Reuso de Software

Reuso de Software

- Na maioria das disciplinas de engenharia, os sistemas são projetados por meio de composição de componentes existentes
- Em engenharia de software, a situação ainda é diferente
  - Normalmente, grande parte de um novo sistema é construída do zero
  - Essa situação vem mudando, porém

Modalidades de Reuso

- **Reuso de sistemas**
  - Um sistema inteiro pode ser reusado (reuso de COTS)
  - Uma arquitetura + partes do sistema (famílias de aplicações)
- **Reuso de componentes**
  - Os componentes de uma aplicação, desde subsistemas até objetos simples, podem ser reusados
  - Componentes podem ser construídos com reuso em mente desde o início
- **Reuso de bibliotecas**
- **Reuso de conhecimento**
  - Princípios e padrões

Benefícios do 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 padroes	Alguns padroes, tais como padroes 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.

Problemas com 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 suposto 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.

O Panorama de Reuso

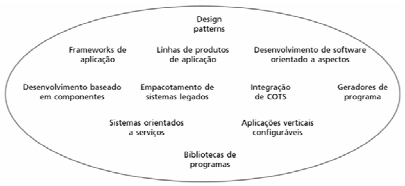
- Existem muitas abordagens diferentes para reuso
- O reuso é possível em uma variedade de níveis, desde funções simples até sistemas completos de aplicação.
- Reuso sempre deve ser feito com disciplina!
  - Reuso Acidental vs. Reuso Sistemático
- Desenvolvimento com Reuso vs. Desenvolvimento para Reuso

O Panorama de Reuso

- Existem muitas abordagens diferentes para reuso
- O reuso é possível em uma variedade de níveis, desde funções simples até sistemas completos de aplicação.
- Reuso sempre deve ser feito com disciplina!
  - Reuso Acidental vs. Reuso Sistemático
- Desenvolvimento com Reuso vs. Desenvolvimento para Reuso

O Panorama de Reuso

Figura 18.1
Panorama de reuso.



Abordagens de reuso

Tabela 18.3 *Abordagens que apóiam 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.

Fatores de planejamento de reuso

- 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 de qualidade
- O domínio da aplicação
- A plataforma de execução para o software
- A abordagem de reuso empregada

Frameworks

- *Frameworks* são um projeto de sistema ou subsistema feito de uma coleção de classes e as interfaces entre elas
- O *framework é instanciado através da implementação de classes concretas que estendem certas partes abstratas do framework*
  - *Classes abstratas e interfaces => Hotspots*
- *Frameworks* são entidades moderadamente complexas
  - Esforço para compreendê-los
  - Podem promover grande economia de esforço

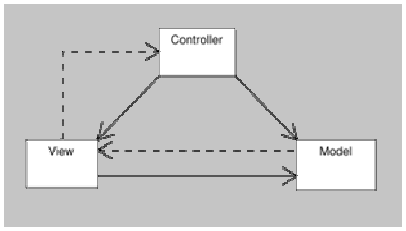
Classes de *framework*

- *Frameworks* de infra-estrutura de sistema
  - Apóiam o desenvolvimento de partes fundamentais de um sistemas, tais como comunicações, interfaces de usuário e compiladores.
- *Frameworks* de integração/*middleware*
  - Padrões e classes que apóiam a comunicação e a troca de informações de componentes.
- *Frameworks* de domínio específico
  - Apóiam o desenvolvimento de tipos específicos de aplicações, tais como sistemas médicos, de telecomunicações e financeiros

Exemplo: Modelo-Visão-Controlador

- *Framework* de infra-estrutura de sistema para projeto de GUI.
- Permite múltiplas apresentaçõs de um objeto e interações separadas com essas apresentações.
- Similar ao padrão *Observer*
- Exemplos de frameworks MVC:
  - Struts, Eclipse, Smalltalk Browser.

Modelo-Visão-Controlador



Reuso de Sistemas

- Reuso de aplicações inteiras
  - Pela configuração de um sistema para um ambiente
  - Pela integração de dois ou mais sistemas para criar uma nova aplicação.
- Exemplos:
  - Integração de componentes COTS
  - Linhas de produtos de software

Reuso de Componentes COTS

- COTS - *Commercial Off-The-Shelf systems*.
  - Geralmente são sistemas de aplicação completos que oferecem uma API
- É uma estratégia viável de desenvolvimento para alguns tipos de sistemas tais como os de e-*commerce*.
- O benefício-chave é o desenvolvimento mais rápido da aplicação
  - Geralmente com um custo menor
- Exige uma etapa de exploração

Escolha de projeto COTS

- 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

Exemplos de Componentes COTS

- No cliente, programas de *email* e de *Web browsing* padrão são usados.
- No lado do servidor:
  - Servidores Web
  - Sistemas gerenciadores de banco de dados
- Sistemas específicos de domínio
  - *Constraint Solvers*
  - *Sistemas para a emissão de faturas*

Problemas de integração de sistemas COTS

- Falta de controle sobre funcionalidades e características de qualidade
  - Sistemas COTS podem ser menos eficientes do que parecem.
- Problemas com a interoperabilidade
  - Sistemas COTS diferentes podem fazer suposições diferentes
- Nenhum controle sobre a evolução do sistema
  - Vendedores de COTS, e não usuários de sistema, controlam a evolução
- Suporte dos vendedores de COTS