

Engenharia de Software

REUSO

em Projeto de Software

Design Patterns/Frameworks

Prof. Luís Fernando Garcia

LUIS@GARCIA.PRO.BR

REUSO

- **Desenvolver Software ...**
 - Lento
 - Difícil
 - Arriscado
 - Caro
- **“Solução” = REUSO (reutilização)**
- **Enfoque sistêmico (abrangente)**

REUSO (+)

Tabela 18.1 Benefícios do reuso de software

Benefício	Explicação
Confiança aumentada	Software reusado, experimentado e testado em sistemas de trabalho, deve ser mais confiável do que software novo, pois seus defeitos de projeto e implementação já foram encontrados e corrigidos.
Risco de processo reduzido	O custo de software existente já é conhecido, enquanto os custos de desenvolvimento são sempre um problema de avaliação. Isso é um fator importante para o gerenciamento de projetos porque reduz a margem de erro de estimativa de custo de projeto. Tal fato é especialmente verdadeiro quando componentes de software relativamente grandes, tais como subsistemas, são reusados.
Uso eficiente de especialistas	Em vez de fazer o mesmo trabalho repetidas vezes, esses especialistas podem desenvolver software reusável englobando seus conhecimentos.
Conformidade com padrões	Alguns padrões, tais como padrões de interface com o usuário, podem ser implementados como um conjunto de componentes reusáveis padronizados. Por exemplo, se os menus numa interface com o usuário são implementados usando componentes reusáveis, todas as aplicações apresentarão o mesmo formato de menu para o usuário. O uso de interfaces padronizadas com o usuário melhora a confiança porque os usuários provavelmente cometerão menos erros quando apresentados a uma interface familiar.
Desenvolvimento acelerado	A apresentação de um sistema para o mercado tão cedo quanto possível é muitas vezes mais importante do que os custos totais de desenvolvimento. O reuso de software pode tornar rápida a produção do sistema porque tanto o tempo de desenvolvimento quanto o de validação devem ser reduzidos.

REUSO

- **Mas ... REUSO ...**
- **Não é fácil ... Não é “automático”**
- **Curva de aprendizagem difícil**
- **Overhead**
- **Falta de motivação (humano)**

REUSO (-)

Tabela 18.2 Problemas com reuso

Benefício	Explicação
Custos de manutenção aumentados	Se o código-fonte de um sistema ou componente de software reusável não estiver disponível, então os custos de manutenção poderão ser aumentados, pois os elementos reusados do sistema podem tornar-se cada vez mais incompatíveis com as mudanças.
Falta de apoio de ferramenta	Os conjuntos de ferramentas CASE podem não apoiar o desenvolvimento com reuso. Pode ser difícil ou impossível integrar essas ferramentas a um sistema de biblioteca de componentes. O processo de software suportado por essas ferramentas pode não levar em conta o reuso.
Síndrome do não-inventado-aqui	Alguns engenheiros de software preferem reescrever componentes porque acreditam que podem aprimorá-los. Isso tem, em parte, a ver com responsabilidade e, em outra, com o fato de que a escrita do software original é vista como mais desafiadora do que reusar software de outras pessoas.
Criação e manutenção de uma biblioteca de componentes	Preencher uma biblioteca de componentes reusáveis e assegurar aos desenvolvedores de software que podem usar essa biblioteca pode ser caro. Nossas técnicas atuais de classificação, catalogação e recuperação de componentes de software são imaturas.
Procura, compreensão e adaptação de componentes reusáveis	Componentes de software precisam ser descobertos numa biblioteca, compreendidos e, às vezes, adaptados para trabalhar num novo ambiente. Os engenheiros devem estar razoavelmente confiantes de encontrar um componente na biblioteca antes de incluírem uma busca de componente como parte de seu processo normal de desenvolvimento.

REUSO (=)

- REUSO ...
- Cultura na/da empresa ...
- Convencimento .. Treinamento ...
- Investimento da empresa (\$\$)
- Patrimônio da empresa (\$\$\$\$)

REUSO

- **REUSO ... ONTEM**
- **Copy-and-Paste**
- **Subrotinas**
- **Bibliotecas**
- **OO**

REUSO

- **REUSO ... HOJE**
- **Design Patterns**
- **Frameworks**
- **Componentes**
- **E mais ...**

REUSO

Figura 18.1

Panorama de reuso.



REUSO

Tabela 18.3 Abordagens que apoiam o reuso de software

Abordagem	Descrição
Design patterns	Abstrações genéricas que ocorrem ao longo das aplicações são representadas como design patterns que mostram objetos abstratos e concretos e interações.
Desenvolvimento baseado em componentes	Sistemas desenvolvidos pela integração de componentes (conjuntos de objetos) que estão em conformidade com padrões de modelos e componentes. Isso é explicado no Capítulo 19.
Frameworks de aplicação	Conjuntos de classes abstratas e concretas podem ser adaptados e ampliados para criar sistemas de aplicações.
Empacotamento de sistemas legados	Sistemas legados (veja Capítulo 2) que podem ser 'empacotados' pela definição de um conjunto de interfaces e fornecimento de acesso a esses sistemas herdados através das interfaces.
Sistemas orientados a serviços	Sistemas desenvolvidos pela ligação de serviços compartilhados, que podem ser providos externamente.
Linhas de produtos de aplicação	Um tipo de aplicação generalizada com base em uma arquitetura comum de tal maneira que possa ser adaptada para clientes diferentes.
Integração de COTS	Sistemas desenvolvidos pela integração de sistemas de aplicações existentes.
Aplicações verticais configuráveis	Um sistema genérico projetado de tal maneira que pode ser configurado para as necessidades de clientes de sistemas específicos.
Bibliotecas de programa	Bibliotecas de classes e funções que implementam abstrações comumente usadas disponíveis para reuso.
Geradores de programa	Um sistema gerador que incorpora conhecimento de um determinado tipo de aplicação e pode gerar sistemas ou fragmentos de sistema no domínio.
Desenvolvimento de software orientado a aspectos	Componentes compartilhados integrados em uma aplicação em diferentes lugares programa é compilado.

REUSO

- **REUSO ... Fatores a considerar...**
 - *O cronograma de desenvolvimento para o software.*
 - *O ciclo de vida previsto do software.*
 - *O conhecimento, habilidades e experiência da equipe de desenvolvimento.*
 - *A importância do software e seus requisitos não funcionais.*
 - *O domínio da aplicação.*
 - *A plataforma de execução para o software.*

REUSO – tipos...

- **Reuso de Conceitos**
 - Design Patterns
 - Geradores de programa
- **Reuso Orientado a Aspectos**
- **Frameworks**
- **Reuso de Sistema de Aplicações**
- ...

REUSO de Conceito

- *Quando você reusa componentes de programa ou de projeto, você tem de seguir as decisões de projeto feitas pelo desenvolvedor original do componente.*
- *Isso pode limitar as oportunidades de reuso.*
- *Contudo, uma forma de reuso mais abstrata é o reuso de conceitos quando uma abordagem particular é descrita de maneira independente de implementação e, então, uma implementação é desenvolvida.*
- *As duas principais abordagens para concept reuse são:*
 - *Design patterns;*
 - *Geradores de programas.*

Design Pattern

- *“Cada Padrão de Projeto descreve um problema que ocorre frequentemente e então descreve o cerne da solução ao problema de forma a poder reusar a solução milhares de vezes em situações diferentes” ...*
- *“soluções já testadas para problemas de modelagem que ocorrem com freqüências em situações específicas” ...*
- *Origem = Engenheiro Civil Christopher Alexander – escreveu sobre suas experiências em resolver problemas de projetos em construções em geral.*

Design Pattern

- Reuso de IDÉIAS, não de código ...
- “Gabarito” de soluções para várias situações
- Isolar o que muda do que é igual

- Reuso de conhecimento abstrato sobre um problema e sua solução
- Padrão = descrição do problema e a essência da solução
- Chave:
 - Ser abstrato = Poder ser reusado em diferentes aplicações

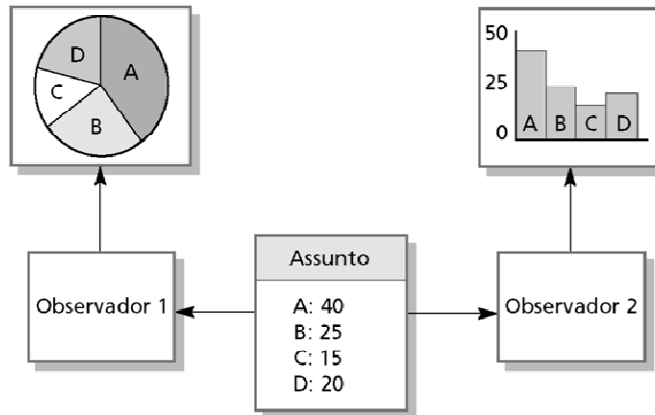
Design Pattern

- *Nome*
 - *Um identificador significativo padrão.*
- *Descrição do problema*
- *Descrição da solução*
 - *Não é um projeto concreto mas um template para uma solução de projeto que pode ser instanciado em maneiras diferentes.*
- *Conseqüências*
 - *Os resultados e os compromissos de aplicação do padrão.*

Design Pattern

Figura 18.2

Vários displays.



Design Pattern

- **Nome**
 - *Observer.*
- **Descrição**
 - *Separa o display de estado do objeto a partir do objeto em si.*
- **Descrição do problema**
 - *Usado quando displays múltiplos de estado são necessários.*
- **Descrição da solução**
 - *Veja prox. Slide com descrição UML.*
- **Conseqüências**
 - *Otimizações para melhorar o desempenho do display são impraticáveis.*

Design Pattern

Patterns tem relacionamentos – trabalham em conjunto

Exemplos de relacionamento entre patterns:

- MVC – modelo de dados, conjunto de visões e conjunto de controladores.
 - Aplicação original – interface HTML
 - Nova aplicação – WML e/ou XML
- DAO
- DTO
- Session Façade

Geradores

- Geradores de programas envolvem o reuso de padrões e algoritmos padronizados.
- Esses são embutidos no gerador e parametrizados pelos comandos do usuário. Então, um programa é automaticamente gerado.
- O reuso baseado em geradores é possível quando abstrações de domínio e seu mapeamento para código executável podem ser identificados.
- Uma linguagem específica de domínio é usada para compor e controlar estas abstrações.

Geradores

- Tipos de geradores de programa
 - Geradores de aplicações para processamento de dados de negócio;
 - Geradores de parser e analisadores léxicos para processamento de linguagem;
 - Geradores de código em ferramentas CASE.
- O reuso baseado em geradores é muito adequado em termos de custo, mas sua aplicabilidade é limitada a um número relativamente pequeno de domínios de aplicação.
- É mais fácil para os usuários finais desenvolverem programas usando geradores comparados a outras abordagens baseadas em componentes para reuso.

Geradores

Figura 18.4

Reuso baseado em geradores.



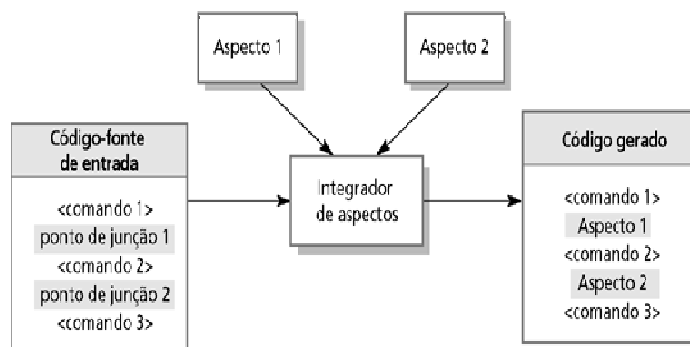
Aspectos

- O desenvolvimento orientado a aspectos soluciona um dos maiores problemas da engenharia de software – a separação de assuntos.
- Os assuntos não são freqüentemente associados simplesmente à funcionalidade da aplicação, mas são transversais – por exemplo, todos os componentes podem monitorar suas próprias operações, todos os componentes podem ter que manter proteção, etc.
- Assuntos transversais são implementados como aspectos e dinamicamente integrados dentro de um programa. O código do assunto é reusado e o novo sistema é gerado pelo integrador de aspectos.

Aspectos

Figura 18.5

Integração de aspectos.



Frameworks

Forma de REUSO que vai além do código

REUSO de análise, projeto e código

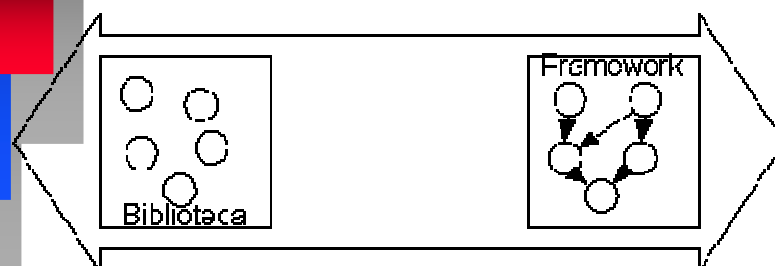
FRAMEWORK captura a funcionalidade comum a várias aplicações - pertencentes a um mesmo DOMÍNIO de problema

"Um framework provê uma solução para uma família de problemas semelhantes, Usando um conjunto de classes e interfaces que mostra como decompor a família de problemas e como objetos dessas classes colaboram para cumprir suas responsabilidades ... O conjunto de classes deve ser flexível e extensível para permitir a construção de várias aplicações com pouco esforço, especificando apenas as particularidades de cada aplicação"

Uma aplicação QUASE COMPLETA ...

Desenvolvedor = prover os pedaços que estão faltando = que são específicos para a aplicação

Frameworks



- Classes instanciadas pelo cliente
- Cliente chama funções
- Não tem fluxo de controle predefinido
- Não tem interação predefinida
- Não tem comportamento default

- Customização com sucesso ou composição
- Chama funções da "aplicação"
- Controla o fluxo de execução
- Define interação entre objetos
- Provê comportamento default

Frameworks

Framework x Design Patterns

- Design Patterns são mais abstratos – exemplos de uso
- Design Patterns são menores arquiteturalmente falando
- Design Patterns são menos específicos
- Framework – inclui código
- Framework – inclui vários design patterns
- Framework – tem domínio de aplicação particular

Frameworks

• Características de um Framework

- Reusável
- Documentado
- Fácil de usar
- Extensível
- Seguro
- Eficiente
- Completo



Frameworks

- **Vantagens no uso de Frameworks**

- Economia de tempo e dinheiro
- Maximização do reuso – análise, projeto, código e testes
- Herança (generalização) nos processos de manutenção
- Mais qualidade – menos bugs
- Compatibilidade entre aplicações
- Armazenamento do conhecimento dos especialistas



Frameworks

- **Desvantagens no uso de Frameworks**

- Complexidade de construção – planejamento, custo e tempo inicial
- Benefícios em longo prazo
- Restrições humanas

Frameworks

- Frameworks de infra-estrutura de sistema
 - Apoiam o desenvolvimento de infra-estruturas de sistemas, tais como comunicações, interfaces de usuário e compiladores.
- Frameworks de integração de middlewares
 - Padrões e classes que apóiam a comunicação e a troca de informações de componentes.
- Frameworks de aplicações empresariais
 - Apóiam o desenvolvimento de tipos específicos de aplicações, tais como sistemas de telecomunicações e financeiros.

Reuso de Aplicações

- Envolve o reuso de sistemas de aplicação inteiros pela configuração de um sistema para um ambiente ou pela integração de dois ou mais sistemas para criar uma nova aplicação.
- São feitas duas abordagens aqui:
 - Integração de produto COTS;
 - Desenvolvimento de linha de produto.

Reuso de Aplicações

- COTS - Commercial Off-The-Shelf systems.
- Sistemas COTS são, geralmente, sistemas de aplicação completos que oferecem uma API (Application Programming Interface).
- A construção de sistemas grandes pela integração de sistemas COTS é, atualmente, uma estratégia viável de desenvolvimento para alguns tipos de sistemas tais como sistemas de e-commerce.
- O benefício-chave é o desenvolvimento mais rápido da aplicação e, geralmente, a custos de desenvolvimento mais baixos.

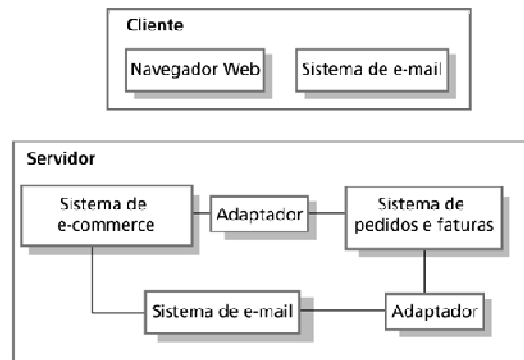
Reuso de Aplicações

- Quais produtos COTS oferecem a funcionalidade mais apropriada?
 - Pode haver diversos produtos similares que podem ser usados.
- Como os dados serão trocados?
 - Produtos individuais usam estruturas únicas de dados e formatos.
- Quais características do produto serão realmente usadas?
 - A maioria dos produtos têm mais funcionalidade do que é necessário. Você deve tentar negar acesso à funcionalidade que não for usada.

Reuso de Aplicações

Figura 18.7

Sistema de aquisição baseado em COTS.



Reuso de Aplicações

■ Problemas com reuso de COTS:

- Falta de controle sobre a funcionalidade e desempenho
 - Sistemas COTS podem ser menos eficientes do que parecem.
- Problemas com a interoperabilidade de sistemas COTS
 - Sistemas COTS diferentes podem assumir suposições diferentes, o que significa que a integração é difícil.
- Nenhum controle sobre a evolução do sistema
 - Vendedores de COTS, e não usuários de sistema, controlam a evolução.
- Suporte dos vendedores de COTS
 - Vendedores de COTS não podem oferecer apoio durante o ciclo de vida do produto.