Comentários finais sobre Design Patterns

Como selecionar um Design Pattern?

Considerando como Design Patterns resolvem Problemas de Design

- Vimos aqui uma lista de situações que forçam um redesign
- Listamos novamente os problemas juntamente com os Design Patterns que ajudam a resolvê-los
- Criar um objeto especificando sua classe explicitamente
  - Abstract Factory, Factory Method, Prototype
- Dependência de operações específicas
  - Chain of Responsibility, Command
- Dependência de plataformas de hardware ou software específicas
  - Abstract Factory, Bridge
- Dependência da representação ou implementação de objetos
  - Abstract Factory, Bridge, Memento, Proxy
- Dependências algorítmicas
  - Builder, Iterator, Strategy, Template Method, Visitor
- Acoplamento forte
  - Abstract Factory, Bridge, Chain of Responsibility, Command, Façade, Mediator, Observer
- Extensão de funcionalidade através da herança
  - Bridge, Chain of Responsibility, Composite, Decorator, Observer, Strategy

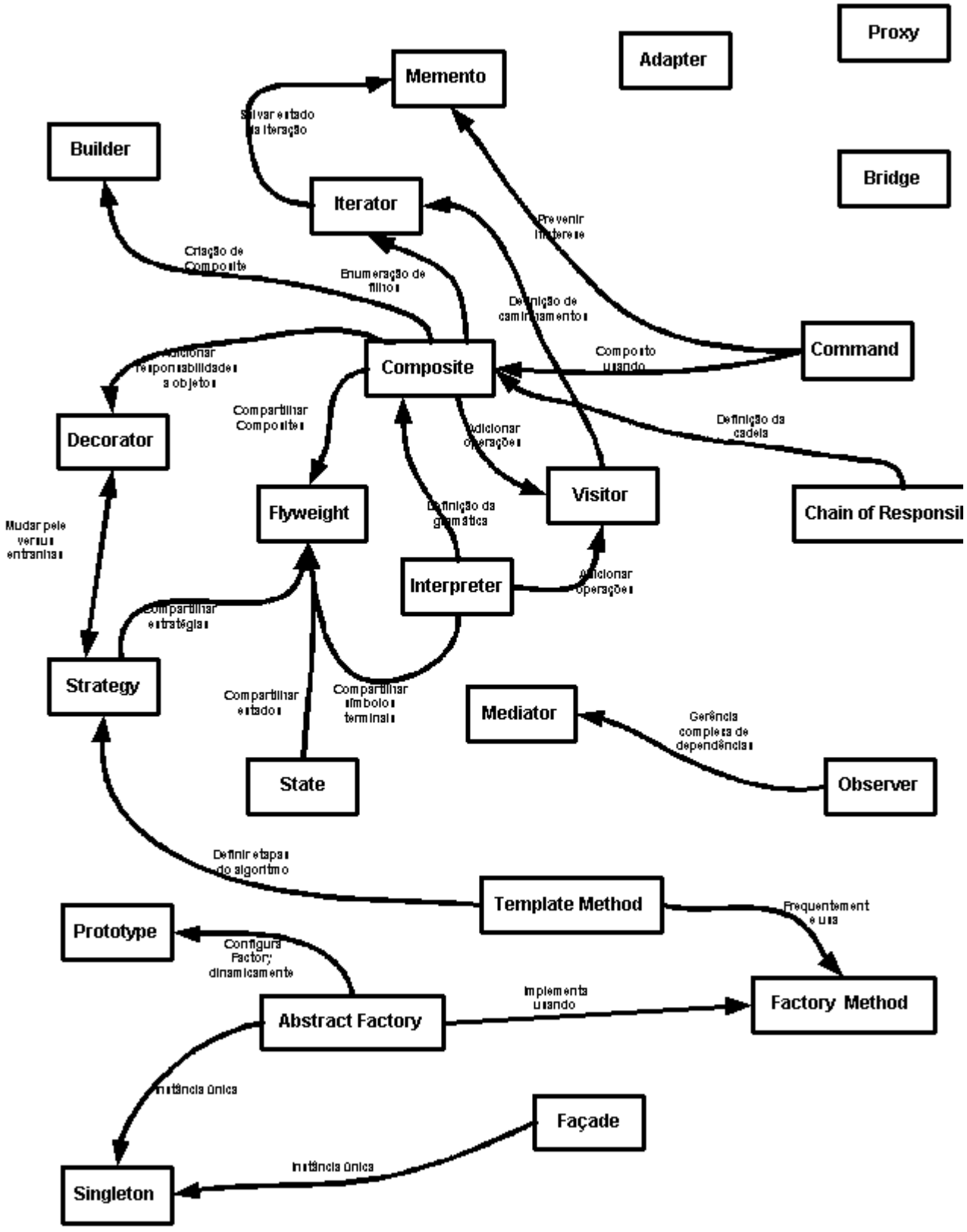
- Dificuldade de alterar classes convenientemente
  - Adapter, Decorator, Visitor

Examinando os Objetivos de cada Pattern

- Veja a lista de todos os objetivos aqui

Estudando o relacionamento entre Design Patterns

- Ver figura abaixo



Examinando no livro da GoF as semelhanças e diferenças entre padrões

- Ver o final da apresentação de cada padrão
- Ver a introdução aos 3 capítulos do catálogo

- Há uma discussão das semelhanças

Considerando o que deve ser variável do Design

- Em vez de verificar as causas de redesign, podemos fazer o contrário e verificar o que gostaríamos que fosse modificável *sem* redesign
- O importante aqui é *Encapsular o conceito que varia*
- Ver tabela abaixo

Propósito	Design Pattern	Aspecto(s) que varia(m) (O que pode mudar sem redesign)
Criação	Abstract Factory	Famílias de objetos-produto
	Builder	Como um objeto composto é criado
	Factory Method	Subclasse do objeto que é instanciado
	Prototype	A classe do objeto que é instanciado
	Singleton	A única instância de uma classe
Estrutura	Adapter	A interface para acessar um objeto
	Bridge	A implementação de um objeto
	Composite	A estrutura e composição de um objeto
	Decorator	As responsabilidades de um objeto (sem uso de herança)
	Façade	A interface de um subsistema
	Flyweight	O custo de armazenamento de objetos
	Proxy	Como um objeto é acessado; sua localização
Comportamento	Chain of Responsibility	O objeto que pode atender a um pedido
	Command	Quando e como um pedido é atendido
	Interpreter	Gramática e interpretação de uma linguagem
	Iterator	Como os elementos de uma coleção são acessados, varridos
	Mediator	Como e quais objetos interagem uns com os outros
	Memento	Qual informação privada é armazenada fora de um objeto, e quando
	Observer	O número de objetos que dependem de um outro objeto; como os objetos dependentes ficam atualizados
	State	Os estados de um objeto
	Strategy	Um algoritmo

	Template Method	Certas etapas de um algoritmo
	Visitor	Operações que podem ser aplicadas a objetos sem mudar suas classes

Finalmente ... O que esperar de Design Patterns

- Um vocabulário comum de Design
  - "Vamos usar um Observer aqui ..."
- Ajuda no entendimento de software existente
  - Desde que se especifique os Design Patterns usados
- Patterns farão de você um projetista melhor
- Um alvo para *Refactoring*
  - Mais sobre isso aqui
  - Consiste na reorganização de software existente para melhorar sua qualidade

programa