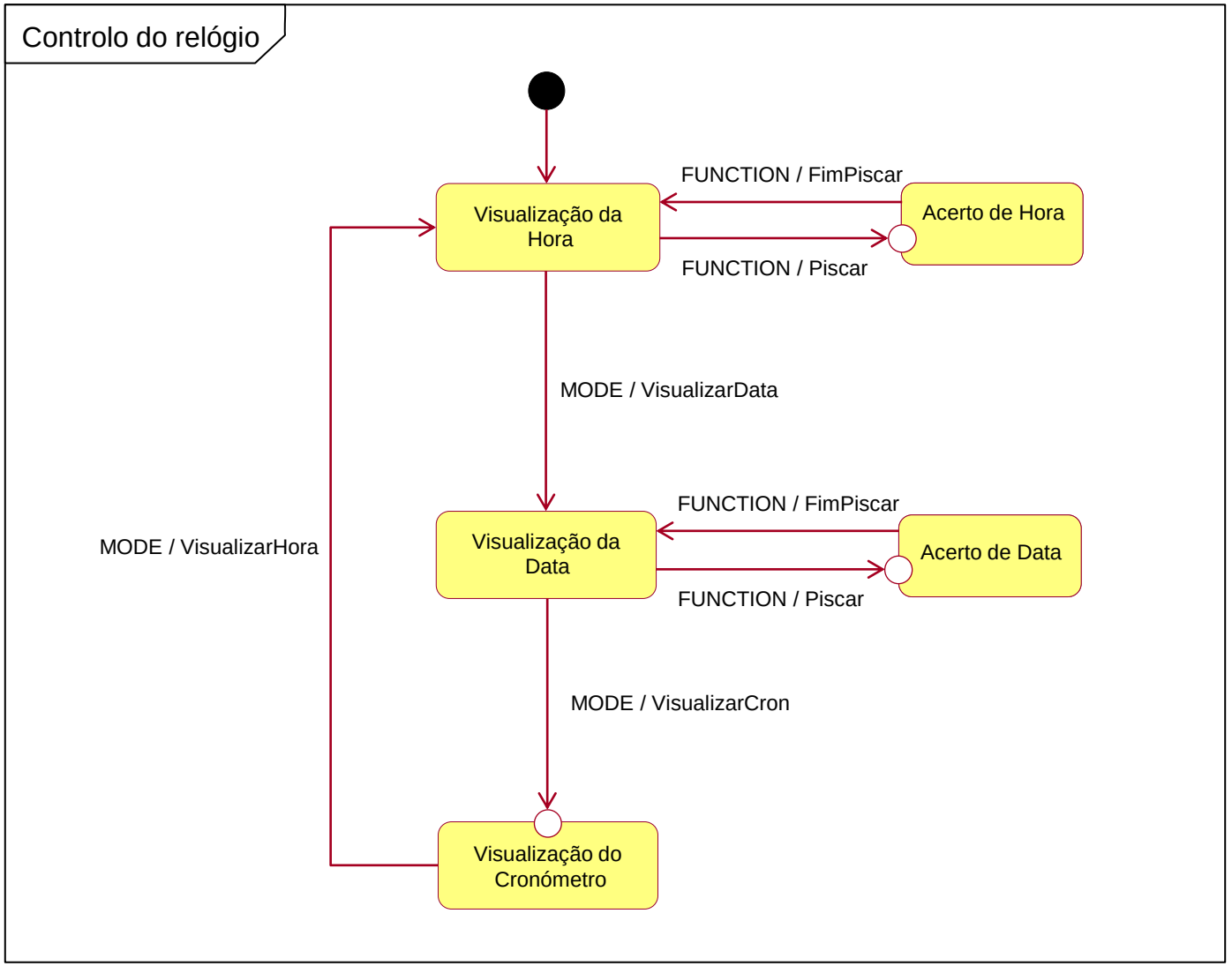
Cenário principal:

1. O caso de utilização inicia-se quando o utilizador pressiona o botão ADV.
2. O cronómetro inicia a contagem **(ver requisito R1)**.
3. O utilizador pressiona o botão FUNCTION.
4. O cronómetro mantém o tempo parcial.
5. Internamente o cronómetro continua a contar mas não é feita a actualização dos campos.
6. Após algum tempo o utilizador pressiona novamente o botão FUNCTION.
7. O cronómetro retoma a actualização dos campos de acordo com a contagem interna.
8. Após algum tempo o utilizador pressiona novamente o botão ADV.
9. O cronómetro pára.
10. O utilizador pressiona o botão FUNCTION (função *Reset*).
11. O sistema coloca todos os campos do cronómetro a zero.
12. O caso de utilização termina.



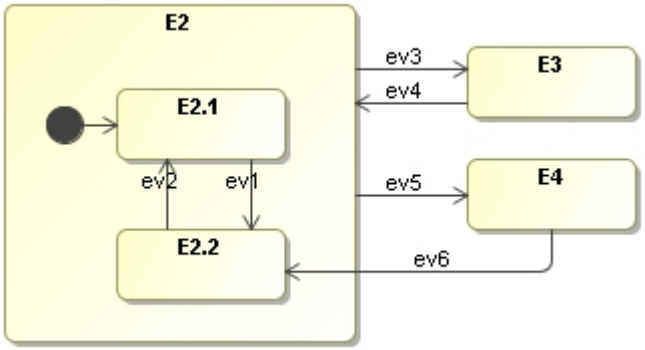
Caso Prático: Relógio-Agenda

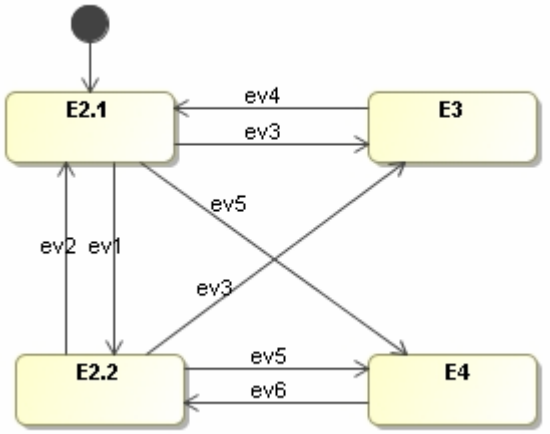
Dinâmica global do controlo do relógio



Máquinas de Estados Hierárquicas

- Planificação
  - Conversão de uma máquina de estados hierárquica numa máquina de estados plana (não hierárquica)
    - Pode ser útil para efeitos de implementação, mas requer desmultiplicação de transições





- As *máquinas de estados hierárquicas* não são apenas uma forma de representação visual, são um método de reutilização comportamental
  - **Factorização comportamental**
  - Ao **evitar repetições (redundância)** permite o crescimento da complexidade do sistema a descrever sem que isso provoque um **aumento exponencial da complexidade do modelo**

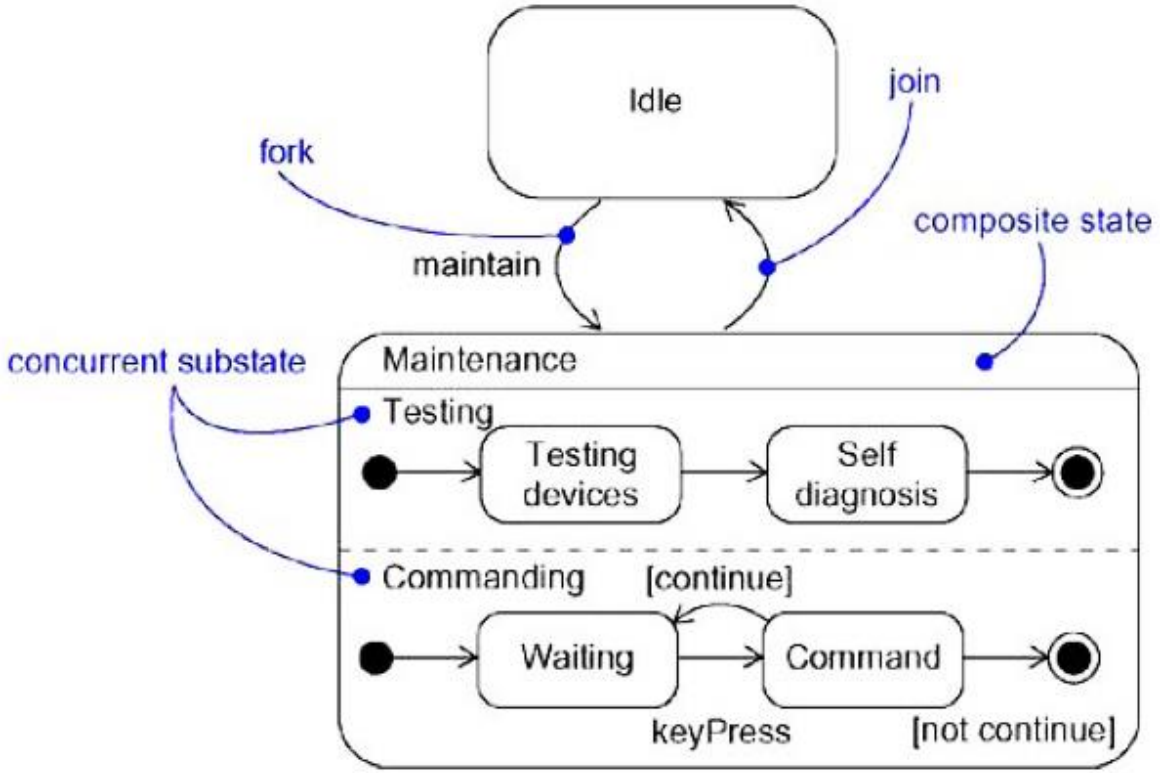
Regiões Ortogonais

- Decomposição hierárquica de máquinas de estados
  - Decomposição disjuntiva (***OR***)
    - Exemplo: O sistema está no estado E2.1 **ou** no estado E2.2
  - Decomposição conjuntiva (***AND***)
    - Duas ou mais *regiões ortogonais* (independentes)
    - Um sistema está em todos os estados ortogonais em simultâneo
    - As *regiões ortogonais* definem submáquinas com execução independente que pode acontecer em simultâneo, de forma concorrente
    - Permite lidar com o problema do aumento combinatório de estados em sistemas compostos por partes independentes que operam em concorrência

Regiões Ortogonais

Exemplo

Estado composto termina quando termina a última máquina concorrente (região ortogonal)



[Booch *et al.* 1998]

Regiões Ortogonais

- As regiões ortogonais (realizadas por sub-máquinas de estados) podem não ser totalmente ortogonais
  - Pode existir *interacção* entre regiões, por exemplo, coordenação de comportamento através da troca de eventos
  - Pode haver necessidade de *sincronização*
- Linguagem UML
  - Não requer:
    - Execução independente (*thread*) para cada região ortogonal (apesar de poder ser assim implementado)
  - Requer:
    - Não assumir uma ordem específica de processamento de eventos entre regiões ortogonais

Sequência de Transição de Estado

Máquinas de estados hierárquicas

- Um sistema pode estar em diferentes estados
  - Todos os estados da **hierarquia**
  - Estados das **regiões ortogonais**
- Estado composto definido por uma árvore de estados
  - Estado global do sistema
    - Corresponde a uma *configuração de estado*

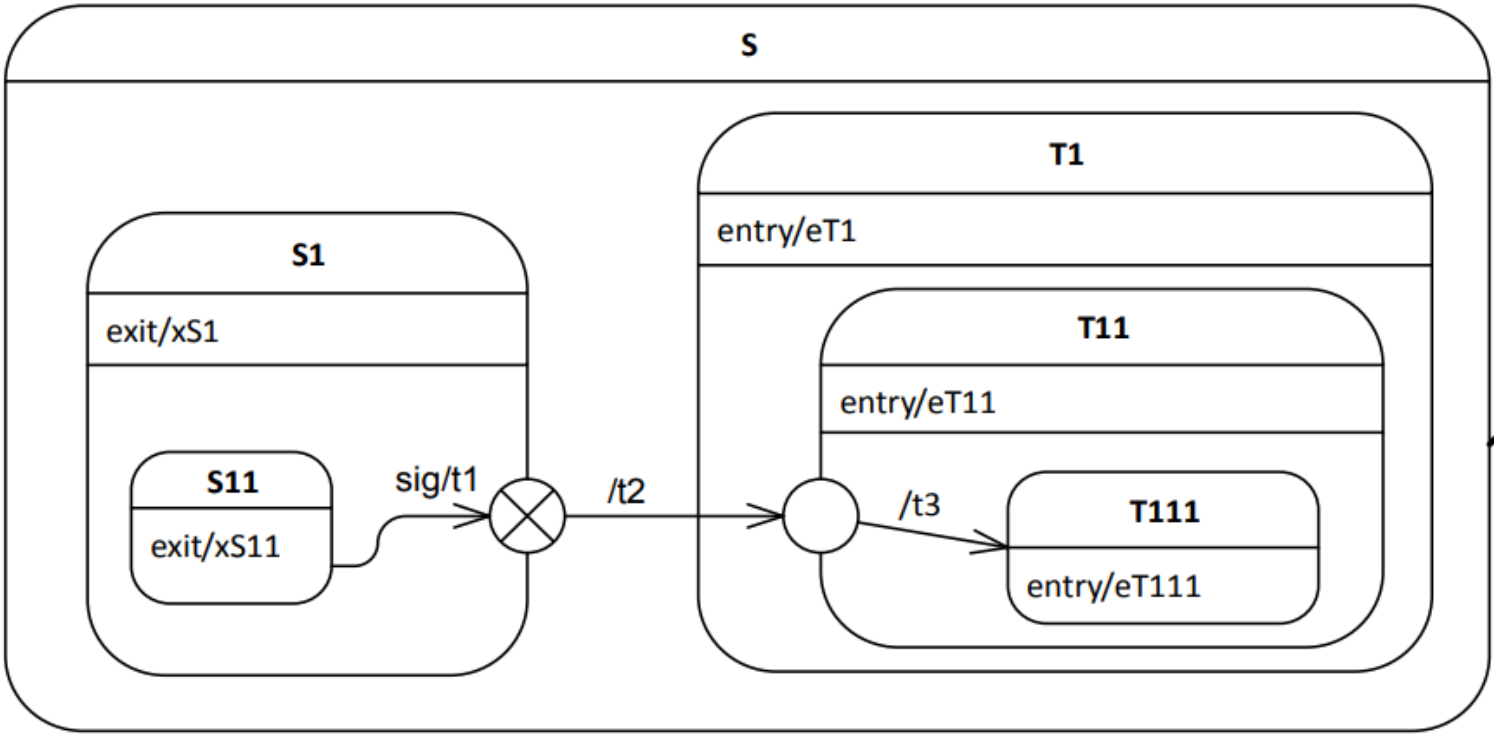
- Numa transição de estado de um estado composto, executar:
  - Acções de saída da **configuração de estado** origem (*exit*)
  - Acções associadas à transição
  - Acções de entrada da **configuração de estado** destino (*entry*)

Sequência de Transição de Estado

Exemplo

Quando o evento **sig** ocorrer enquanto a máquina de estados estiver no estado **S11**, será executada a seguinte sequência de acções:

xS11, **t1**, **xS1**, **t2**, **eT1**, **eT11**, **t3**, **eT111**



[OMG, 2020]

Acções de Entrada e de Saída

- Sequência de activação
  - A activação de acções de entrada (***entry***) deve acontecer do estado mais **exterior para o mais interior**
  - A activação de acções de saída (***exit***) deve acontecer na ordem inversa, do estado mais **interior para o mais exterior**

Modelo de Execução de Maquinas de Estados

- Na prática as acções não são instantâneas
- Duas situações de operação de uma máquina de estados
  - *Inactiva*
  - *Activa*
- O que acontece se ocorrer um evento enquanto uma máquina está *activa*, ou seja, enquanto está a decorrer o processamento de outro evento?
- Gestão de eventos
  - *Preemptiva*
    - Interrupção de processamento na ocorrência de um evento
    - Potenciais problemas de concorrência
  - Não *preemptiva*
    - Executar processamento sem interrupção

Processamento de Eventos

Tipos específicos de eventos na linguagem UML

- **Sinal**
  - Representa a recepção de um sinal assíncrono
  - *<nome-sinal> (<lista-parâmetros>)*
- **Evento temporal**
  - Representa o expirar de um limite temporal
  - *AFTER <duração>*
- **Evento condicional**
  - Representa a satisfação de uma condição booleana específica
  - *WHEN <condição>*
- **Evento de evocação**
  - Representa a evocação síncrona de uma operação
  - *<nome-operação> (<lista-parâmetros>)*

Caso Prático

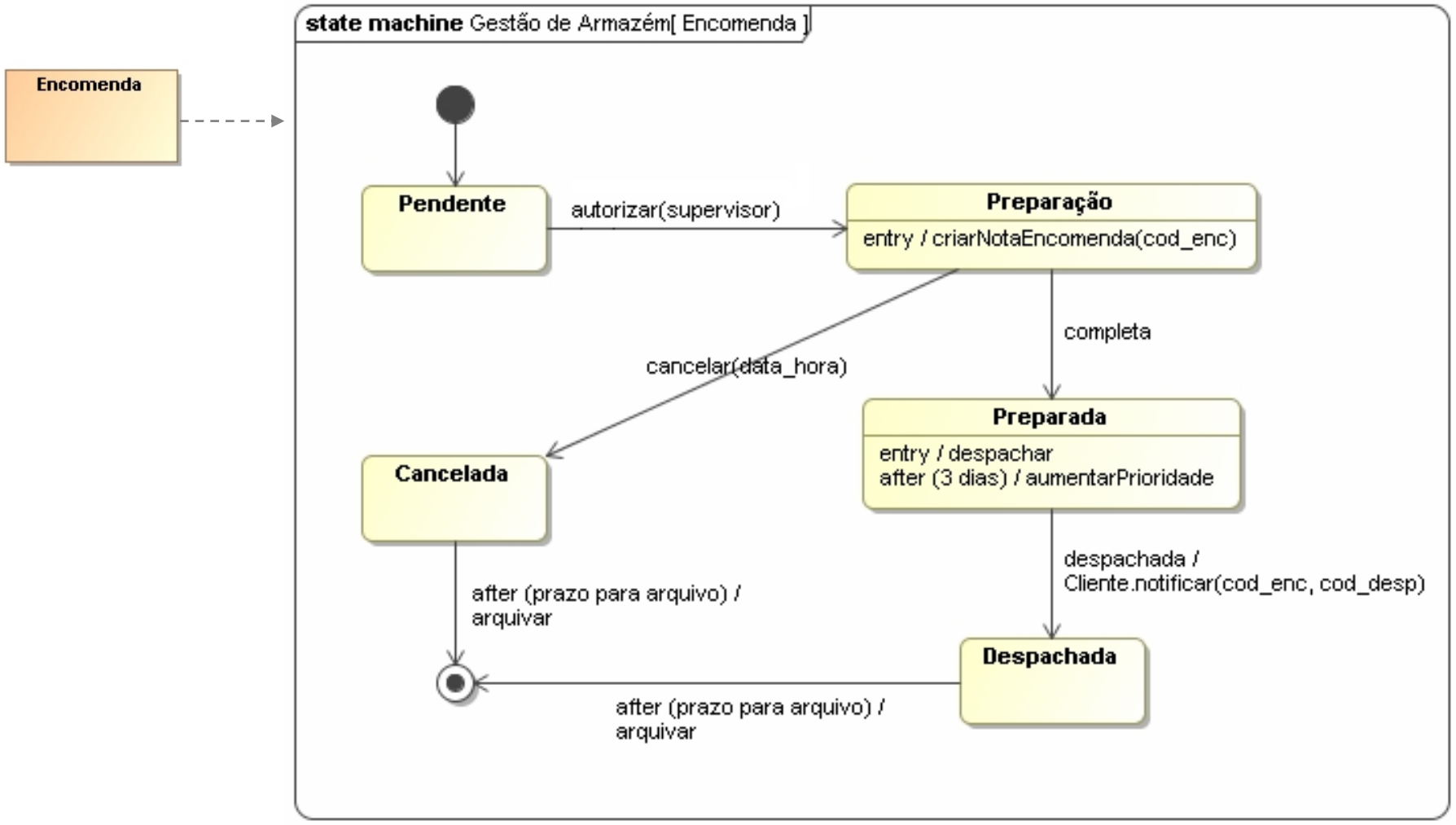
A empresa XYZ, proprietária de uma rede de lojas alimentares, pretende implementar um sistema que permita gerir a circulação de produtos nas suas várias lojas [...]

Os clientes podem fazer encomendas. Quando uma encomenda de um cliente é recebida fica pendente, até que um supervisor dê autorização para a sua realização. Deve ser mantido o registo do supervisor que autorizou cada encomenda. Após a autorização de um supervisor é preparada a encomenda. No início da preparação é sempre criada uma nota de encomenda com o código da encomenda.

Após a encomenda estar completa esta fica preparada para ser despachada para o cliente. O despacho pode no entanto não acontecer de imediato, pelo que, após 3 dias de espera por despacho, a prioridade da encomenda deve ser aumentada. Após o despacho o cliente deve ser notificado.

Uma encomenda pode ser cancelada enquanto está em preparação. Após o despacho ou o cancelamento, o registo da encomenda mantém-se durante um prazo predefinido. Após esse prazo a encomenda é arquivada e o seu registo eliminado do armazém.

Caso Prático



Bibliografia

[Booch et al., 1998]
G. Booch, J. Rumbaugh, I. Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley, 1998.

[Eriksson et al., 2004]
H. Eriksson, M. Penker, B. Lyons, D. Fado, *UML 2 Toolkit*, Wiley, 2004.

[OMG, 2020]
Unified Modeling Language (Specification), OMG, 2020.