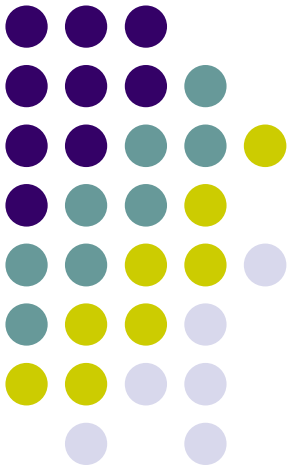
Introdução a Padrões de Software

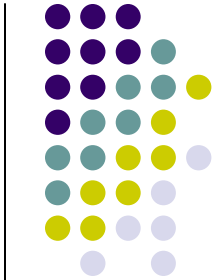
Prof. Alberto Costa Neto
DComp/UFS



O que é um Padrão de Software?

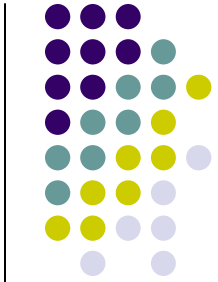
- Gênesis
  - Christopher Alexander, et. al. 1977: “Cada padrão descreve um problema que ocorre sempre em nosso ambiente, e então descreve o núcleo de uma solução para o problema, de maneira que você pode usar esta solução um milhão de vezes, mas sem fazê-la do mesmo jeito duas vezes.”
- Apropriada para prédios e pontes
- Aplicada em software somente nas últimas décadas
- Padrões de projeto são catálogos sistemáticos
  - nomes, explicações e avaliações

O que é um Padrão de Software?



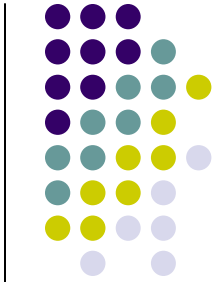
- “Um padrão é a abstração de uma forma concreta que ocorre muitas vezes em contextos não arbitrários específicos.” Riehle and Zullighoven, 1996
- “Cada padrão é uma regra de três partes, que expressa uma relação entre um certo contexto, um certo sistema de forças que ocorre repetidamente naquele contexto, e uma certa configuração de software que permite que estas forças os resolvam” Gabriel 1996.
- “Resolve um problema. É um conceito provado. A solução não é óbvia. Descreve um relacionamento. Os padrões têm um componente humano significativo.” Coplien 1996.

O que é um Padrão de Software?



- O que não é
  - Projeto que pode ser codificado diretamente
  - Projeto complexo de domínio específico
- Descrição de objetos comunicantes e classes que são customizadas para resolver um problema de projeto geral num contexto particular
- Nomeia, abstrai e identifica aspectos chaves de projetos comuns
- Identifica classes e objetos participantes, papéis e colaborações, e distribuição de responsabilidades

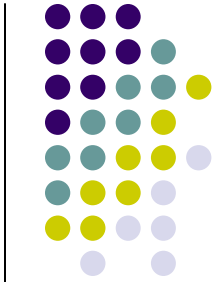
Características



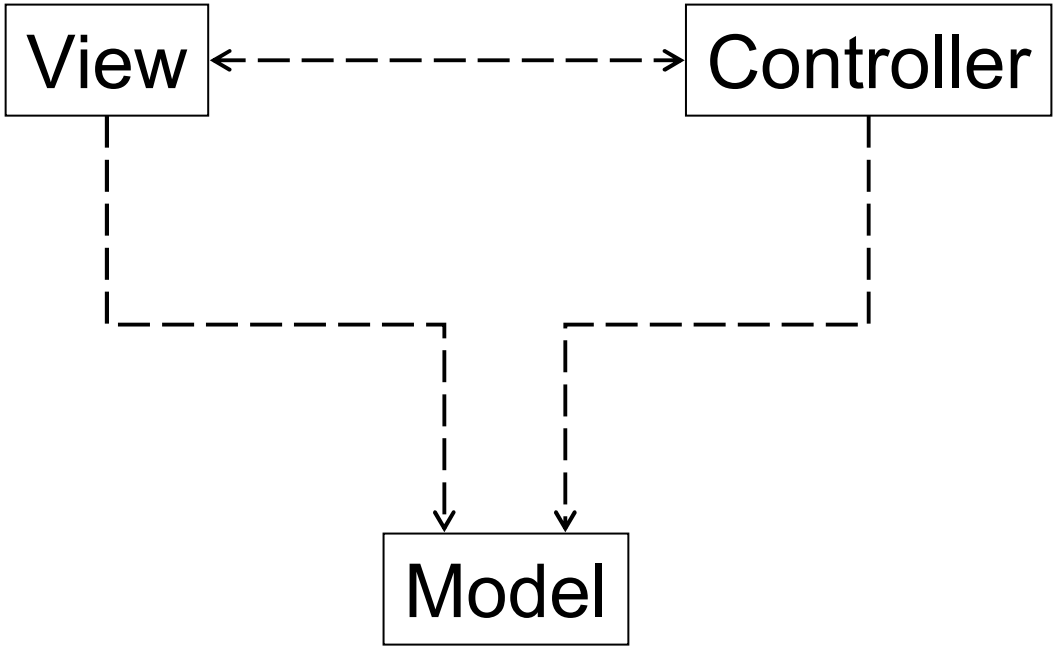
- Observados através da experiência
- Escritos num formato estruturado
- Evitam a reinvenção da roda
- Padrões para diferentes níveis de abstração
- Suportam melhorias contínuas
- Reutilizáveis
- Comunicam designs e melhores práticas
- Em conjunto permitem resolver grandes problemas

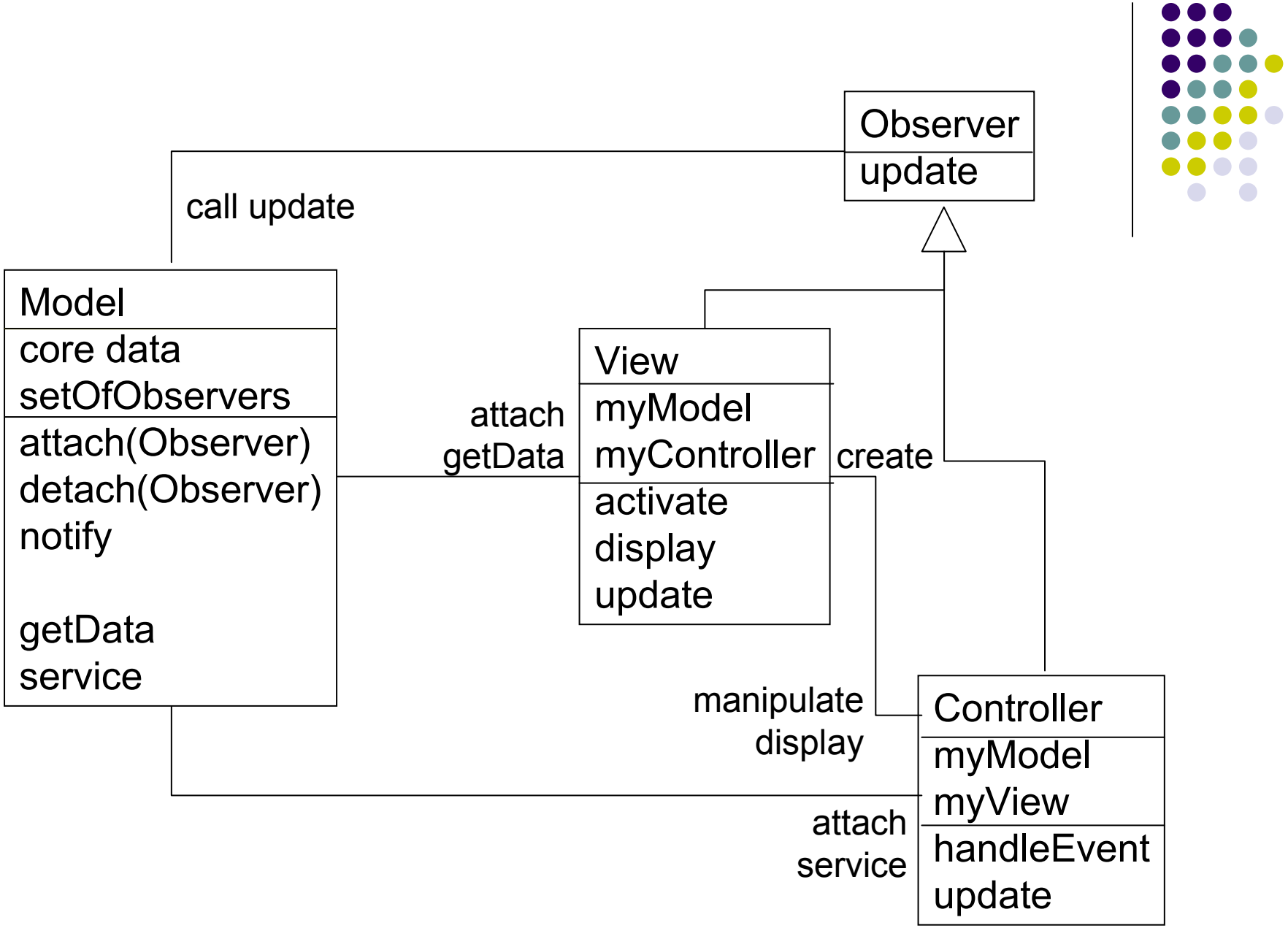
Exemplo: Padrão Arquitetural MVC

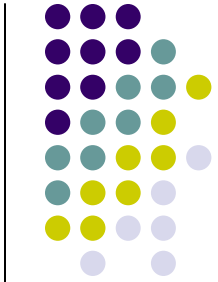
- Contexto
  - Adequado para Sistemas com interface homem-máquina
- Problema:
  - Interfaces estão sujeitas a mudanças
  - Acoplamento forte entre interface e o núcleo funcional implicam falta de flexibilidade
  - Mudanças na interface devem ser fáceis e possíveis no tempo de execução
  - Adaptar ou mudar a interface não deve causar impactos no núcleo funcional da aplicação



- Solução: Dividir uma aplicação interativa em três áreas: processamento, saída e entrada
  - Model
    - Dados e funções do núcleo funcional
  - View
    - Apresenta informação ao usuário
  - Controller
    - Cada *view* tem um componente controlador
    - Transforma as entradas em requisições
    - Usuário interage somente através de controladores

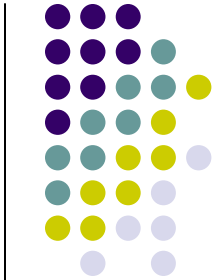

View
Controller
Model


Observer
update
call update
Model
core data
setOfObservers
attach(Observer)
detach(Observer)
notify
getData
service
attach
getData
View
myModel
myController
activate
display
update
create
manipulate
display
Controller
myModel
myView
handleEvent
update
attach
service

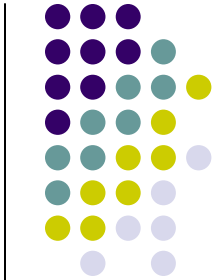


- Conseqüências
  - É possível ter várias visões para um único modelo
  - Se o modelo muda na execução, as visões refletem a mudança
  - Visões podem ser substituídas ou alteradas sem precisar mexer com o modelo
    - Visões dependem do modelo mas não vice-versa

O que é um Padrão?

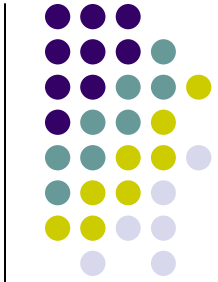


- Aborda um problema de projeto recorrente que aparece em situações específicas, e apresenta uma solução
- Documenta experiências de designs bem sucedidas
- Identificam e especificam abstrações que estão num nível mais alto do que classes ou componentes individuais
- Provêm um vocabulário e entendimento comum para princípios de projeto



- São um meio de documentação de arquiteturas de software
- Suportam a construção de software com propriedades definidas
- Ajudam a lidar com a complexidade do software

Elementos essenciais de um padrão



GoF

- Nome
- Problema
- Solução
- Conseqüências

POSA

- Contexto
- Problema
- Solução

Categorização

- Padrões de Análise
- Padrões Arquiteturais
- Padrões de Projeto
- Idiomas

- Classe
- Objeto

- De criação
- Estruturais
- Comportamentais

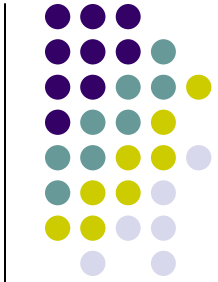
- De concorrência
- De Sistemas de Informação (enterprise applications)
- De comunicações
- .....

Categorização dos Padrões POSA

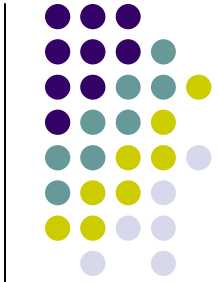
- Da camada de Persistência
- Da camada de Apresentação
- ...

Categorização dos Padrões de Projeto GoF

Categorização pelo nível de abstração (POSA)

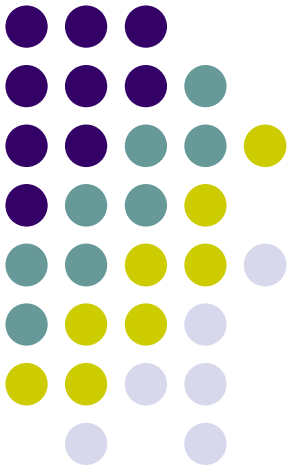


- Padrões de Análise:
  - Descrevem grupos de conceitos que representam construções comuns na modelagem do domínio
- Padrões de Arquitetura:
  - Descrevem a estrutura e os relacionamentos fundamentais dos componentes de um sistema de software
  - Provêm um conjunto de subsistemas, especificam suas responsabilidades e incluem regras e guias para organizar os relacionamentos entre eles

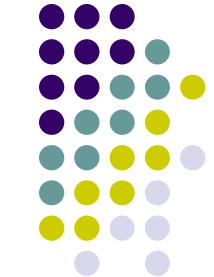


- Padrões de Projeto:
  - Identificam as relações internas entre um grupo de componentes de software e descreve suas responsabilidades, colaborações e relações estruturais.
- Idiomas
  - Descrevem como implementar aspectos específicos de um sistema de software numa dada linguagem de programação

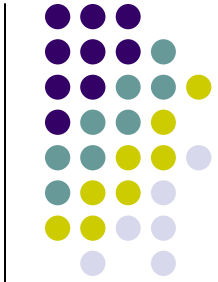
Arquitetura e Padrões



Arquitetura do Software

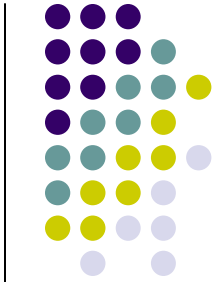


- A Arquitetura de Software é uma descrição dos subsistemas e componentes e as relações entre eles.
- Subsistemas e componentes são especificados através de diferentes visões para mostrar propriedades funcionais e não funcionais relevantes do sistema
- A arquitetura de software é um artefato, é o resultado do projeto do software



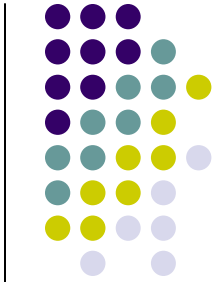
- Segundo Fowler
  - “The highest level breakdown of a systems in parts”
  - “Decisions that are hard to change”
- Compreende questões como
  - Decisões de projeto (early)
  - Escalabilidade e desempenho
  - Organização e estrutura geral de controle
  - Protocolos de comunicação e sincronização
  - Atribuição de responsabilidades a componentes de projeto

Componente



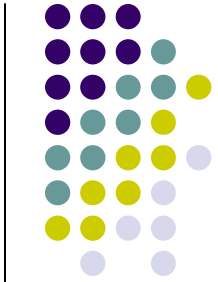
- Parte encapsulada de um sistema
  - Possui uma interface
  - *Building blocks* para estruturar o sistema
  - Ao nível de linguagens: módulos, classes, objetos ou um conjunto de funções relacionadas
- De natureza diferente
  - Exemplo: um processo ou uma biblioteca
- Em alguns casos, usa-se o termo componente de forma vaga

Categorizando componentes



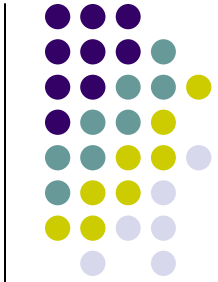
- Elementos de processamento
- Elementos de dados
- Elementos de conexão
- Controladores
- Coordenadores
- Interfaces
- Provedores de Serviços
- Repositório de informação
- Componente de estruturação

Relações



- Denota uma conexão entre componentes
- Estática ou Dinâmica
- Exemplos
  - Estática: agregação e herança
  - Dinâmica: criação de objetos, comunicação entre objetos, transferência de dados, cliente/servidor, observer/observable, ...
- Relações têm impacto na qualidade
  - Exemplo: flexibilidade é melhor suportado por relações que suportam variação de componentes

Visão

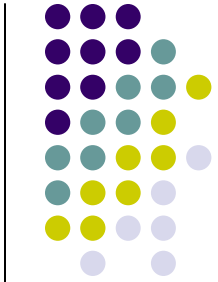


- Representa um aspecto parcial da arquitetura que mostra propriedades específicas do sistema
- Exemplos de visões:
  - Diagrama de classes – visão de estrutura estática
  - Diagrama de objetos – visão de estrutura dinâmica
  - Diagrama de interação – visão de comunicação entre componentes

Uma proposta de visões para descrever uma arquitetura

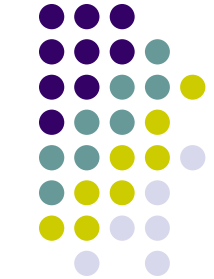
- Arquitetura conceitual: componentes, conectores, ..
- Arquitetura de Módulos: subsistemas, módulos, exports, imports, ...
- Arquitetura de Código: arquivos, diretórios, bibliotecas, includes, ...
- Arquitetura de execução: threads, processos, ...

Outra proposta



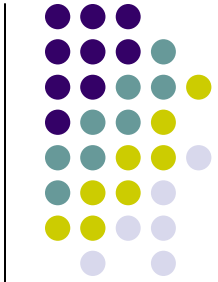
- Visão Lógica: o projeto do modelo de objetos ou um modelo correspondente tal como um diagrama E/R
- Visão de processo: aspectos de concorrência e sincronização
- Visão Física: mapeamento do software em hardware e aspectos de distribuição
- Visão de desenvolvimento: organização estática do software no ambiente de desenvolvimento

Padrões em Arquitetura de Software



- Padrões estão intimamente relacionados com arquitetura de software, análise, projeto e implementação de sistemas (O.O. e procedural)
- Padrões não são “novos”, e não são criados (artificialmente)
- São construídos sobre “velhos” princípios
- Objetivam construir software com propriedades não funcionais previsíveis

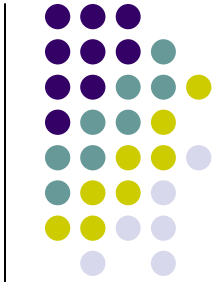
Padrões e Metodologias



- Padrões e Metodologias são sinérgicos
  - Metodologias definem o processo e oferecem passos e guias para construção de software com qualidade
  - Padrões oferecem técnicas para resolver problemas específicos, porém recorrentes
- Metodologias não são focadas em problemas, padrões são

Padrões e Processos de Software

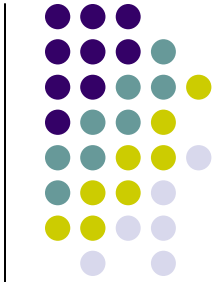
- O Processo não deve determinar estritamente como o projeto e a implementação procedem
- Padrões permitem tornar os ciclos mais previsíveis em processos iterativos
  - Padrões ajudam a produzir projetos melhores e mais estáveis
- Refactorings to patterns



- Em que ponto do desenvolvimento devemos usar os padrões?
  - Padrões arquiteturais nas etapas iniciais (alto nível)
  - Padrões de projeto nas etapas de projeto de nível médio
  - Idiomas na implementação

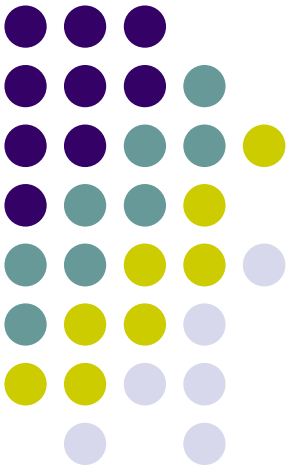
Padrões e Estilos Arquiteturais

"An architectural style defines a family of software systems in terms of their structural organization. An architectural style expresses components and the relationships between them, with the constraints of their application, and the associated composition and design rules for their construction" [POSA]

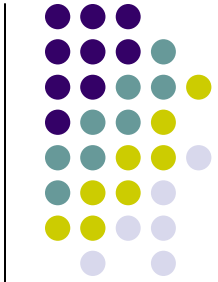


- Estilos arquiteturais podem ser capturados como padrões, porém:
  - Padrões são um conceito mais amplo
  - Estilos arquiteturais são independentes um do outro. Padrões dependem de outros padrões.
  - Padrões são mais orientados a problemas

Padrões de Projeto

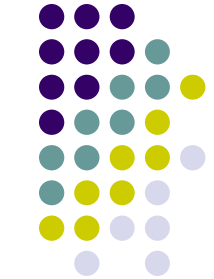


Conteúdo



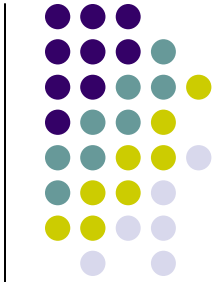
- Introdução
- O que é um Padrão de Projeto
- Descrevendo Padrões de Projeto
- Como Padrões de Projeto resolvem problemas de projeto

Introdução



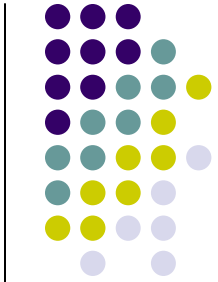
- Projetando software orientado a objetos
  - Projetar software OO não é uma tarefa muito simples.
  - Mais difícil ainda é desenvolver de forma a torná-los **reutilizáveis** em outras situações
  - Esses pedaços reutilizáveis de software podem ser considerados um **patrimônio**
    - Podem reduzir o custo/tempo de desenvolvimento de novos sistemas

Introdução



- **Criar componentes reutilizáveis requer mais tempo** (e dinheiro) do que soluções que resolvem o problema de imediato
  - É necessário que **projetemos pensando no futuro**, ou seja, tentando criar situações onde esses componentes possam ser reutilizados
  - Às vezes, acabamos **introduzindo características desnecessárias no momento**, mas que podem ser úteis futuramente

Introdução



- Os Padrões de Projeto (Design Patterns) ajudam a criar **software mais flexível e reutilizável**
- Permitem **entender de forma mais fácil** a solução aplicada, caso se conheça o Padrão de Projeto que foi aplicado
- Padrões de Projeto ajudam o projetista a **chegar de forma mais rápida a um projeto correto**, ou seja, um projeto que possua a flexibilidade e o nível de reutilização desejados

O que é um Padrão de Projeto

- Cada padrão descreve um problema que ocorreu várias vezes e o cerne da solução para aquele problema, de forma que essa solução possa ser reutilizada milhões de vezes em situações diferentes

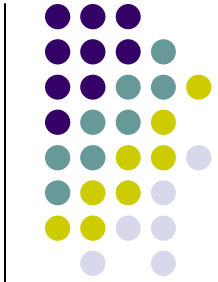
O que é um Padrão de Projeto

- É algo que descreve soluções simples e elegantes para problemas específicos no projeto de software orientado a objetos
- Refletem a experiência de diversos desenvolvedores na abordagem de certos tipos de problema

O que é um Padrão de Projeto

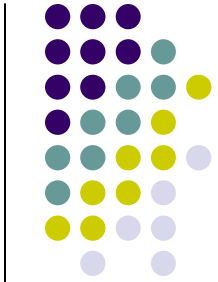
- **Não requerem**
  - Características não usuais das linguagens
  - Truques mirabolantes
- Podem ser implementados em linguagens OO comuns
- Há linguagens que se adaptam melhor a um Padrão de Projeto e acabam exigindo menos esforço para implementá-lo

Descrevendo Padrões de Projeto



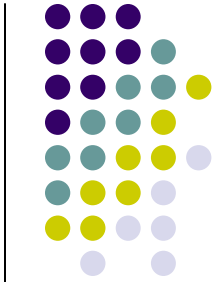
- Nome
- Problema
- Solução
- Conseqüências

Descrevendo Padrões de Projeto



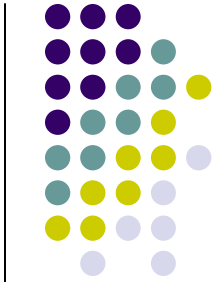
- Nome
  - Descreve em uma ou duas palavras o problema de projeto, suas soluções e conseqüências
  - Facilita o diálogo com outros desenvolvedores através dos nomes. Equivalente ao padrão de nomes de estruturas de dados, como ”Árvore Binária”, “Fila”, “Pilha”, etc

Descrevendo Padrões de Projeto



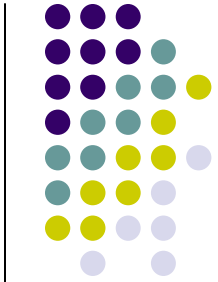
- Problema
  - Descreve quando aplicar o padrão
  - Descreve o problema e o contexto
  - Pode descrever problemas específicos de projeto
    - Exemplo: de que forma representar algoritmos como objetos?
  - Pode descrever estruturas de objetos ou de classes que são sintomas de um projeto inflexível
  - Às vezes, o padrão lista condições que devem ser satisfeitas para sua aplicação

Descrevendo Padrões de Projeto



- Solução
  - Descreve os elementos constituintes do projeto, seus relacionamentos, responsabilidades e colaborações
  - A solução não descreve um projeto ou implementação concretos porque um padrão é como um gabarito da solução para várias situações
  - Provê uma descrição abstrata de um problema de projeto e uma organização genérica de elementos (classes e objetos) que o solucionam

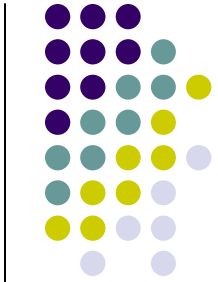
Descrevendo Padrões de Projeto



- Aplicabilidade
  - Indica em quais situações o Padrão de Projeto pode ser aplicado
  - Dá exemplos de projetos ruins que o Padrão de Projeto pode melhorar
  - Mostra como reconhecer tais situações

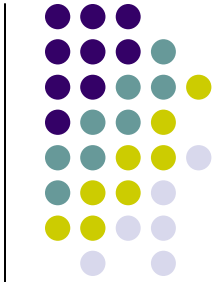
- Estrutura
  - Mostra, através de representações gráficas (um diagrama de classes UML, por exemplo), as classes, suas dependências e seus relacionamentos

Descrevendo Padrões de Projeto



- Participantes
  - As classes e/ou objetos participando no Padrão de Projeto e suas responsabilidades
- Colaboração
  - Como os participantes colaboram para cumprir as suas responsabilidades
- Implementação
  - Quais são as armadilhas, dicas ou técnicas que deve-se ter atenção durante a implementação

Descrevendo Padrões de Projeto



- Conseqüências
  - Os resultados e *trade-offs* da aplicação do padrão (espaço x tempo, por exemplo)
  - Analisa aspectos como: flexibilidade, portabilidade e extensibilidade
  - Listar as conseqüências explicitamente ajuda a entender e a avaliá-las

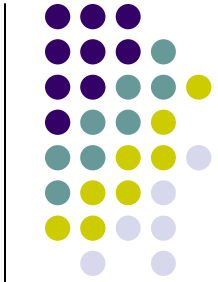
Como Padrões de Projeto resolvem problemas de projeto

- Encontrando os objetos apropriados
  - Muitos dos objetos existentes na fase de Projeto são oriundos da Análise
  - Há, entretanto, vários outros que surgem durante essa fase e dizem respeito a aspectos mais ligados à implementação do que às regras de negócio

Como Padrões de Projeto resolvem problemas de projeto

- Uma das tarefas mais difíceis é **decompor** o sistema **em objetos** da **forma mais apropriada**
- Também é importante considerar a distribuição dos dados e métodos entre os objetos de forma a respeitar critérios como:
  - granularidade, encapsulamento, dependência, flexibilidade, desempenho, evolução e reusabilidade
- É difícil atingir uma concordância total sobre essa decomposição

Como Padrões de Projeto resolvem problemas de projeto



- Os Padrões de Projeto ajudam a encontrar os objetos apropriados porque definem estruturas, relacionamentos e dependências que devem existir para solucionar o problema

Como Padrões de Projeto resolvem problemas de projeto

- Especificando a interface dos objetos
  - Os objetos são conhecidos através da sua interface. É através dela que se pode saber quais são os métodos disponíveis e como chamá-los
  - Uma interface não define a implementação de cada método. Isso permite que haja várias implementações para uma mesma interface

Como Padrões de Projeto resolvem problemas de projeto

- Padrões de Projeto ajudam a planejar as interfaces pois através deles é possível prever quais são os dados que podem ser obtidos e devem ser passados através de cada interface

Como Padrões de Projeto resolvem problemas de projeto

- Especificando implementações de objetos
  - Alguns Padrões de Projeto definem o comportamento de certos objetos (reutilização)
  - Outros permitem que o comportamento seja modificado (flexibilidade na troca de implementações)

Como Padrões de Projeto resolvem problemas de projeto

- Projetando para reutilizar
  - Padrões de Projeto permitem escolher antecipadamente quais pontos do sistema aceitam mudanças mais facilmente
  - Evitam modificar o projeto original, possivelmente afetando os clientes existentes