Introdução aos Padrões de Software

Profa. Rosana T. Vaccare Braga

Reuso de Software

- replicação de qualquer tipo de conhecimento sobre um sistema em outros sistemas similares, com o objetivo de reduzir o esforço de desenvolvimento e a manutenção nesses novos sistemas *(Biggerstaff e Perlis, 1989)*.

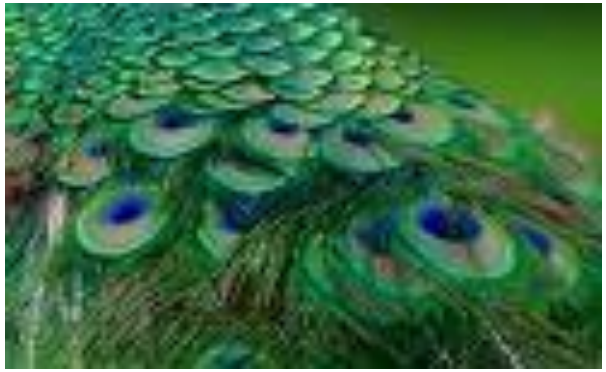
Introdução

- **Reuso de Software:**
  - **Anos 70: módulos e sub-rotinas**
  - **Anos 80: classes e geradores de aplicação**
  - **Anos 90: análise de domínio, componentes, padrões e frameworks;**

Benefícios da reutilização

- Melhores índices de produtividade
- Produtos de melhor qualidade, mais confiáveis, consistentes e padronizados
- Redução dos custos e tempo envolvidos no desenvolvimento de software
- Maior flexibilidade na estrutura do software produzido, facilitando sua manutenção e evolução

Introdução aos Padrões



Introdução aos Padrões

Introdução aos Padrões

- **Origem dos padrões:**
  - **Christopher Alexander (1977, 1979)**
    - **"Cada padrão descreve um problema que ocorre repetidas vezes em nosso ambiente, e então descreve o núcleo da solução para esse problema, de forma que você possa utilizar essa solução milhões de vezes sem usá-la do mesmo modo duas vezes"**
    - **Proposta de padrões extraídos a partir de estruturas de edifícios e cidades de diversas culturas, com o intuito de ajudar as pessoas a construir suas comunidades com a melhor qualidade de vida possível**

Introdução aos Padrões

- **Origem dos Padroes em software:**
  - **Beck e Cunningham: 1987**
    - **pequena linguagem de padrões para guiar programadores inexperientes em Smalltalk**
  - **Peter Coad: 1992**
    - **padrões de análise descobertos na modelagem de sistemas de informação**
  - **James Coplien – 1992**
    - **catálogo de vários estilos ("idioms"), com o intuito de padronizar a escrita de código em C+**
  - **Gamma et al – 1995**
    - **padrões de projeto derivados a partir de experiência prática com desenvolvimento de software orientado a objetos**

Padrões de Software

- **Por que Padrões?**
  - **Desenvolvedores acumulam soluções para os problemas que resolvem com freqüência**
  - **Essas soluções são difíceis de serem elaboradas e podem aumentar a produtividade, qualidade e uniformidade do software**
  - **Como documentar essas soluções de forma que outros desenvolvedores, menos experientes, possam utilizá-las?**

Padrões de Software

- **Padrões de Software:**
  - –Descrevem soluções para problemas que ocorrem com freqüência no desenvolvimento de software *(Gamma 95)*
- **Um bom padrão:**
  - –resolve um problema
  - –é um conceito aprovado (não apenas teoria)
  - –a solução não é óbvia
  - –descreve um relacionamento (estruturas e mecanismos)
  - –tem um componente humano significativo

Padrões de Software

- **Vantagens de Padrões?**
  - **Aumento de produtividade**
  - **Uniformidade na estrutura do software**
  - **Aplicação imediata por outros desenvolvedores**
  - **Redução da complexidade: blocos construtivos**

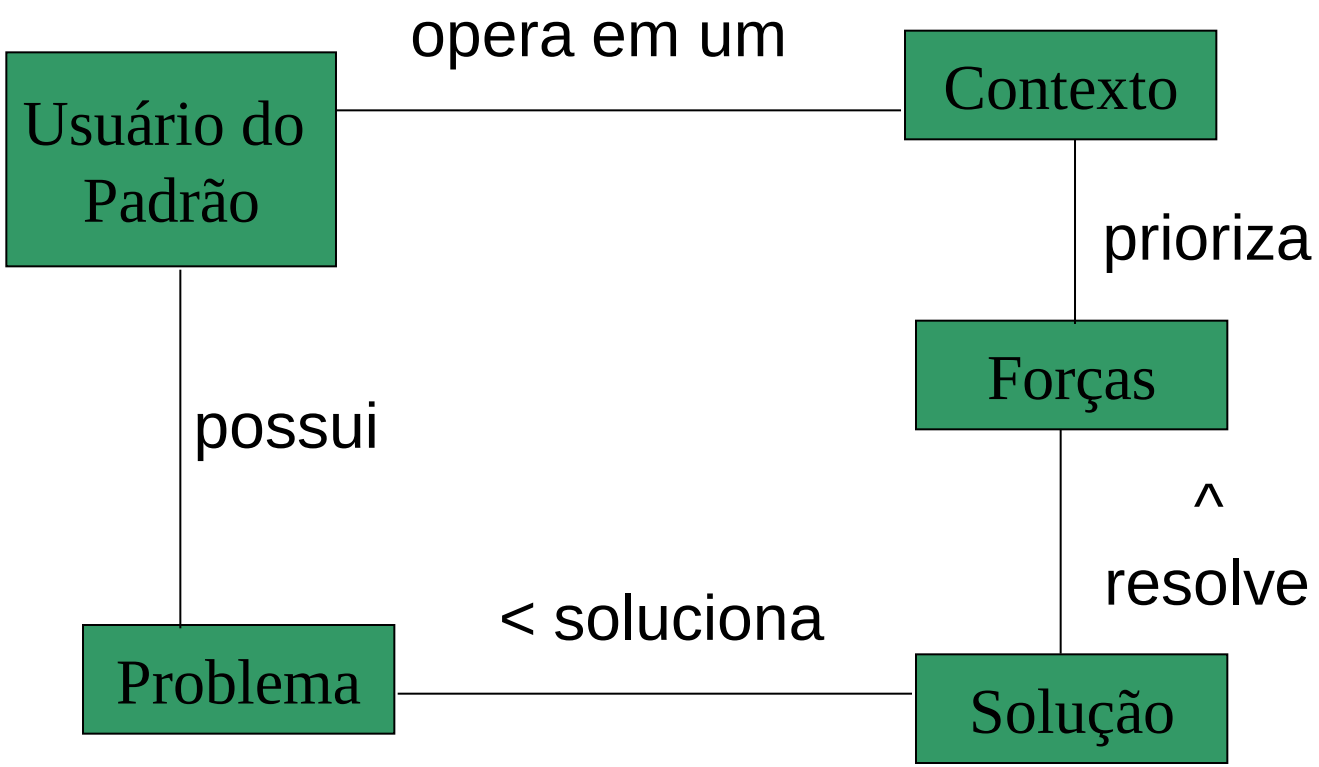
Elementos de um Padrão de Software

- Um padrão tem elementos essenciais e outros opcionais
- Os seguintes elementos são essenciais em um padrão:
  - Nome
  - Problema
  - Solução
  - Conseqüências

Elementos Essenciais de um padrão

- **Nome**: descreve o padrão de maneira muito sucinta (uma ou duas palavras), servindo como referência para comunicação entre os desenvolvedores;
- **Problema**: descreve o problema e o contexto em que ele ocorre. Pode conter um conjunto de condições que devem ser satisfeitas para a aplicação do padrão;
- **Solução**: descreve os elementos necessários para a solução do problema, seus relacionamentos, responsabilidades e colaborações. Essa solução não é restrita a um caso em particular, e sim genérica o suficiente para ser aplicada a diferentes situações;
- **Conseqüências**: são os resultados, positivos e negativos, de se utilizar o padrão. As conseqüências permitem analisar os custos e benefícios da aplicação de um determinado padrão.

Relacionamento entre Elementos do Padrão (Meszaros & Doble)



Elementos Opcionais em um padrão

- **Intenção**: resume o que o padrão faz, bem sucintamente, para ajudar na escolha entre um padrão ou outro;
- **Contexto**: descreve o ambiente em que o problema ocorre e no qual o padrão pode ser aplicável.
  - Trata-se de uma alternativa em que o elemento "problema" da estrutura básica é desmembrado, originando esta seção;
- **Motivação**: mostra um cenário, por meio de um exemplo prático, em que há um problema de projeto e como estruturas de classes e objetos no padrão resolvem o problema;

Elementos Opcionais em um padrão

- **Forças**: apresentam restrições ou pontos positivos que devem ser levados em consideração ao tentar solucionar o problema.
  - Por um lado há forças mais técnicas, como por exemplo, questões de desempenho ou memória, e por outro há forças mais humanas, como legibilidade e compreensibilidade.
- **Participantes**: descreve em detalhes cada um dos elementos que compõem a solução;
- **Colaborações ou Dinâmica**: descreve como os participantes colaboram entre si para cuidarem de suas responsabilidades;

Elementos Opcionais em um padrão

- **Figura (ou Sketch): descreve o funcionamento do padrão por meio de desenhos, modelos ou diagramas.**
  - **Pode ser um elemento à parte ou ser embutido na Solução;**
- **Aliases ou Também Conhecido Por: cita outros nomes usados para o padrão, se existirem.**
  - **Muitas vezes esses outros nomes são variantes ou foram escritos por outros autores com outro enfoque;**

Elementos Opcionais em um padrão

- **Contexto Resultante**: descreve o estado do sistema após a aplicação do padrão, mostrando:
  - quais forças foram resolvidas
  - que novos problemas podem surgir decorrentes de sua aplicação e
  - que padrões podem ou devem ser aplicados depois dele *[Coplien, 1996]*.
  - pode também ser embutido no elemento "Conseqüências";

Elementos Opcionais em um padrão

- **Padrões Relacionados ou Veja Também**: mostra o relacionamento do padrão com outros padrões que se referem ao mesmo problema, podendo citar outros padrões usados em conjunto, outras soluções para o mesmo problema ou variações do padrão;
- **Raciocínio** (ou Rationale): descreve porque a solução é a mais apropriada para o problema dentro do contexto existente e detalha como e porquê a solução funciona.
  - Alternativamente, pode contar a história que deu origem ao padrão, funcionando como uma fonte de aprendizagem;

Elementos Opcionais em um padrão

- **Implementação**: para padrões em níveis de abstração mais baixos, como os padrões de projeto, este elemento ajuda a descrever técnicas, dicas ou questões específicas de linguagem necessárias para implementar o padrão;
- **Exemplos**: apresenta exemplos de como aplicar o padrão, auxiliando o entendimento da solução quando ela é muito abstrata;
- **Usos Conhecidos**: mostra exemplos da aplicação do padrão em sistemas existentes.
  - Recomenda-se incluir pelo menos três usos conhecidos (regra de 3).

Formato de um padrão

- **Ao escrever um padrão, pode-se combinar os elementos essenciais e os elementos opcionais.**
- **A princípio, não deve se prender a nenhum formato de escrita e não há obrigatoriedade de uso de todos os elementos em todos os padrões.**
- **Na medida em que o padrão vai sendo reusado, pode-se mudar seu formato para facilitar seu entendimento e reuso**
- **Existem algumas sugestões de gabaritos já testados e aprovados : *Gamma et al. (1995), Coplien (1996) e Buschmann et al. (1996b)***

Formato de um padrão de Software - exemplos

Gamma	**Coplien**	**Buschmann**
Nome	Nome	Nome
Classificação	Contexto	Também Conhecido Como
Intenção	Problema	Exemplo
Também Conhecido Como	Forças	Contexto
Motivação	Solução	Problema
Aplicabilidade	Esquema	Solução
Estrutura	Contexto	Estrutura
Participantes	Resultante	Dinâmica
Colaborações	Raciocínio	Implementação
Implementação		Exemplo Resolvido
Código Exemplo		Variantes
Conseqüências		Usos Conhecidos
Usos Conhecidos		Conseqüências
Padrões Relacionados		Veja Também

Tipos ou Categorias de Padrões

- Padrões podem ser
  - Organizacionais
  - Arquiteturais
  - De Análise
  - De Projeto
    - Gamma et al
    - Outros : p.ex J2EE
  - De programação (estilos)
  - De usabilidade
  - Educacionais
  - Etc.

Padrões de Projeto - GoF

- **Catálogo de Padrões de Projeto [Gamma95]**
  - **Dois critérios de classificação**
    - **Propósito - reflete o que o padrão faz**
      - **De Criação: trata da criação de objetos**
      - **Estrutural: cuida da composição de classes e objetos**
      - **Comportamental: caracteriza o modo como as classes e objetos interagem e distribuem responsabilidades**
    - **Escopo**
      - **Classe: trata do relacionamento entre classes e subclasses (herança - relacionamento estático)**
      - **Objetos: lida com a manipulação de objetos (podem ser modificados em tempo de execução)**

GoF: Gang of Four – apelido dado aos quatro autores do livro

Padrões de Projeto - GoF

Escopo	Propósito: De Criação	Propósito: Estrutural	Propósito: Comportamental
Classe	Método-fábrica	Adaptador	Interpretador Método Gabarito
Objeto	Fábrica Abstrata Construtor Protótipo Objeto Unitário	Adaptador Ponte Composto Decorador Fachada Peso-pena Procurador	Cadeia de Responsabilidade Comando Iterador Mediador Memento Observador Estado Estratégia Visitador

Coletâneas de padrões

- **Padrões podem ser agrupados em diversos tipos de coletâneas para facilitar sua recuperação:**
  - **Coleções**
  - **Catálogos**
  - **Linguagens**
  - **Sistemas**

Coleções de padrões

- coletânea qualquer de padrões que não possuem nenhum vínculo entre si e, em geral, nenhuma padronização no formato de apresentação.
  - padrões apresentados em um mesmo congresso, propostos pelo mesmo autor, ou que se referirem a um mesmo domínio.
  - Pouco relacionamento semântico significativo.
- Exemplos
  - Livros como os da coleção PloPD *[Carlian

Catálogo de padrões

- coleção de padrões relacionados – talvez apenas fracamente ou informalmente
  - em geral, subdivide os padrões em um pequeno número de categorias abrangentes e pode incluir algumas referências cruzadas entre os padrões.
- Exemplo: os padrões de projeto do livro de *Gamma et al. (1995)*

Sistema de padrões

- **conjunto coeso de padrões correlacionados que trabalham juntos para apoiar a construção e evolução de arquiteturas completas** ***[Buschmann et al., 1996b]***
  - **organizados em grupos e subgrupos relacionados em múltiplos níveis de granularidade.**
- **Exemplos: Livros como o POSA** ***[Buschmann et al., 1996] e POSA 2 [Schmidt et al., 2000]***

Linguagem de padrões

- **coleção estruturada de padrões que se apóiam uns nos outros para transformar requisitos e restrições numa arquitetura *[Coplien, 1998]*.**
  - **Seus padrões constituintes cobrem todos os aspectos importantes de um dado domínio e pelo menos um padrão deve estar disponível para cada aspecto da construção e implementação de um sistema de software: não pode haver "vazios" ou "brancos".**

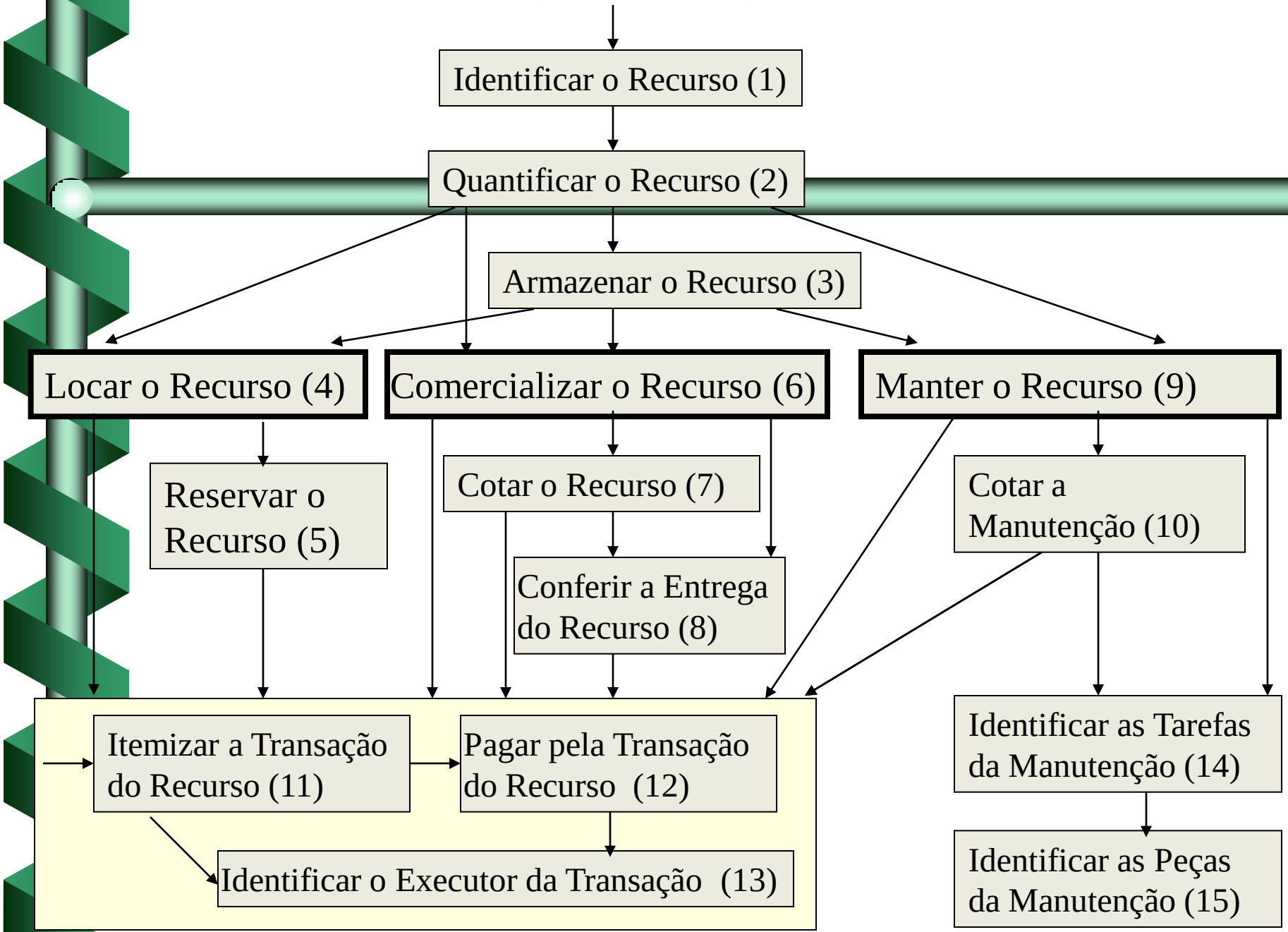
Linguagem de padrões

- **Exemplos**
  - **Linguagem de padrões para Checagem de integridade da Informação *[Cunningham et al, 1996]***
  - **Linguagem de padrões para escrita de padrões *[Meszaros e Doble, 1996]***
  - **Linguagem de Padrões para Gestão de Recursos de Negócios *[Braga et al., 1999]***

Categorização de Padrões

- **Objetivo**: facilitar a recuperação
- Exemplos:
  - Por **ciclo de vida**: Padrões de análise, de projeto, de implementação, ...
  - Por **camada de implementação**: padrões de persistência, padrões específicos de domínio, padrões de interface gráfica com o usuário, ...

Padrões de análise – GRN: Uma linguagem de Padrões para Gestão de Recursos de Negócios (Braga et al 1999)



Padrões de Persistência

Persistence Layer
CRUD
SQL Code Description
Attribute Mapping Methods
Type Conversion
Change Manager
OID Manager
Transaction Manager
Connection Manager
Table Manager

Connecting Business Objects to Relational Databases
Joseph W. Yoder, Ralph E. Johnson, Quince D. Wilson, 1998.

Tabela 5.2. Classificação dos padrões arquiteturais (Buschmann et al., 1996).

Categorias de Problemas	Categorias de Padrões		
	Padrões Arquitetônicos	**Padrões de Projeto**	**Idiomas**
Da Lama à Estrutura	Camadas Dutos e Filtros Quadro negro		
Sistemas Distribuídos	Intermediário Dutos e Filtros Microkernel		
Sistemas Interativos	Modelo-Visão-Controlador (MVC) Apresentação-Abstração-Controle		
Sistemas Adaptáveis	Microkernel Reflexão		
Decomposição Estrutural		Todo-parte	
Organização de Trabalho		Mestre-Escravo	
Controle de Acesso		Procurador	
Gerenciamento		Processador de Comandos Tratador de Visões	
Comunicação		Forwarder-Receiver Cliente-Despachante-Servidor Publicação-Assinatura	
Tratamento de Recursos			Ponteiro Contador

Tabela 5.3. Classificação dos padrões de projeto GoF (Gamma et al., 1995).

Escopo	Propósito		
	Criação	*Estrutural*	*Comportamental*
Classe	Método-Fábrica	Adaptador (classe)	Interpretador Método-gabarito
Objeto	Fábrica Abstrata Construtor Protótipo Objeto Unitário	Adaptador (objeto) Ponte Composto Decorador Fachada Peso-pena Procurador	Cadeia de Responsabilidade Comando Iterador Mediador Memória Observador Estado Estratégia Visitador

Conferências sobre Padrões - PLoP

- **Submissão de artigos: processo de "shepherding"**
- **Conferência: Sessões de "workshop do autor – *writers' workshop*", intercaladas por "jogos"**
- **Grupos com 6 a 8 autores e 6 a 8 não autores, 1 líder e 1 secretário.**

Conferências sobre Padrões - PLoP

- **Workshop:**
  - Líder pede a alguém para fazer um resumo do padrão
  - Autor tem 5 minutos para ler trechos do artigo para os demais participantes, que já leram de antemão.
  - Autor “congela”
  - 15 minutos: demais participantes levantam pontos fortes do padrão
  - 15 minutos: demais participantes apresentam sugestões para melhoria
  - Autor “descongela” e tem mais 10 minutos para perguntas e respostas.

Introdução aos Padrões

- Conferências sobre padrões
  - Primeira conferência PLoP – 1994 - EUA
  - Outras conferências:
    - EuroPLoP – Europa
    - ChiliPLoP – EUA (Arizona)
    - KoalaPLoP – Austrália
    - MensorePLoP – Japão
    - VikingPLoP - Escandinávia
  - No Brasil: SugarloafPLoP!!!

Allerton House – Monticello – Illinois - EUA

PLoP

EuroPLoP - Alemanha

2001: 1st Latin-American Conference on Pattern Languages of Programs

Rio de Janeiro

2005: 5th Latin-American Conference on Pattern Languages of Programs

Campos do Jordão – São Paulo

2007: 6th Latin-American Conference on Pattern Languages of Programs

Porto de Galinhas – PE

SugarLoafPLoP: Fortaleza – CE

2008

SugarLoafPLoP: Salvador – BA

2010

SugarLoafPLoP: Natal – RN

2012

SugarLoafPLoP: Ilhabela – SP

2014

Discussão

- **Padrões documentam soluções para problemas que ocorrem com freqüência durante o desenvolvimento de software**
  - **aumento da produtividade e qualidade no desenvolvimento.**
- **A proliferação e divulgação de padrões, bem como de ferramentas que auxiliam o desenvolvimento com padrões, fará com que em pouco tempo o uso de padrões fique inerente ao processo de desenvolvimento.**

Discussão

- **O uso de padrões têm crescido visivelmente na indústria de software.**
  - **cada vez mais empresas fornecem aos funcionários recém-contratados cursos sobre padrões de projeto, padrões J2EE, padrões de persistência, padrões arquiteturais, etc.**
  - **Frameworks de apoio ao uso de padrões também estão se popularizando. Os próprios ambientes de desenvolvimento, tais como ambientes para Java [Core J2EE, 2001; Java 2002] e ferramentas CASE, como a Rational Rose [Rose, 2005; Forbrig et al., 2001], já disponibilizam formas de reusar padrões durante o desenvolvimento.**

Discussão

- **Pesquisas sobre padrões**
  - **Apresentação de novos padrões**
  - **Uso de padrões na indústria de software, explorando principalmente comparações relativas à produtividade e qualidade decorrentes do uso de padrões**
  - **Ferramentas para facilitar a recuperação e uso de padrões. Por exemplo**
    - **Repositório de Padrões**
    - **GREN-Wizard**

Mais informações

- www.hillside.net

Referências

- Alexander, Christopher et. al. (1977) A Pattern Language. Oxford University Press.
- Alexander, Christopher (1979) The Timeless Way of Building. Oxford University Press.
- Beck, Kent; Cunningham, Ward (1987) Using Pattern Languages for Object-Oriented Programs, Technical Report nº CR-87-43,
- Coplien, J. O. (1996) Software Patterns. SIGS books and Multimedia, Junho.
- Cunningham, W. (1996) The CHECKS Pattern Language of Information Integrity, in Coplien & Schmidt (1996), p. 145-155.
- Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. (1995) Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley.
- MESZAROS, G.; DOBLE, J. A pattern language for pattern writing, cap. 29 in J. Coplien; D. Schmidt. Pattern Languages of Program Design, Reading-MA, Addison-Wesley, p. 529–574, 1998.