

Especificação, Modelação e Projecto de Sistemas Embutidos

- **Especificação: conceitos introdutórios**
- **Modelos de computação**

Paulo Pedreiras
pbrp@ua.pt



Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

Especificação de ES: requisitos das técnicas de especificação

- Começando pelo princípio ...
 - Definição de “Especificação” (*)
 - Acção de **especificar**;
 - Descrição especificada;
 - Discriminação.
 - Definição de “especificar” (*)
 - especializar;
 - indicar a espécie de;
 - esmiuçar;
 - dizer separadamente;
 - **exprimir de forma precisa**;
 - individualizar.

(*) Dicionário de Língua Portuguesa da Priberam, 2/Set/2008

Especificação de ES: requisitos das técnicas de especificação

- Será a **linguagem natural adequada** para capturar a especificação de um sistema?
 - Considere-se o seguinte excerto do Código da Estrada da Califórnia, relativo ao direito de passagem em cruzamentos:
 - 21950. (a) The driver of a vehicle shall yield the right-of-way to a pedestrian crossing the roadway within any marked crosswalk or within any unmarked crosswalk at an intersection, except as otherwise provided in this chapter.
 - (b) The provisions of this section shall not relieve a pedestrian from the duty of **using due care** for his or her safety. No pedestrian shall **suddenly** leave a curb or other place of safety and walk or run into the path of a vehicle which is **so close** as to constitute an immediate hazard. No pedestrian shall **unnecessarily stop** or delay traffic while in a marked or unmarked crosswalk.
 - (c) The provisions of subdivision (b) shall not relieve a driver of a vehicle from the duty of exercising **due care** for the safety of any pedestrian within any marked crosswalk or within any unmarked crosswalk at an intersection.

Especificação de ES: requisitos das técnicas de especificação

- Será a **linguagem natural adequada** para capturar a especificação de um sistema?
 - Considere-se o seguinte excerto do Código da Estrada da Califórnia, relativo ao direito de passagem em

Imaginem: que tinham de **programar um veículo autónomo** por forma a garantir que o código da estrada é cumprido.

Conseguem identificar algumas **dificuldades?**

- (b) The provisions of this section shall not relieve a pedestrian from the duty of **using due care** for his or her safety. No pedestrian shall **suddenly** leave a curb or other place of safety and walk or run into the path of a vehicle which is so close as to constitute an immediate hazard. No pedestrian shall **unnecessarily stop** or delay traffic in a marked or unmarked crosswalk.
- (c) The provisions of subdivision (b) shall not relieve a driver of a vehicle from the duty of exercising **due care** for the safety of any pedestrian within any marked crosswalk or within any unmarked crosswalk at an intersection.

Especificação de ES: requisitos das técnicas de especificação

- A captura da especificação em **linguagem natural** é, em geral, **inadequada**.
- De uma forma genérica uma linguagem de especificação deve:
 - Permitir verificar se o sistema está descrito de uma forma **completa**
 - Permitir verificar se não há **contradições** na descrição do sistema
 - Deve permitir **derivar implementações** de uma forma sistemática
- Assim, a **especificação de sistemas** deve ser capturada em **linguagens formais**, processáveis por sistemas computacionais (***machine readable***)

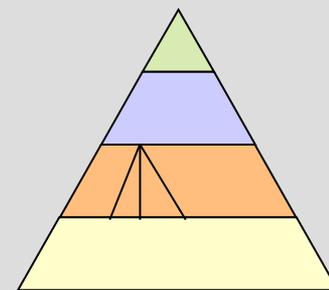
Especificação de ES: requisitos das técnicas de especificação

- As linguagens de especificação para ES devem ter a capacidade de representar os seguintes aspectos:

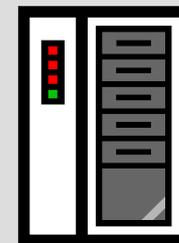
- **Hierarquia**

Os seres humanos têm **dificuldade** em **compreender** sistemas compostos por mais de ~5 objectos (e.g. estados, componentes). A esmagadora maioria dos sistemas actuais requerem maior número de objectos.

- A representação hierárquica de objectos permite resolver este dilema permitindo que em cada instante o ser humano apenas lide com um subconjunto limitado de objectos.



proc
proc
proc



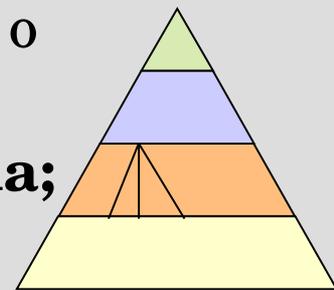
Especificação de ES: requisitos das técnicas de especificação

▪ Hierarquia (cont).

▪ Hierarquia comportamental

Contêm os objectos necessários para descrever o comportamento do sistema.

Exemplos: estados, eventos, sinais de saída;

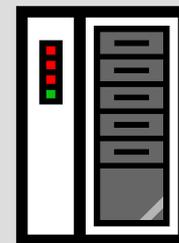
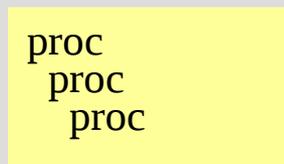


▪ Hierarquia estrutural

Descrevem como os sistemas são compostos por componentes físicos.

Exemplo:

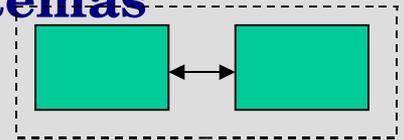
- um ES pode ser composto por CPU, memória, sensores e actuadores.
- Por sua vez os processadores são compostos por registos, multiplexers, ...



Especificação de ES: requisitos das técnicas de especificação

- **Composição de comportamentos**

Deve ser “fácil” determinar o comportamento do **sistema** a partir do comportamento dos **subsistemas**



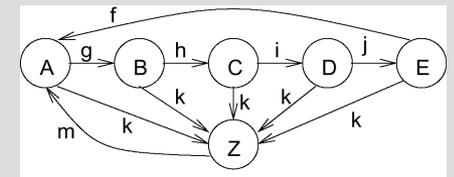
- **Comportamento temporal**

Algumas aplicações de ES possuem requisitos críticos de **pontualidade** que deve ser capturados pelas especificação



- **Comportamento orientado ao estado**
(*State-oriented behavior*)

Os automata são mecanismos adequados para modelar sistemas reactivos. Assim, os **comportamentos orientados ao estado**, disponibilizados pelos automata, devem ser **fáceis de descrever**. (Automata “clássicos” são insuficientes – comportamento temporal e hierarquia não suportados)



Especificação de ES: requisitos das técnicas de especificação

- ***Event-handling***

Devido à natureza reactiva dos ES , devem existir mecanismos apropriados para **descrever eventos** (externos e/ou internos)

- **Não devem impor obstáculos em termos da eficiência da implementação**

Muitos das aplicações de ES são muito **sensíveis a custos**; implementações eficientes permitem minimizar custos (CPU, memória, energia, ...)



- **Suporte para o projecto de sistemas críticos em termos de dependabilidade**

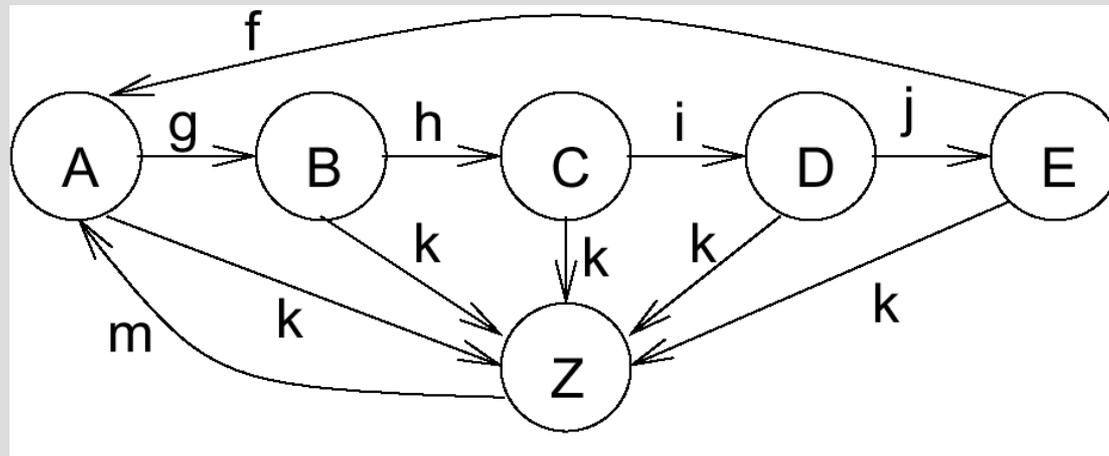
Semântica **não ambígua**, permitir/facilitar a **verificação formal**, ser capaz de descrever os **requisitos de segurança**, ...

Especificação de ES: requisitos das técnicas de especificação

- **Comportamento orientado à exceção**

Em muitos sistemas reais ocorrem exceções. Para projectar sistemas confiáveis deve ser possível tratar as exceções facilmente.

Não é desejável (ou mesmo possível, dependendo da complexidade do sistema) descrever exceções para cada estado – **descrição do sistema complexa!**



Nota: irão estudar-se técnicas em que as setas designadas por “k” serão substituídas por uma única

Especificação de ES: requisitos das técnicas de especificação

- **Concorrência**

Os sistemas reais são **distribuídos e concorrentes**.

Torna-se então necessário poder representar a concorrência de uma forma conveniente e rigorosa

- **Comunicação e Sincronização**

Os diversos **componentes** têm de **comunicar** e, frequentemente, de **partilhar** recursos (e.g. garantir exclusão mútua)



- **Presença de elementos de programação**

As linguagens de programação “**convencionais**” (e.g. “C”) têm provado ser um meio conveniente para **expressar computações** que devem ser efectuadas. Deve então ser possível **integrar** elementos da linguagem de programação (e.g. operações aritméticas, loops, chamadas a funções) na técnica de especificação usada. (e.g. diagramas de estado “classicos” não permitem)

Especificação de ES: requisitos das técnicas de especificação

- **“Executabilidade”**

A possibilidade de executar uma especificação permite verificar se esta corresponde ao “**modelo mental**”/ideia das pessoas; verificação de “plausibilidade”
(uso de linguagens de programação pode ser vantajosa neste aspecto)

- **Suporte ao projecto de sistemas de grande dimensão/complexidade**

Há uma tendência para o **aumento** da **dimensão/complexidade** dos ES. As linguagens de especificação devem contemplar esta realidade.

- **Suporte a domínios específicos**

Seria desejável que a mesma linguagem de especificação servisse todos os tipos de ES. Todavia, na prática **diferentes domínios** possuem **especificidades** que, não sendo contempladas, podem por em causa a eficiência da representação.

(e.g. domínios de aplicação *control-dominated*, *data-dominated*, centralizadas ou distribuídas podem beneficiar de particularidades da linguagem específicas para cada um deles)

Especificação de ES: requisitos das técnicas de especificação

- **Legibilidade**

Como é óbvio a especificação deve ser **legível**/entendível por **seres humanos**. Também é desejável que possam ser processáveis (legíveis) por **computadores**



- **Portabilidade e flexibilidade**

A especificação deve ser tanto quanto possível independente do hardware; deve ser **utilizável em várias plataformas**. Devem ser flexíveis por forma a que pequenas modificações no sistema requeiram apenas pequenas modificações na especificação

- **Terminação**

Deve ser claro que processos é que **terminam e quando**



- **Suporte a dispositivos de I/O não standard**

Muitos ES empregam **dispositivos de IO específicos** que é necessário descrever devidamente



Especificação de ES: requisitos das técnicas de especificação

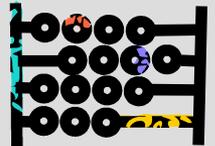
- **Capacidade para descrever propriedades não funcionais**

Os sistemas reais apresentam um grande número de **propriedades não funcionais** que devem ser definidas de formalmente:

- **Pontualidade** na execução de tarefas, **latência** na **reacção** a certos eventos, tolerância a **falhas**, EMI, consumo, peso, tamanho, tempo médio de vida, expansibilidade, ...

- **Modelo de computação adequado**

Descrição **fácil e rigorosa** do processamento que deve ser efectuado.



- **Nenhuma** linguagem de especificação **cumpre** todos estes requisitos;
 - na prática tem de se viver com **compromissos**
- A escolha da linguagem a usar depende do **domínio de aplicação**

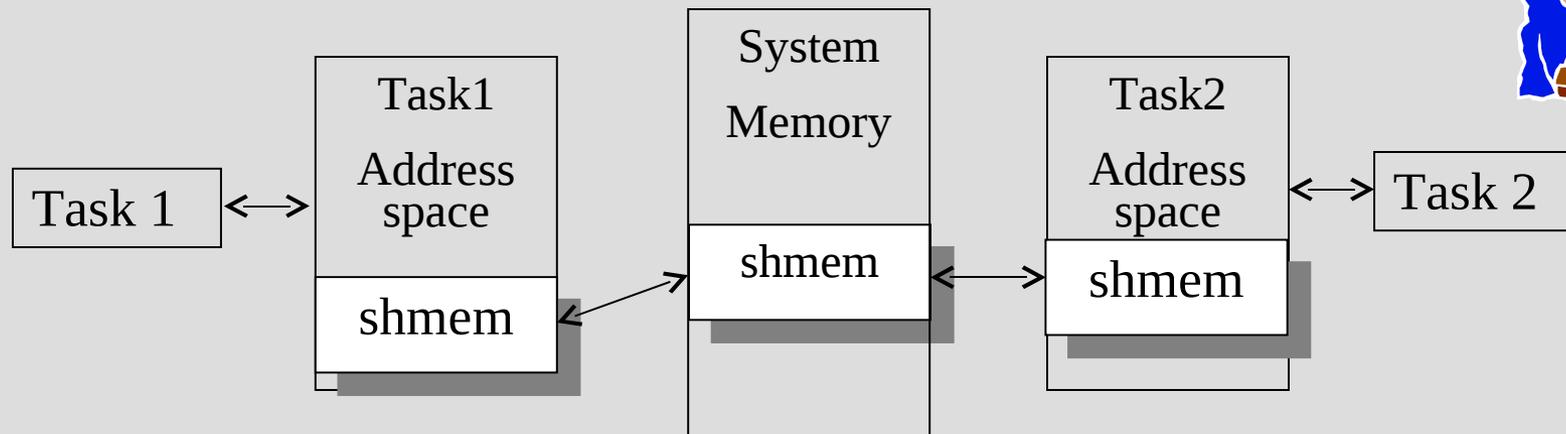
Modelos de computação

Modelos de computação: definição

- As aplicações de tecnologias de informação “convencionais” têm sido baseadas essencialmente na arquitectura de computação sequencial de Von Neumann.
 - **Este paradigma não é adequado para ES.** Por exemplo não há noção de tempo nesta arquitectura. Outros modelos de computação são necessários.
- **Um modelo de computação define:**
 - **Componentes:** e.g. processos, funções, máquinas de estados finitos, ...
 - **Protocolos de comunicação:** definem os mecanismos pelos quais os componentes podem interagir
 - **Interfaces:** definem o que os componentes conhecem uns acerca dos outros.

Modelos de computação: comunicação

- **Memória partilhada / *Shared memory***



- Variáveis **acessíveis** pelas **várias tarefas**
- Mecanismos de **implementação** possíveis:
 - **Espaço de endereçamento único** para as várias tarefas
 - E.g. threads em Linux
 - Primitivas para **criação** de zonas de memória partilhada
- Mecanismo simples, com baixo *overhead* mas **útil apenas para arquiteturas centralizadas**

Modelos de computação: comunicação

▪ Shared Memory

- Potencial para *race conditions*
 - Possibilidade de **resultados inconsistentes**
- Secções críticas = secções em que tem de se garantir exclusão mutua no acesso a um recurso (e.g. *shared mem*)



```
process a {
```

```
  P(S) //obter lock
  ..   // secção critica
  V(S) //libertar lock
  ..
}
```

```
process b {
```

```
  P(S) //obter lock
  ..   // secção critica
  V(S) //libertar lock
  ..
}
```

E.g. **exclusão mutua garantida** no acesso à *shared memory* por meio de semáforo S

Este modelo de comunicação pode ser suportado por:

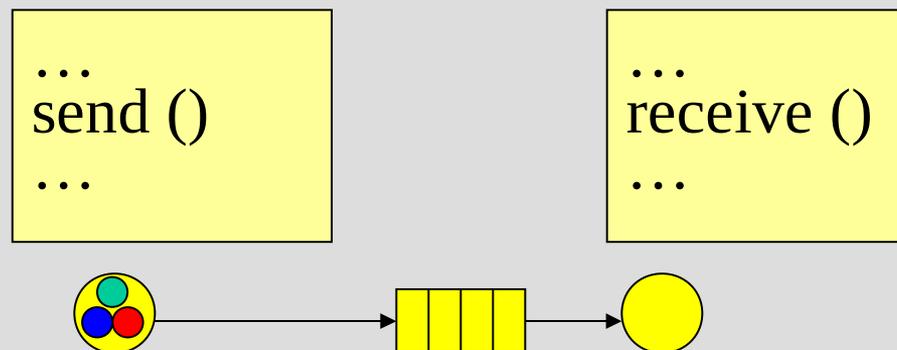
- **Exclusão mútua** para secções críticas
- Protocolos para garantia de **coerência de cache**

Modelos de computação: comunicação

- **Passagem de mensagens não bloqueante/assíncrona**

(Non-blocking / asynchronous message passing)

- Integra um **buffer** onde a tarefa **produtora deposita** as mensagens e a **tarefa consumidora as recolhe**
- O produtor **não tem** de esperar que as mensagens sejam **lidas** pela(s) tarefas consumidoras
 - Potencial problema: dimensionamento dos buffers. **Como evitar overflows?**

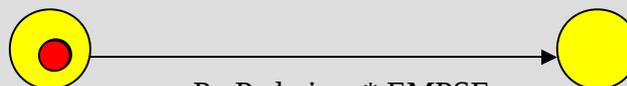
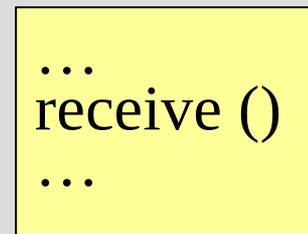
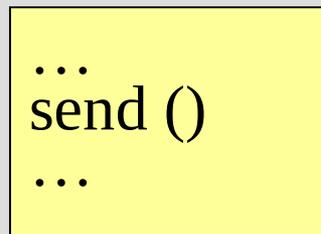


Modelos de computação: comunicação

- **Passagem de mensagens bloqueante/síncrona**

(Blocking / synchronous message passing / rendez-vous)

- O produtor **tem de aguardar** que o consumidor leia a mensagem
- Permite garantir **relações de precedência** entre tarefas
- Dimensionamento dos buffers trivial (porquê?)



Modelos de computação: comunicação

- Extended *rendez-vous*
 - Requerida **confirmação** (*acknowledge*) **explícita** do consumidor
 - Consumidor pode efectuar uma **verificação** **antes** de enviar a **confirmação** (ack)



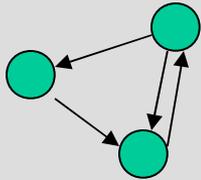
```
...  
send ()  
...
```

```
...  
receive ()  
...  
ack  
...
```

Modelos de computação: componentes

- Modelos de computação relevantes para ES

- **Máquinas de estados finitos comunicantes (CFSM)**



- Coleções de FSM com capacidade de intercomunicação. Os métodos de comunicação são diversos mas bem definidos (e.g. variáveis globais)
 - Modelo usado e.g. em linguagens de especificação como StateChart, StateFlow e SDL

- **Modelo de eventos discretos**

- Os eventos transportam um *timestamp* totalmente ordenado (i.e. que permite estabelecer univocamente a ordem temporal dos eventos) indicando o instante em que o evento aconteceu. Os simuladores de eventos discretos gerem uma fila global de eventos que são processados por ordem cronológica.
 - Depende da existência de uma ou mais filas globais e de uma noção de tempo comum;
 - Modelo usado e.g. em VHDL, Verilog, e MatLab/Simulink (MathWorks)

Modelos de computação: componentes

- Modelos de computação relevantes para ES (cont.)

- **Equações diferenciais**

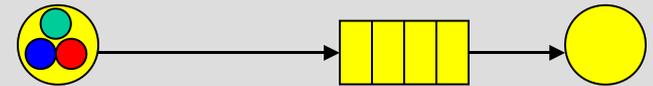
- Estas equações permitem modelar sistemas analógicos e físicos. Podem assim ser usadas para modelar ES.

$$\frac{\partial^2 x}{\partial t^2} = b$$

- **Passagem assíncrona de mensagens**

- Os processos comunicam enviando mensagens por canais que podem armazenar estas mensagens (buffered). O transmissor não necessita de aguardar que o receptor leia a mensagem
 - Potencial problema: como evitar overflows dos buffers?
 - Há diversas variantes deste esquema. E.g. *Khan process networks* e modelos *dataflow*

- **Passagem síncrona de mensagens**



- Os componentes são processos. A comunicação é atômica e instantânea (acção denominada *rendez-vous*). Quando um processo atinge um ponto de comunicação tem de aguardar que o seu parceiro também o atinja.
 - Não há risco de overflow de buffers, contrariamente ao caso anterior. A performance pode ser má (Porquê??)
 - Exemplos: CSP e ADA



Modelos de computação: componentes

- **Modelos combinados**

Muitas das linguagens usadas na prática **combinam** alguns dos **modelos** anteriormente definidos

- Exemplos:

- **SDL**

- FSM + passagem assíncrona de mensagens

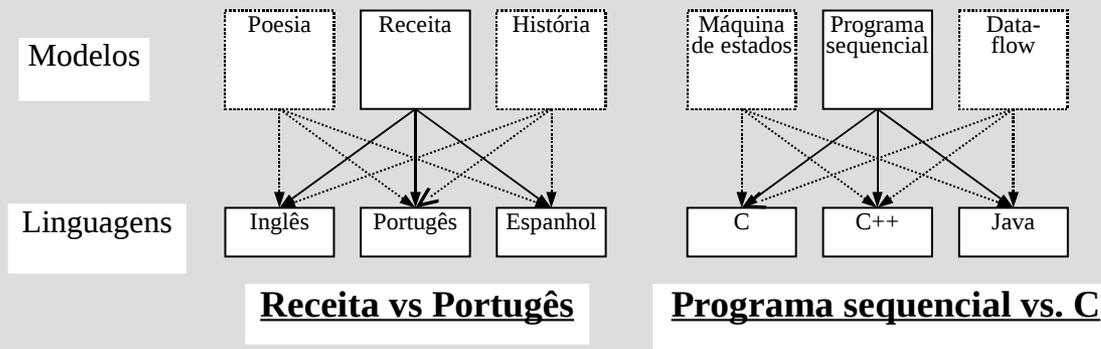
- **StateCharts**

- FSM + memória partilhada

- **CSP, ADA**

- Execução sequencial (von Neumann) + passagem síncrona de mensagens

Modelos vs. linguagens



- Os **modelos** de computação descrevem o **comportamento** do sistema
 - Noção conceptual, e.g. receita de culinária, programa sequencial
- As **linguagens capturam** os **modelos**
 - Forma concreta. E.g. Português, C
- Uma **variedade de linguagens** pode **capturar** um modelo
 - E.g., modelo de programação sequencial → C, C++, Java
- Uma **linguagem** pode capturar **vários modelos**
 - E.g., C++ → modelo de programação sequencial, modelo orientado a objectos, modelo de máquinas de estados
- Todavia certas linguagens podem ser **mais adequadas** para capturar **certos modelos** ...

Recursos adicionais

- Ptolemy (UC Berkeley)
 - Ambiente para simulação de ES com suporte a diversos modelos de computação
 - <http://ptolemy.berkeley.edu/>
- Axel Jantsch. “Modeling Embedded Systems and Soc's: Concurrency and Time in Models of Computation”, Morgan-Kaufman, 2004



Enfrentando a realidade ...

- **Nenhuma linguagem de especificação cumpre todos os requisitos**
- Necessidade de **compromissos**
- Escolher a **mais adequada** a cada domínio de aplicação



Sumário

Requisitos das linguagens de especificação

- Hierarquia
- Comportamento temporal
- Comportamento orientado ao estado
- Concorrência
- Sincronização e comunicação
- E muitos outros ...

Modelos de computação

- Definição de modelo de comunicação
 - Componentes
 - Protocolos de comunicação
 - Interfaces
- Composição de modelos
- Modelos vs linguagens