Padrões de Projeto

Prof. Ma. Marina Girolimetto

marina.girolimetto@uffs.edu.br

Padrões de Projeto

- Um projeto deve ser específico para o problema a resolver, mas também genérico o suficiente para atender problemas e requisitos futuros.
- Também se deseja evitar o re-projeto, ou pelo menos minimizá-lo.
- Os mais experientes projetistas de software orientado a objetos lhe dirão que **um projeto reutilizável e flexível é difícil**, se não impossível, de obter corretamente da primeira vez. Por isso, **estes quando encontram uma boa solução, a utilizam repetidamente**.

Padrões de Projeto

- Você já deve ter encontrado padrões, de classes e de comunicação entre objetos, que reaparecem frequentemente em muitos sistemas orientados a objetos.
- **Esses padrões resolvem problemas específicos de projetos** e tornam os projetos orientados a objetos mais flexíveis e, em última instância, reutilizáveis.

- **ATENÇÃO: Nenhum dos padrões de projeto descreve projetos novos ou não-testados.**

O que é um padrão de projeto?

Christopher Alexander afirma: “cada padrão descreve um problema no nosso ambiente e o cerne da sua solução, de tal forma que você possa usar essa solução mais de um milhão de vezes, sem nunca fazê-lo da mesma maneira ”.

- Em geral, um padrão tem quatro elementos essenciais:
- 1. O **nome do padrão** é uma referência que podemos usar para descrever um problema de projeto, suas soluções e consequências em uma ou duas palavras.
- 2. O **problema** descreve em que situação aplicar o padrão. Ele explica o problema e seu contexto. Pode descrever problemas de projeto específicos, descrever estruturas de classe ou objeto.

O que é um padrão de projeto?

- 3. A **solução** descreve os elementos que compõem o padrão de projeto, seus relacionamentos, suas responsabilidades e colaborações.
- 4. As **consequências** são os resultados e análises das vantagens e desvantagens da aplicação do padrão. São críticas para a avaliação de alternativas de projetos e para a compreensão dos custos e benefícios da aplicação do padrão.

Uma vez que a reutilização é frequentemente um fator no projeto orientado a objetos, **as consequências de um padrão incluem o seu impacto sobre a flexibilidade, a extensibilidade ou a portabilidade de um sistema**.

Padrões de Projeto

- **Padrões de projeto não são projetos mas sim descrições de objetos e classes que precisam ser personalizadas para resolver um problema geral de projeto num contexto particular.**

- O padrão de projeto identifica as classes e instâncias participantes, seus papéis, colaborações e a distribuição de responsabilidades.
- Cada padrão de projeto focaliza um problema ou tópico particular de projeto orientado a objetos.

Padrões de Projeto

- **Ele descreve em que situação pode ser aplicado**, se ele pode ser aplicado em função de outras restrições de projeto e as consequências, custos e benefícios de sua utilização.

- **Um padrão de projeto também fornece exemplos em código para ilustrar uma implementação.**

Descrevendo os padrões de projeto

- **Nome e classificação do padrão**: Um bom nome é vital, porque ele se tornará parte do seu vocabulário de projeto.

- **Intenção e objetivo:** É uma curta declaração que responde às seguintes questões: O que faz o padrão de projeto? Quais os seus princípios e sua intenção? Que tópico ou problema particular de projeto ele trata?

- **Motivação**: Um cenário que ilustra um problema de projeto e como as estruturas de classes e objetos no padrão solucionam o problema.

Descrevendo os padrões de projeto

- **Aplicabilidade:** Quais são as situações nas quais o padrão de projeto pode ser aplicado? Que exemplos de maus projetos ele pode tratar? Como você pode reconhecer essas situações?

- **Estrutura:** Uma representação gráfica das classes do padrão, diagramas de interação...

- **Participantes:** As classes e/ou objetos que participam do padrão de projeto e suas responsabilidades.

Descrevendo os padrões de projeto

- **Colaborações:** Como as classes participantes colaboram para executar suas responsabilidades.

- **Consequências**: Como o padrão suporta a realização de seus objetivos? Quais são os seus custos e benefícios e os resultados da sua utilização? Que aspecto da estrutura de um sistema ele permite variar independentemente?

- **Implementação**: Que armadilhas, sugestões ou técnicas você precisa conhecer quando da implementação do padrão? Existem considerações específicas de linguagem?

Descrevendo os padrões de projeto

- **Exemplo de código**: Fragmentos ou blocos de código que ilustram como você pode implementar o padrão.

- **Usos conhecidos**: Exemplos do padrão encontrados em sistemas reais.

- **Padrões relacionados**: Que padrões de projeto estão intimamente relacionados com este? Quais são as diferenças importantes? Com quais outros padrões este deveria ser usado?

O catálogo de padrões de projeto

- **O catálogo contém 23 padrões de projeto.**
  - Abstract Factory (95): Fornece uma interface para criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.
  - **Adapter (140): Converte a interface de uma classe em outra interface esperada pelos clientes. O Adapter permite que certas classes trabalhem em conjunto, pois de outra forma seria impossível por causa de suas interfaces incompatíveis.**
  - Bridge (151): Separa uma abstração da sua implementação, de modo que as duas possam variar independentemente.
  - Builder (104): Separa a construção de um objeto complexo da sua representação, de modo que o mesmo processo de construção possa criar diferentes representações.
  - Chain of Responsibility (212): Evita o acoplamento do remetente de uma solicitação ao seu destinatário, dando a mais de um objeto a chance de tratar a solicitação. Encadeia os objetos receptores e passa a solicitação ao longo da cadeia até que um objeto a trate.

O catálogo de padrões de projeto

- Command (222): Encapsula uma solicitação como um objeto, desta forma permitindo que você parametrize clientes com diferentes solicitações, enfileire ou registre (log) solicitações e suporte operações que podem ser desfeitas.
- **Composite (160): Compõe objetos em estrutura de árvore para representar hierarquias do tipo partes-todo. O Composite permite que os clientes tratem objetos individuais e composições de objetos de maneira uniforme.**
- Decorator (170): Atribui responsabilidades adicionais a um objeto dinamicamente. Os decorators fornecem uma alternativa flexível a subclasses para extensão da funcionalidade.
- Façade (179): Fornece uma interface unificada para um conjunto de interfaces em um subsistema. O Façade define uma interface de nível mais alto que torna o subsistema mais fácil de usar.

O catálogo de padrões de projeto

- Factory Method (112): **Define uma interface para criar um objeto, mas deixa as subclasses decidirem qual classe a ser instanciada. O Factory Method permite a uma classe postergar a instanciação às subclasses.**
- Flyweight (187): Usa compartilhamento para suportar grandes quantidades de objetos, de granularidade fina, de maneira eficiente.
- Interpreter (231): Dada uma linguagem, define uma representação para sua gramática juntamente com um interpretador que usa a representação para interpretar sentenças nessa linguagem.
- Iterator (244): Fornece uma maneira de acessar seqüencialmente os elementos de uma agregação de objetos sem expor sua representação subjacente.
- Mediator (257): Define um objeto que encapsula a forma como um conjunto de objetos interage. O Mediator promove o acoplamento fraco ao evitar que os objetos se refiram explicitamente uns aos outros, permitindo que você varie suas interações independentemente.

O catálogo de padrões de projeto

- Memento (266): Sem violar o encapsulamento, captura e externaliza um estado interno de um objeto, de modo que o mesmo possa posteriormente ser restaurado para este estado.
- **Observer (274): Define uma dependência um-para-muitos entre objetos, de modo que, quando um objeto muda de estado, todos os seus dependentes são automaticamente notificados e atualizados.**
- Prototype (121): Especifica os tipos de objetos a serem criados usando uma instância prototípica e criar novos objetos copiando esse protótipo.
- Proxy (198): Fornece um objeto representante (surrogate), ou um marcador de outro objeto, para controlar o acesso ao mesmo.
- Singleton (130): Garante que uma classe tenha somente uma instância e fornece um ponto global de acesso para ela.
- State (284): Permite que um objeto altere seu comportamento quando seu estado interno muda. O objeto parecerá ter mudado de classe.

O catálogo de padrões de projeto

- Strategy (292): Define uma família de algoritmos, encapsula cada um deles e os torna intercambiáveis. O Strategy permite que o algoritmo varie independentemente dos clientes que o utilizam.
- Template Method (301): Define o esqueleto de um algoritmo em uma operação, postergando a definição de alguns passos para subclasses. O Template Method permite que as subclasses redefinam certos passos de um algoritmo sem mudar sua estrutura.
- Visitor (305): Representa uma operação a ser executada sobre os elementos da estrutura de um objeto. O Visitor permite que você defina uma nova operação sem mudar as classes dos elementos sobre os quais opera.

Organizando o catálogo

- Os padrões de projeto variam na sua granularidade e no seu nível de abstração.
- A classificação ajuda a aprender os padrões mais rapidamente, bem como direcionar esforços na descoberta de novos.

- **Critério 1 – Finalidade:** reflete o que um padrão faz (criação, estrutural ou comportamental).

Organizando o catálogo

- Os **padrões de criação** se preocupam com o processo de criação de objetos.
- Os **padrões estruturais** lidam com a composição de classes ou de objetos.
- Os **padrões comportamentais** caracterizam as maneiras pelas quais classes ou objetos interagem e distribuem responsabilidades.

Organizando o catálogo

- **Critério 2 – Escopo**: especifica se o padrão se aplica primariamente a classes ou a objetos.
  - Os **padrões para classes** lidam com os relacionamentos entre classes e suas subclasses.
  - Os **padrões para objetos** lidam com relacionamentos entre objetos que podem ser mudados em tempo de execução e são mais dinâmicos.

Organizando o catálogo

- Os **padrões de criação voltados para classes** deixam alguma parte da criação de objetos para subclasses, enquanto que os **padrões de criação voltados para objetos** postergam esse processo para outro objeto.
- Os **padrões estruturais voltados para classes** utilizam a herança para compor classes, enquanto que os **padrões estruturais voltados para objetos** descrevem maneiras de montar objetos.
- Os **padrões comportamentais voltados para classes** usam a herança para descrever algoritmos e fluxo de controle, enquanto que **os voltados para objetos** descrevem como um grupo de objetos coopera para executar uma tarefa que um único objeto não pode executar sozinho.

Organizando o catálogo

Tabela 1.1 O espaço dos padrões de projeto

		Propósito		
		De criação	**Estrutural**	**Comportamental**
Escopo	**Classe**	Factory Method (112)	Adapter (class) (140)	Interpreter (231) Template Method (301)
	Objeto	Abstract Factory (95) Builder (104) Prototype (121) Singleton (130)	Adapter (object) (140) Bridge (151) Composite (160) Decorator (170) Façade (179) Flyweight (187) Proxy (198)	Chain of Responsibility (212) Command (222) Iterator (244) Mediator (257) Memento (266) Observer (274) State (284) Strategy (292) Visitor (305)

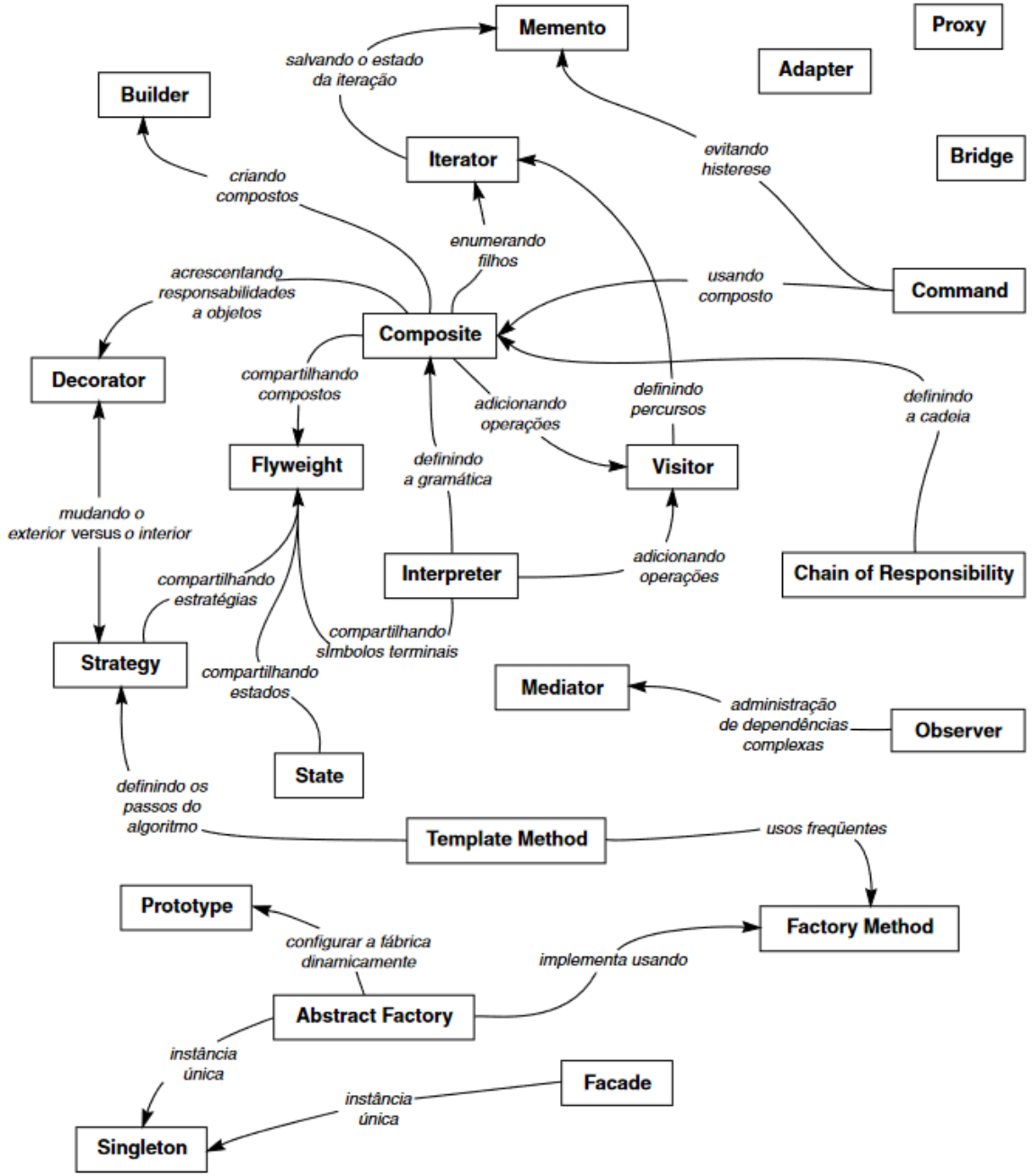

Figura 1.1 Relacionamentos entre padrões de projeto.

Como selecionar um padrão de projeto

- Considere como padrões de projeto solucionam problemas de projeto.
- Examine a Intenção.
- Estude como os padrões se interrelacionam.
- Estude padrões de finalidades semelhantes.
- Examine uma causa de reformulação de projeto.
- Considere o que deveria ser variável no seu projeto, o que você quer ser capaz de mudar sem reprojetar.

Como selecionar um pad

- Considere como padrões de projeto s
- Examine a Intenção.
- Estude como os padrões se interrelac
- Estude padrões de finalidades semell
- Examine uma causa de reformulação
- Considere o que deveria ser variável
ser capaz de mudar sem reprojetar.

Propósito	Padrão	Aspecto(s) que pode(m) variar
De Criação	Abstract Factory (95)	famílias de objetos-produto
	Builder (104)	como um objeto composto é criado
	Factory Method (112)	subclasse de objeto que é instanciada
	Prototype (121)	classe de objeto que é instanciada
	Singleton (130)	a única instância de uma classe
Estruturais	Adapter (140)	interface para um objeto
	Bridge (151)	implementação de um objeto
	Composite (160)	estrutura e composição de um objeto
	Decorator (170)	responsabilidade de um objeto sem usar subclasses
	Façade (179)	interface para um subsistema
	Flyweight (187)	custos de armazenamento de objetos
	Proxy (198)	como um objeto é acessado; sua localização
Comporta-mentais	Chain of Responsibility (212)	objeto que pode atender a uma solicitação
	Command (222)	quando e como uma solicitação é atendida
	Interpreter (231)	gramática e interpretação de uma linguagem
	Iterator (244)	como os elementos de um agregado são acessados, percorridos
	Mediator (257)	como e quais objetos interagem uns com os outros
	Memento (266)	que informação privada é armazenada fora de um objeto e quando
	Observer (274)	número de objetos que dependem de um outro objeto; como os objetos dependentes se mantêm atualizados
	State (284)	estados de um objeto
	Strategy (292)	um algoritmo
	Template Method (301)	passos de um algoritmo
	Visitor (305)	operações que podem ser aplicadas a (um) objeto(s) sem mudar sua(s) classe(s)

Padrões de Projeto

Um padrão de projeto deverá apenas ser aplicado quando a flexibilidade que ele oferece é realmente necessária.

Padrões de Projeto - Referência

- GAMMA, Erich. **Padrões de projeto : soluções reutilizáveis de software orientado a objetos. Porto Alegre : Bookman, 2005**. BRANT, John; Beck, Kent; Roberts, Don; Opdyke, William; Fowler, Martin.

Exercícios

- 1. O que você entende sobre padrão de projeto?
- 2. Quais são os quatro elementos essenciais de um padrão de projeto?
- 3. Comente em um parágrafo com pelo menos cinco linhas um padrão de projeto. Os nomes estão na imagem ao lado.

- GAMMA, Erich. Padrões de projeto : soluções reutilizáveis de software orientado a objetos. Porto Alegre : Bookman, 2005. BRANT, John; Beck, Kent; Roberts, Don; Opdyke, William; Fowler, Martin.

Padrão
Abstract Factory (95)
Builder (104)
Factory Method (112)
Prototype (121)
Singleton (130)
Adapter (140)
Bridge (151)
Composite (160)
Decorator (170)
Façade (179)
Flyweight (187)
Proxy (198)
Chain of Responsibility (212)
Command (222)
Interpreter (231)
Iterator (244)
Mediator (257)
Memento (266)
Observer (274)
State (284)
Strategy (292)
Template Method (301)
Visitor (305)