Universidade do Minho

Aula 11: Programação Orientada por Padrões

OBJECTIVOS:

- **Escrever** Programas Orientados para Padrões (OPP)

Universidade do Minho

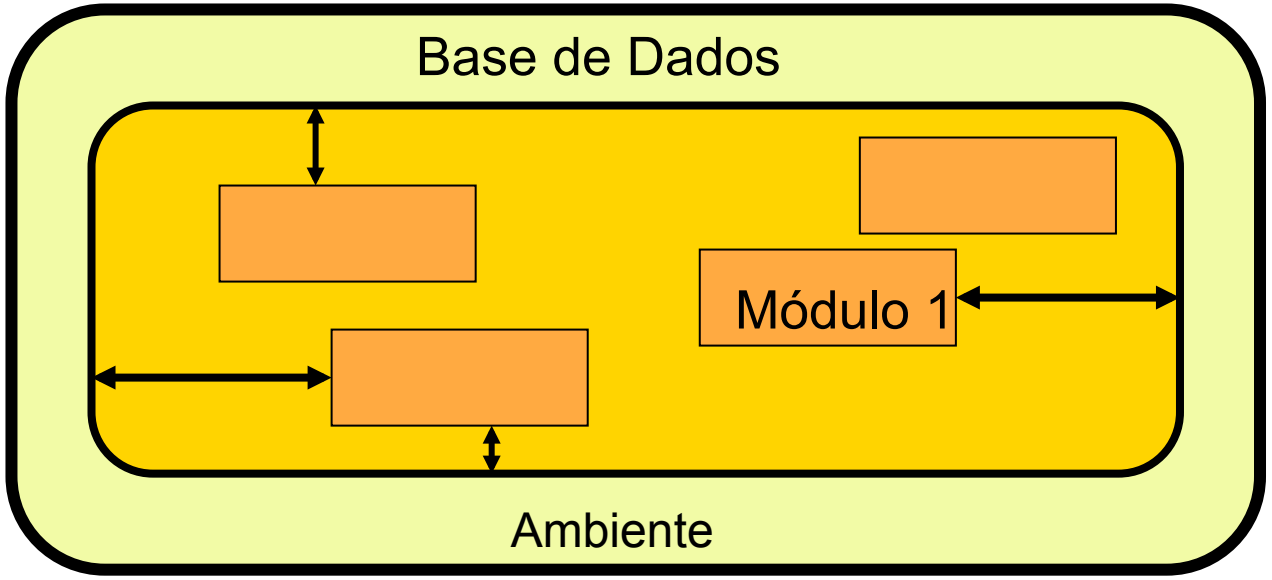
Programação Orientada por Padrões

- Trata-se de uma arquitectura para programar sistemas;
- Um **programa orientado para padrões (POPP)** é uma colecção de **módulos orientados para padrões (OPP)**;
- Cada **módulo OPP** é definido por:
  - um **padrão** de pré-condição;
  - uma **acção** a ser executada se os dados do ambiente unificam com o padrão.

Universidade do Minho

Programação Orientada por Padrões

- Na organização OPP, os módulos são despoletados pelos **padrões** que ocorrem no **ambiente do sistema** (**base de dados**).



- O elevado nível de modularidade é desejável em SBC complexas, pois é difícil prever com antecedência todas as interacções.
- Na arquitectura OPP, cada regra se-então pode ser vista como um módulo **OPP**.

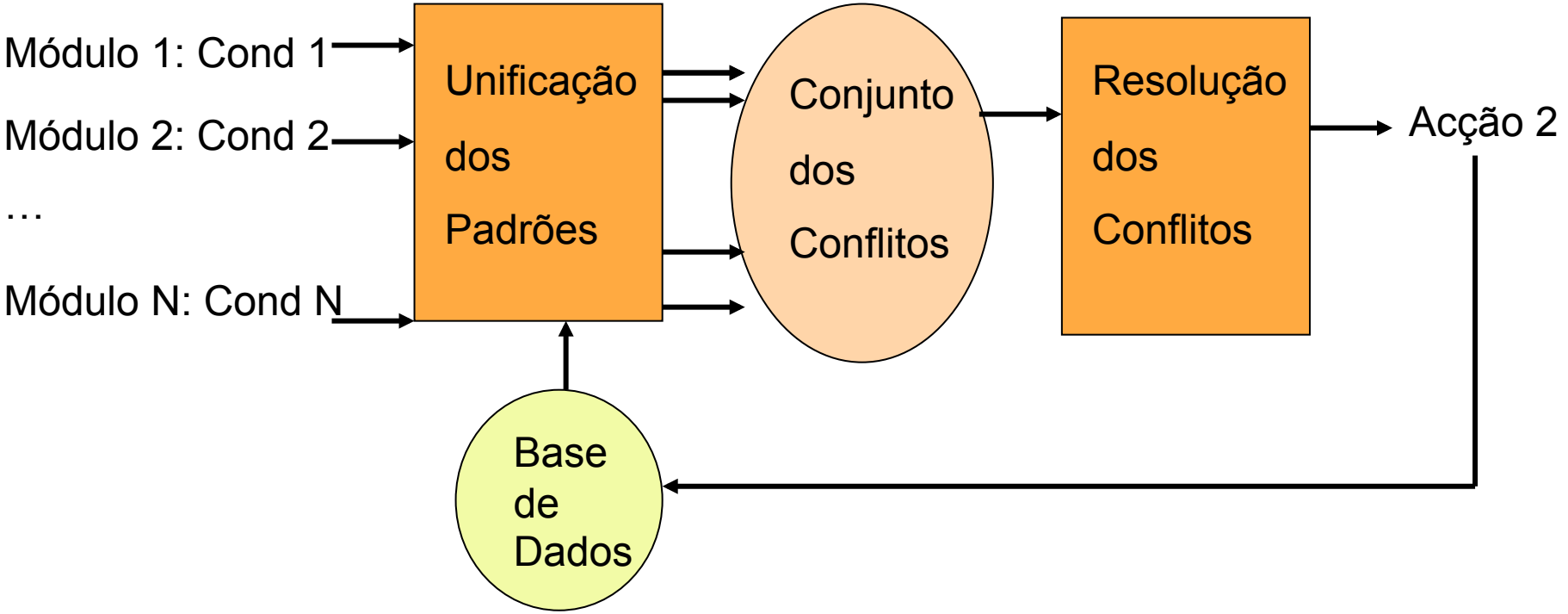
Universidade do Minho

Vantagens

- **Não existe hierarquia entre módulos** e não existe a indicação explícita de que módulo por invocar que módulo;
- Os **módulos comunicam com a Base de Dados** e não entre eles. Tal permite a **execução paralela** de vários módulos, pois o estado da BD pode satisfazer várias condições;
- Modelo natural de **computação paralela**: cada módulo pode ser implementado fisicamente num processador próprio;
- Cada módulo pode ser desenhado e implementado de um modo **autónomo**;
- O sistema pode **resistir à remoção/inserção/modificação** de alguns módulos, sem que isso lhe seja necessariamente fatal.

Ciclo de Vida dos Sistemas OPP

1) **Unificação de Padrões**: procurar na BD todas as ocorrências de padrões (pré-condições) dos módulos. Criar um conjunto de conflitos.
2) **Resolução do conflito**: seleccionar um dos módulos do conjunto de conflitos.
3) **Execução:** executar o módulo seleccionado na etapa 2.



Universidade do Minho

Exemplo: Máximo Divisor Comum

Cálcular o Máximo Divisor Comum (MDC) de dois números inteiros A e B, via o algoritmo de Euclides:

```
Enquanto A ≠ B fazer
    se A>B então A ← A-B
    senão B ← B-A
Fim Enquanto
MDC<-A
```

Módulo 1: Condição: _____

Acção: substituir X na base de dados por X-Y.

Módulo 2: Condição: _____

Acção: mostrar X como o MDC e abandonar.

Q1: Definir as condições dos módulos 1 e 2? (In-Class Teams 1 min) Nota: este programa OPP é capaz de calcular o MDC de um conjunto qualquer de números inteiros!!!

-Grupos de 3 elementos, o que fizer o aniversário mais próximo deste mês é o representante (caneta e papel).

Universidade do Minho

RC de Programas OPP: meta-interpretador

- Será adoptada a sintaxe, para especificar módulos OPP:

Condições→ Acções

- Condições correspondem a uma lista de condições:

[Condição1, Condição2, Condição3, …]

- A pré-condição é satisfeita com todas condições;
- Acções correponde a uma lista de acções:

[Acção1, Acção 2, …]

- Cada acção é também um objectivo Prolog.
- A acção '**stop**' finaliza a execução do meta-interpretador.

Universidade do Minho

MDC em Prolog:


```
% base de dados inicial
numero(5).
numero(10).
numero(15).

% módulos OPP
[ _____ ] ---> [ _____, substitui(_____,_____)].
[ _____ ] ---> [ write(X),___].
```

Q2: Preencher os _____.

(In-Class Teams 2 min)

-Grupos de 3 elementos, o que fizer o aniversário mais distante deste mês é o representante (caneta e papel).

Universidade do Minho

Sistema de Inferência em Prolog (opp.pl): meta-interpretador

```
% declaração de operadores
:-op(800,xfx,--->). %
:-op(600,fx,~). % negação

% execução do meta-interpretador via: ?- demo.
% executa os módulos OPP até encontrar stop
demo:- Condicao ---> Accao,
      testa(Condicao),
      executa(Accao).

testa([]).
testa([~Primeira|Resto]):- % negacão de condição
!,nao(Primeira),testa(Resto).
testa([Primeira|Resto]):-
!,call(Primeira),testa(Resto).
```

Universidade do Minho

Sistema de Inferência em Prolog (opp.pl): meta-interpretador

```
nao(Condicao):-call(Condicao),!,fail.
nao(_).

executa([stop]):-!. % pára se stop
executa([]):- demo. % continua com próximo OPP
executa([Primeira|Resto])
:-call(Primeira), executa(Resto).

% predicados de manipulacao da BD
substitui(A,B):- retract(A),!,asserta(B).
insere(A):- asserta(A).
retira(A):- retract(A).
```

Universidade do Minho

Exemplo do MDC em Prolog:

```
:-[opp], dynamic(numero/1).

numero(45).
numero(63).

[numero(X),numero(Y),X>Y]--->[D is X-Y,substitui(numero(X),numero(D))].
[numero(X)]--->[write(X),stop].
```

Universidade do Minho

Limitações do Sistema de Inferência:

- A **resolução de conflitos é reduzida** e fixada **à ordem prédefinida** dos módulos OPP. Melhoria: incluir um módulo de resolução de conflitos.
- Quando a Base de Dados tiver uma **elevada dimensão** e o **número de módulos OPP for grande**, então a **unificação é ineficiente**. Melhoria: alterar a organização da indexação de informação.
- O **meta-interpretador não possibilita o retrocesso** (backtracking) dada a forma de manipulação de dados. Não se podem explorar caminhos alternativos. Melhoria: implementação da base de dados sem o assert e retract, onde o estado da BD é um termo (argumento) extra passado ao demo.

Universidade do Minho

Outro Exemplo de Padrões:

- **Regra:** Se X gosta de Y e Y gosta de X então X e Y são namorados.
- **Dados:** O Rui gosta da Rita e da Ana; a Ana e a Rita gostam do rui;
- **Objectivo:** Mostrar todos pares de namorados (só é admitida monogamia!)

```
:-[opp],dynamic(gosta/2).
gosta(rui,rita).
gosta(rui,ana).
gosta(ana,rui).
gosta(rita,rui).
```

Ambiente

```
[gosta(X,Y),gosta(Y,X),X\==Y]--->[ findall(gosta(X,Z),gosta(X,Z),LX),
                                   findall(gosta(Y,W),gosta(Y,W),LY),
                                   remover(LX),remover(LY),
                                   write(namorados(X,Y)),nl].

[]--->[stop].
```

Padrões

```
% --- Predicado auxiliar ---
remover([]).
remover([X|R]):-retract(X),remover(R).
```

- **Mais exemplos de exercícios resolvidos na sebenta a disponibilizar na página da disciplina…**