DO MORE

www.rise.com.br

Variabilidade em Software

Alcemir Santos

Da individualidade à padronização

MOTIVAÇÃO

Sobre a padronização em software

- Antes do advento da produção em massa na industrialização, a produção era essencialmente artesanal

- A **produção em massa** melhorou a produtividade, quando comparada com a produção artesanal
  - A padronização dos produtos reduziu os custos de produção e melhorou a qualidade dos produtos e processos



"Quaisquer cliente pode ter um carro pintado com qualquer cor, contanto que que ela seja preto."

Ford and Crowther (1922, p. 72)

Sobre a padronização em software

- No entanto, perdeu-se a personalização dos produtos
- Somente com o advento das **linhas de produção** foi possível reaver tal personalização
- A ideia central é **compartilhar** entre os produtos tantas partes quantas forem possível

Sobre a padronização em software

- O desenvolvimento de software tem uma história parecida com a dos bens físicos
  - Os produtos de software eram artesanais para um hardware específico
  - Eram vendidos e empacotados juntamente com o hardware e em pequena quantidade

De como a variabilidade pode ser introduzida

VARIABILIDADE

Diferentes abordagens

- O que é variabilidade?
  - É a habilidade de derivar diferentes produtos a partir de um conjunto de artefatos compartilhados

- Como isso é feito?
  - Abordagens baseadas em composição
    - Implementam as funcionalidaes em forma de unidades independentes. Idealmente uma unidade por funcionalidade. Compostas quando da derivação do produto.

  - Abordagens baseadas em anotação
    - Anotam o código comum, de forma que o código que pertence a uma dada funcionalidade pode ser distinguido e separado quando da derivação do produto.

Exemplos

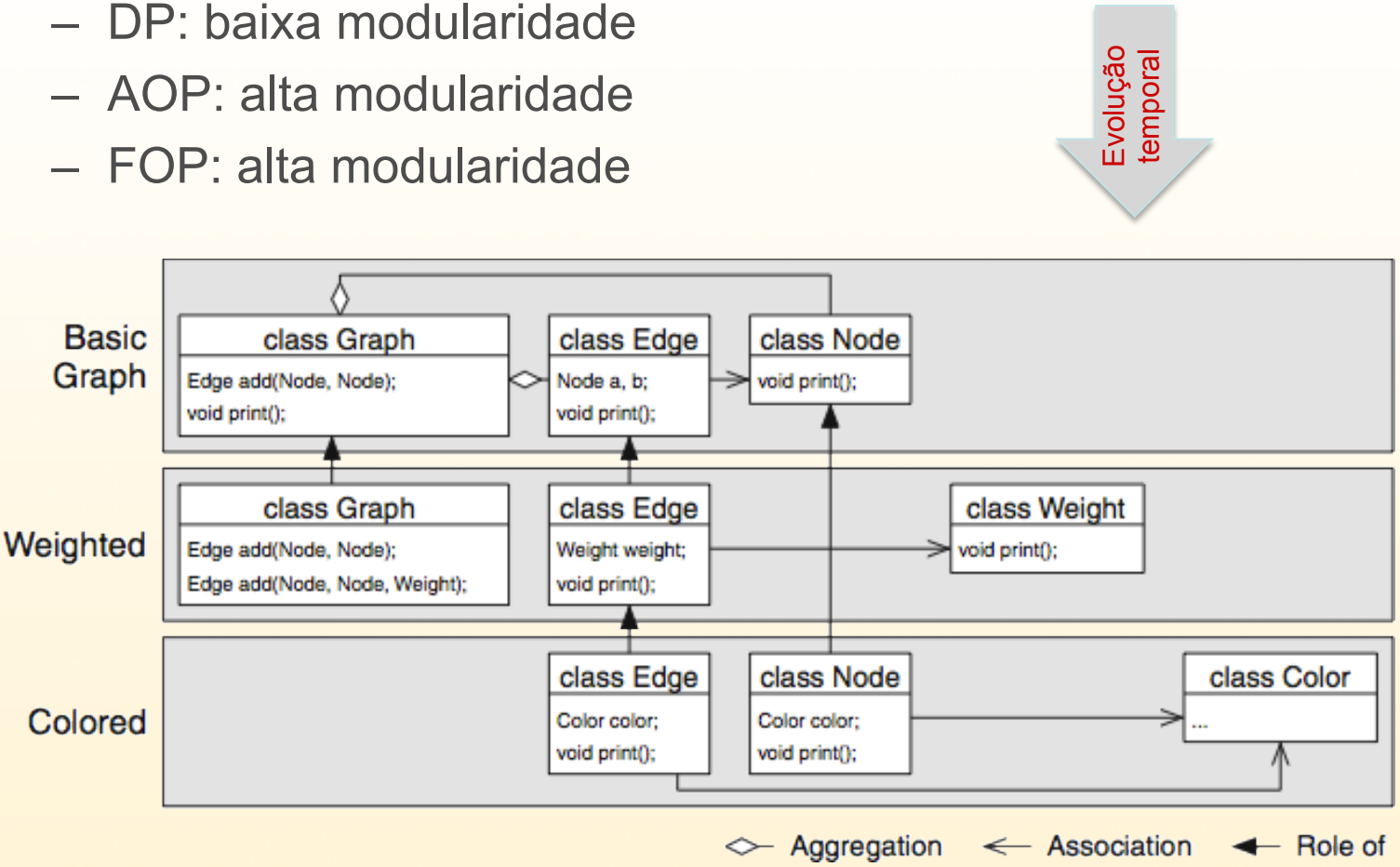
- Abordagens baseadas em composição
  - Padrões de projeto (DP)
  - Programação orientada à funcionalidade (FOP)
  - Programação orientada à aspectos (AOP)
- Abordagens baseadas em anotação
  - Parâmetros
  - Compilação condicional (`#ifdef`'s)
  - Separação virtual de interesses

De como o código é organizado

MODULARIDADE

Abordagens baseadas em composição

- O código tende a ser organizado de forma modular
  - DP: baixa modularidade
  - AOP: alta modularidade
  - FOP: alta modularidade



Abordagens baseadas em anotação

- O código tende a ser organizado de forma espalhada
  - Parâmetros
  - Compilação condicional (`#ifdef`'s)
  - Separação virtual de interesses

Evolução temporal

```
49 class Edge {
50   Node a, b;
51   Color color = new Color();
52   Weight weight;
53   Edge(Node _a, Node _b) {a=_a; b=_b;}
54   void print() {
55     if (Conf.COLORED)
56       Color.setDisplayColor(color);
57     System.out.print(" (");
58     a.print();
59     System.out.print(" , ");
60     b.print();
61     System.out.print(") ");
62     if (Conf.WEIGHTED) weight.print();
63   }
64 }
```

Parameters

```
1 #ifdef FEAT_BIGINT
2   #define SIZE 64
3 #else
4   #define SIZE 32
5 #endif
6
7 ... allocate(SIZE) ...;
```

```
1 #ifdef FEAT_SELINUX
2   #define FEAT_LINUX 1
3   #undef  FEAT_WINDOWS
4 #endif
5
6 #ifdef FEAT_WINDOWS
7 ...
```

Compilação Condicional

Edge.java

```
public class Edge {
    Node a, b;
    Weight w = new Weight(0);
    Color color = new Color(4711);

    Edge(Node _a, Node _b) {
        a = _a;
        b = _b;
    }

    void setWeight(Weight _w) {
        w = _w;
    }

    void print() {
        Color.setDisplayColor(color);
        System.out.print("edge (");
        a.print();
        System.out.print(", ");
        b.print();
        System.out.print(") ");
        System.out.print("[");
        w.print();
        System.out.print("] ");
    }
```

Separação virtual

Time for Discussion

Sobre o que vocês devem fazer

TAREFAS

Characterization Form

- Quem ainda não respondeu, responder agora.

http://goo.gl/hyzusj

Grupos

Grupo A

- Round 1
  - Sistema 1
  - Abordagem 1
- Round 2
  - Sistema 2
  - Abordagem 2

- Arquivo
  - http://goo.gl/hUYjrQ

Grupo B

- Round 1
  - Sistema 1
  - Abordagem 2
- Round 2
  - Sistema 2
  - Abordagem 1

- Arquivo
  - http://goo.gl/bcBVNr

JRE Portável: http://goo.gl/SscEpr

Tarefas

Round 1

- Tarefa única
  - Localizar código que implementa uma dada funcionalidade
- 11 funcionalidades diferentes
- 30 minutos

Round 2

- Tarefa única
  - Localizar código que implementa uma dada funcionalidade
- 10 funcionalidades diferentes
- 30 minutos

Observações

- Cuidado com o tempo utilizado em cada tarefa: são 21

- Código espalhado
  - É possível encontrar pastas vazias. Neste caso, o código encntra-se espalhado.
  - É possível utilizar busca

- Subjectcode
  - ID do seu endereço de e-mail

- Ao terminar…
  - Renomear a pasta para “Group X – Subjectcode”
  - Compactar e enviar para **alcemir.santos@gmail.com**
    - **Assunto da mensagem: Experimento ES**

Feedback Form

- Responder após, finalizar as tarefas do Round 2.

http://goo.gl/kjCVly