Introdução à Programação Orientada pelos Aspectos

Manuel Menezes de Sequeira
DCTI
ISCTE

Desenvolvimento

- Realizado por humanos, em equipa:
  - Somos limitados
  - Não conseguimos lidar com demasiada informação
  - Temos dificuldades de comunicação
  - Erramos…
- Necessárias formas de desenvolvimento adaptadas aos humanos:
  - *Abstracção*: lidar apenas com o que é essencial em cada passo
  - *Encapsulamento*: ocultar o que se sabe ser irrelevante para a utilização de uma ferramenta (modelo de caixa preta, favorece abstracção)
  - *Decomposição*: divisão do sistema em componentes/unidades/módulos

Decomposições

- Procedimental
  - Sistema decomposto em rotinas (funções e procedimentos)
  - Abordagem descendente ou ascendente
- Centrada nos dados
- Orientada para/por Objectos
  - Centrada nos objectos, suas responsabilidades, colaborações e comportamento
  - Baseada na classificação: construção de hierarquias de classes de que objectos são instâncias

Preocupações ou *assuntos* (concerns)

- Requisitos funcionais
- Requisitos não-funcionais
- Preocupações de implementação:
  - Protecção contra erros
  - Auxílio à depuração
  - Imposição de políticas de implementação
  - Etc.

Resolvidas através políticas de desenvolvimento

A tirania da *decomposição principal*

- Decomposição em objectos permite boa modularização de algumas preocupações
- Mas não de outras…
- Exemplo:
  - É necessário garantir que nenhum objecto é criado senão a partir da respectiva fábrica abstracta
  - É necessário fazer um traçado da execução do programa

O problema

- Preocupações transversais, i.e., que *seccionam* a decomposição principal, levam a:
  - *Emaranhamento* do código: várias preocupações no mesmo módulo
  - *Espalhamento* do código: a mesma preocupação em vários módulos
  - Necessidade de disciplina do programador: meio caminho andado para o desastre
  - Código difícil de manter, actualizar, depurar, etc.
- Exemplo: Instruções de tracejamento em todos os métodos de todas as classes…

Análise sintáctica de XML no org.apache.tomcat

"Tomcat is a servlet container with a JSP environment. A servlet container is a runtime shell that manages and invokes servlets on behalf of users."

© Amiram Yehudai. School of Computer Science. Tel-Aviv University.

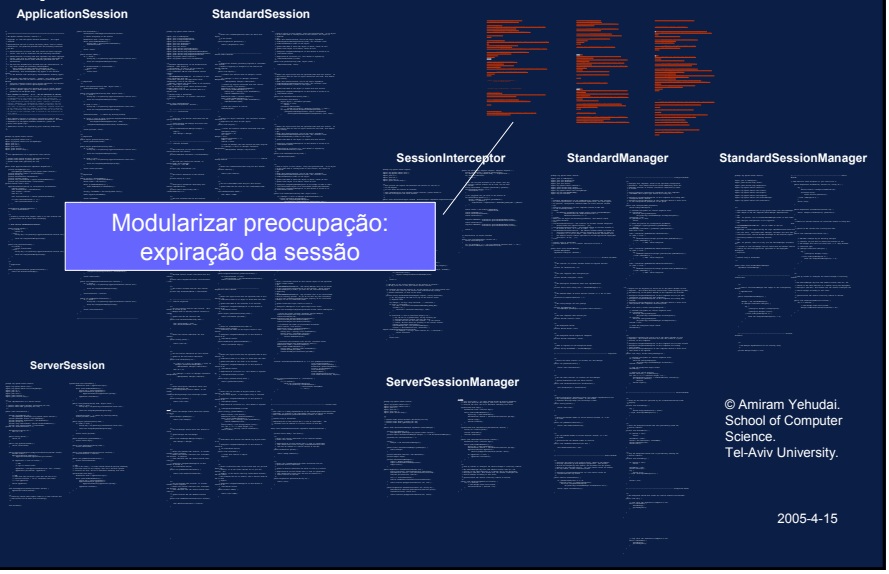
Emparelhamento de URL no org.apache.tomcat

© Amiram Yehudai. School of Computer Science. Tel-Aviv University.

Logging no org.apache.tomcat

© Amiram Yehudai. School of Computer Science. Tel-Aviv University.

O que se pretende

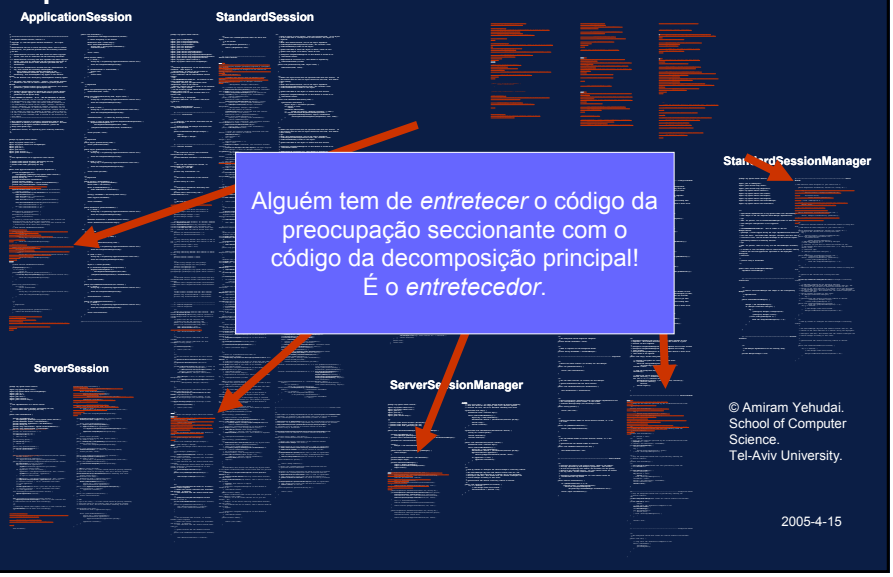


© Amiram Yehudai.
School of Computer Science.
Tel-Aviv University.

Aspecto

- Encapsulamento de uma preocupação que *secciona* a decomposição principal
- Não elimina emaranhamento/espalhamento, mas pode contribuir para o reduzir substancialmente
- *Secção*: Conjunto de pontos na decomposição principal na qual actua um aspecto (ou melhor, onde é executado um seu *conselho*)

Mas…



Desenvolvimento Orientado para Aspectos (DOA)

O DOA pode ser entendido como o desejo de fazer afirmações quantificadas *acerca do comportamento de programas e de as fazer acerca de programas sem qualquer referência explícita à possibilidade de comportamento adicional.*

Robert E. Filman e Daniel P. Friedman

Permite quantificação: *quantificabilidade*

- Utilização clássica:
  - "Qualquer que seja…"
  - "Existe um…"

$$\exists o \in \text{oradores}, \text{éChataÀBrava}(\text{apresentação}(o))$$

- Em POA/DOA:
  - "Antes de todas as invocações de operações tais que…"
  - Especificação de secções por *compreensão*
- Não é de utilização obrigatória:
  - Possível especificar *secções* por *extensão*

Permite alheamento: *alheabilidade*

- Programador da decomposição principal não precisa saber que código é *entretecido* no seu código
- Mas:
  - Em Java/C#, pode-se anotar código
  - Anotações explicitam semântica
  - Programador pode colocar anotações sabendo de possíveis aconselhamentos nelas baseados

Linguagens

- HyperJ
- LogicAJ
- AspectC++
- AspectJ
- Etc.

AspectJ

- Extensão ao Java orientada para aspectos
- Simples
- Fácil de adoptar
- Passível de adopção incremental
- Complementa orientação para objectos, que continua a ser a decomposição principal
- Fácil de fazer proselitismo

OláMundo.aj

[Projecto "Olá mundo"]

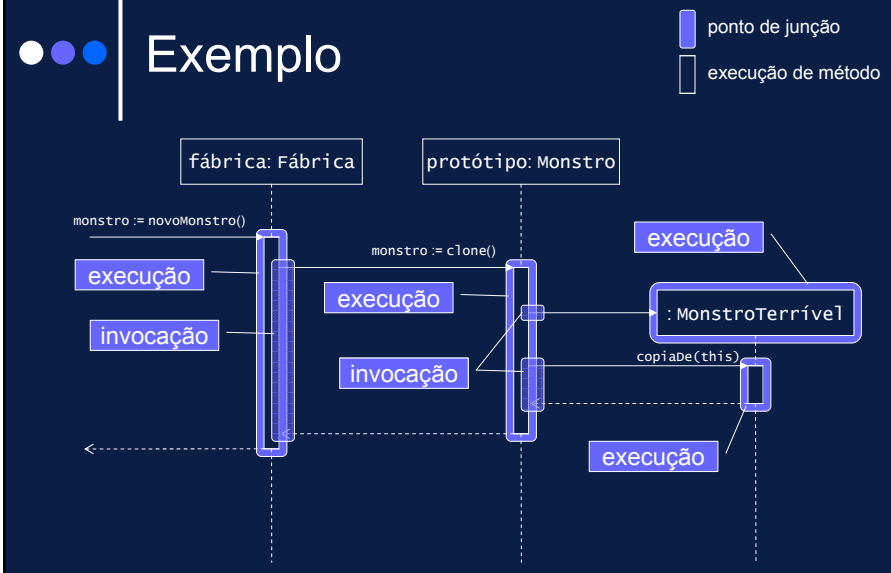
Entidades importantes

- Pontos de junção *(join points)*
  - Locais de inserção de conselhos (e não só)
- Secções (*pointcuts*)
  - Conjuntos de pontos de junção especificados através de quantificação
- Conselhos
  - Pedaços de código Java inseridos nos pontos de junção de uma dada secção
- Aspectos
  - Novas unidades de modularização em POA, definem/declaram pontos de junção, secções e conselhos

Pontos de junção

- Pontos identificáveis na execução de um programa:
  - Execuções de métodos ou construtores
  - Execução de conselhos (aspectos aconselham conselhos…)
  - Invocação de métodos ou construtores
  - Acesso de leitura de, ou escrita em, um atributo

Exemplo



Tipos de pontos de junção

Varia o contexto

- Especificados por *designadores*:
  - Invocações de métodos/construtores
  - Execuções de métodos/construtores
  - Acesso a atributos
  - Alteração de atributos
  - Inicializações
  - Etc.

Secções

- Quantificação identificando conjunto relevante de pontos de junção
- Exemplos:

```
call(Monstro+.new(..))

pointcut naFábrica() : within(Fábrica+);

call(Monstro+.new(..)) && !naFábrica()

!cflow(within(VerificaçãoDeInvariante))
```


Conselhos

- Código a executar nos pontos de junção de uma dada secção
- Exemplo:

```
after() : execution(* Fábrica+.*(..)) {
    System.out.println("Estou em " +
                       thisJoinPoint);
}
```

Reflexão em AspectJ

Tipos de conselhos

- `after`
- `after returning`
- `after throwing`
- `before`
- `around`

Aspectos

- Unidade de modularização que implementa preocupações seccionantes
- Exemplos:

  [Projecto "Fábricas abstractas"]
  [Projecto "Políticas"]

Recentemente

- AspectJ suporta definição de secções com base em anotações Java 5
- Exemplo:
  - Verificação do bom comportamento de inspectores
  - Anotação de atributos como:
    - Não pertencendo ao estado do objecto
    - Sendo parte do objecto
    - Sendo uma mera referência
  - Permite verificação de erros do programador
  - Torna linguagem mais expressiva

A evolução do AspectJ

- Como o C++:
  - Não é a primeira plataforma de POA, mas é a mais conhecida
  - Tradutores evoluem para compiladores
  - Expansão do paradigma e da linguagem
  - Manutenção de compatibilidade com linguagem base
  - Absorção do paradigma
  - Novos paradigmas
  - Morte da linguagem

Dicionário

Alheamento: *Obliviousness*
Aspecto: *Aspect*
Conselho: *Advice*
Emaranhamento: *Tangling*
Entretecedor: *Weaver*
Entretecimento: *Weaving*
Espalhamento: *Scattering*
Ponto de junção: *Join point*
Preocupação/assunto: *Concern*
Secção: *Pointcut*
Seccionante/Transversal: *Crosscutting*
Seccionar: *Crosscut*
Conselho: *Advice*

Fontes de informação

- Desenvolvimento Orientado para/por Objectos (inclui POA): http://www.aosd.net/
- AspectJ: http://www.eclipse.org/aspectj/

- Bibliografia:
  - Ramniva Laddad, *AspectJ in Action*, Manning, 2003.
  - Robert E. Filman, *et al.*, *Aspect-Oriented Software Development*, Addison Wesley, 2005.
  - Joseph D. Gradecki e Nicholas Lesiecki, *Mastering AspectJ*, Wiley Publishing, 2003.
  - Ivan Kiselev, *Aspect-Oriented Programming with AspectJ*, Sams Publishing, 2002.