Componentes

Componentes

- Como enfatizado nas aulas anteriores, reutilização é essencial. Utilizar uma **classe individual** em múltiplas aplicações é um tipo bem sucedido de reutilização. Entretanto reutilizar classes individuais não é o suficiente. Precisamos reutilizar objetos de classes, coleções de classes e recursos adicionais como imagens. Também precisamos ser capazes de consultar essas montagens em tempo de execução para tirar o máximo de proveito delas.

Componentes

- Um tipo comum de reutilização é aquele no qual nós alteramos o elemento sendo reutilizado. Uma analogia é a maneira como os construtores utilizam portas prontas: eles podem pintá-las, colocar pregos nelas para fixá-las, porém o modelo da porta não é alterado. A tecnologia de componentes utiliza essa abordagem.

Componentes

- Ao usarmos componentes do tipo portas, telhados, janelas em uma construção podemos simplesmente substituir tais componentes em uma reforma. Por outro lado, quando o projeto e a construção de uma casa não se baseiam em componentes, tudo deve ser construído integralmente. Por exemplo, cada janela deve ser construída a partir do zero enquanto a parede que a contém fosse construída – dificilmente isso pode ser considerado um plano de construção prático, pois nesse caso poderíamos dizer que estaríamos re-inventando a roda.

Componentes

- No exemplo abaixo toda a parte circulada deverá que ser alterada, no caso da substituição das janelas e não seja utilizado um componente.

Componentes

- Já com a utilização de componentes, pouca seria a mudança para substituir as janelas, veja na figura abaixo a área que deve ser afetada.

Componentes

- Mas afinal, o que é componente? Muitas definições de componentes foram estabelecidas tais como
- Um componente é a implementação de um software que possa ser executado em um dispositivo físico ou lógico devendo obedecer determinadas regras de modo que interaja de maneira praticável podendo ser reutilizado em diferentes ambientes.

Componentes

- Um conjunto de componentes de softwares padronizados, pré-construídos e disponibilizados enquadrando-se a um estilo arquitetural para um domínio de aplicação constitui em uma solução mais fácil de montar e menos dispendioso de construir do que um sistema confeccionado a partir de partes discretas.
- Vamos ficar com a mais geral de D`Souza:

  "Um pacote coerente de artefatos de software que pode ser desenvolvido independentemente e entregue como unidade e que pode ser composto, sem mudança, com outros componentes para construir algo maior."

Componentes

- Usando essa definição, um componente pode incluir:
  - Código executável
  - Código fonte
  - Projetos (designs)
  - Especificações
  - Testes
  - Documentação
  - Etc.

Componentes

- Os componentes podem gerar entidades, em geral instâncias, e essas instâncias podem ser modificadas.
- Retomando ao nosso exemplo, observe que uma janela particular (instância) poderia ser pintada: em outras palavras, o valor da propriedade "cor" de janela poderia ser alterado. O que ele não altera é o modelo de janela que utiliza.

Componentes

- Que tal uma variação quanto a um modelo de janela? Por exemplo, podemos ter um modelo *Janelas Tudibom* 2009 com 1m x 1,30m, sucedido por um modelo *Janela Tudibom* 2010 com 1m x 1,30m. Essas janelas estão relacionadas pelo fato de que a última é uma nova versão do componente anterior.
- A "Especificação de Linguagem de Modelagem" (*Modeling Language Specification*) do Object Management Group define um componente como uma parte física substituível de um sistema que empacota a implementação e fornece a realização de um conjunto de interface.

Componentes

- Um componente representa uma parte física da implementação de um sistema, incluindo o código de software ou equivalentes, como scripts ou arquivos de comando.
- Embora talvez pareça nessa lista que qualquer coisa poderia ser um componente, observe a restrição que é imposta: que eles sejam utilizáveis sem modificações. Se uma pessoa alterar um componente, o resultado é considerado como um componente diferente.

Componentes

- Componentes podem ser feitos de...
  - Código-fonte
    - Classes - uma ou mais, possivelmente relacionadas
  - Código executável
    - Código-objeto
    - Código-objeto virtual
  - Outros arquivos
    - Imagens, texto, índices, etc.

Componentes

1 – Em que consistem os componentes?

1.1 – Propriedades, métodos, eventos

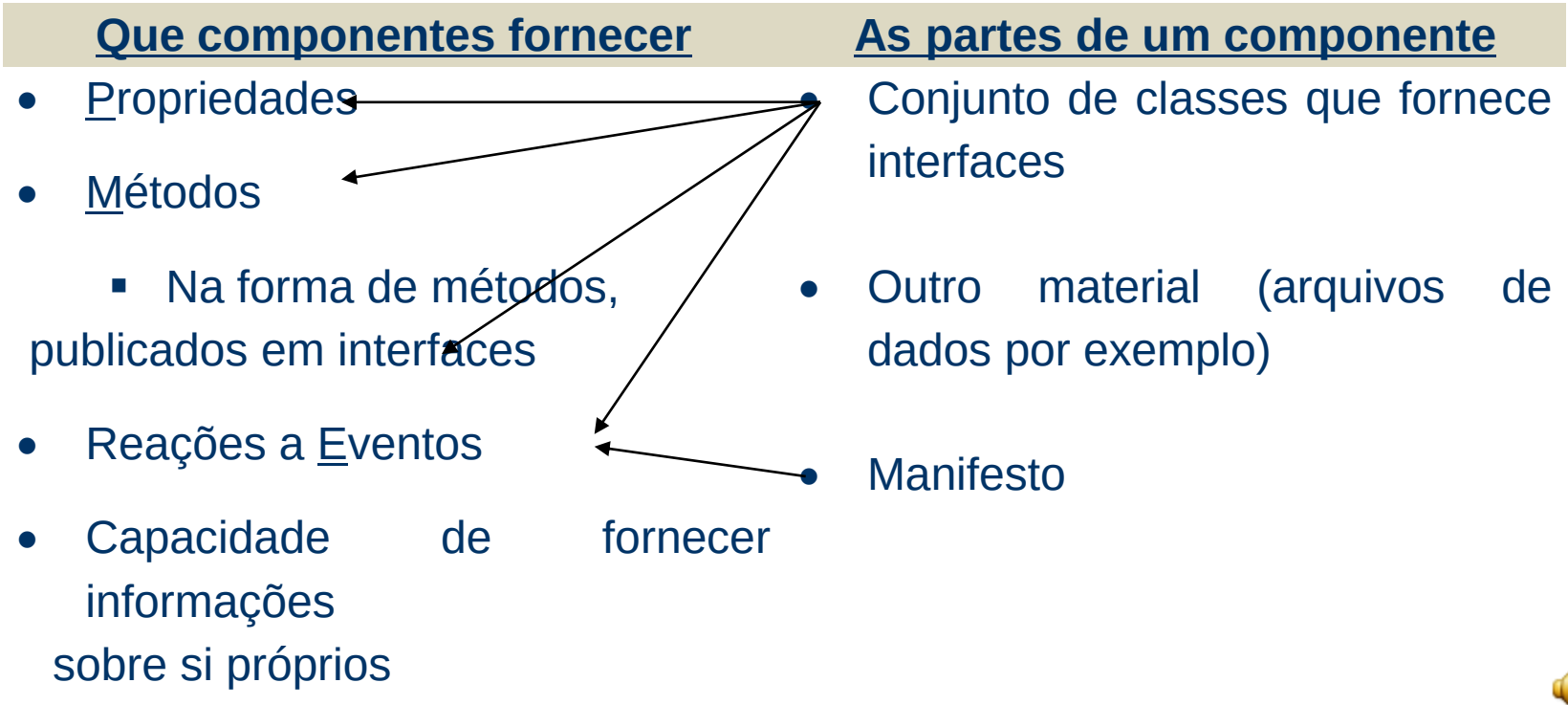
- Geralmente, um componente tem propriedades. Por exemplo, um componente gráfico talvez tenha uma propriedade backgroundColor.

Componentes

- Os componentes são criados para fornecer uma funcionalidade útil para utilização pela aplicação-cliente. Frequentemente, essa funcionalidade é dividida em uma ou mais interfaces. Por exemplo, um componente de planilha eletrônica poderia fornecer uma interface como ManipulaçãoDeCélula, que contém um método como avaliarCélula (int numDaColuna, int numDaLinha). Além disso, um componente talvez seja capaz de reagir aos eventos que acontecem nele, por exemplo, um componente botão reage à ação de um clique. Por fim, deve ser possível aos componentes responderem a perguntas sobre eles mesmos.

Componentes

O gráfico abaixo resume discussão.



Componentes

1.2 – Manifesto

- Uma vez que os componentes possam consistir em várias partes, em geral eles são empacotados com uma lista do seu conteúdo. Essa lista é chamada de manifesto. Esse não é um novo termo: contêineres de navio, por exemplo, têm um manifesto anexado.

Componentes

- Os manifestos de componente contêm as seguintes informações:
  - Identificação do componente;
  - Autoria do componente;
  - Lista de arquivos ou classes que compõem esse componente;
  - Outros componentes com os quais esse conta;
  - Informações de criptografia;
  - Meios de verificar se todas as partes do componente estão presentes;
  - Número da versão.

Componentes

1.3 – Introspecção e metadados

- Normalmente, os componentes são projetados para existir em um ambiente. O ambiente é projetado para diminuir a carga de trabalho do desenvolvedor no projeto e na codificação das operações de rotina. Ele também facilita o desenvolvimento visual. Os ambientes são projetados para assumir o maior número possível de responsabilidades comuns. Não há nada de novo sobre um software requerer um ambiente de operação: mesmo programas convencionais requerem ambientes (um sistema operacional, talvez uma Java Virtual Machine etc).

Componentes

- Para que os ambientes de componentes sejam úteis, eles precisam "conhecer" as entidades que os utilizam. Por exemplo, deve ser possível ao ambiente descobrir detalhes relacionados com as classes que compõem cada componente. Essas informações são chamadas de metadados, e o processo para obtê-los é chamado de introspecção, em particular, fornece as informações em tempo de execução sobre classes, como resumido na tabela a seguir.

Componentes

Classe	Nome, superclasse, superinterfaces, classes internas, campos, construtores, metadados.
Campo	Nome, tipo
Construtor	Parâmetros,
Métodos	Nome, parâmetros, tipo de retorno, exceções

Componentes

2 – Aspectos técnicos do desenvolvimento baseado em componentes

- Com o advento do uso de componentes houve algumas mudanças nos aspectos no desenvolvimento de software. Abaixo segue alguns exemplos dos aspectos que foram alterados.
- **Um novo processo de desenvolvimento:** O uso de componentes trouxe mudanças no processo de desenvolvimento porque além de desenvolver um produto, também os desenvolvedores tiveram que pensar na criação de abstrações reutilizáveis. Isso implicou em uma nova forma de pensar dos desenvolvedores.

Componentes

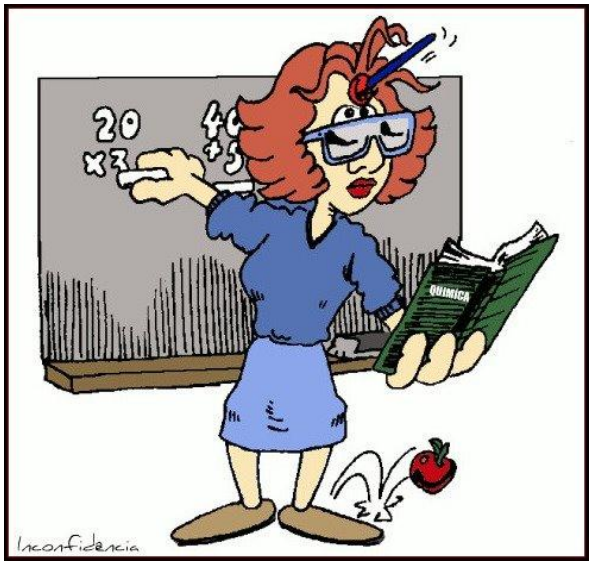
- **Novas etapas surgem:** A seleção de componentes e possíveis adaptações a eles surgiram. Os componentes selecionáveis podem ter sido construídos pelos próprios desenvolvedores ou por terceirizados. E ainda podem ter sido adquiridos através de compras.
- O tempo de montagem para construção dos componentes agora é feito em etapas diferentes. Antes, tudo era feito na mesma etapa, agora a composição é feita em uma etapa separada da construção dos componentes.

Componentes

- A fase de integração era a última fase antes dos componentes agora ela passou a ser a fase principal da construção das aplicações, mas passou a ser chamada de fase de composição.
- **Novos papeis no desenvolvimento:** Surgem os seguintes papéis:
  - Quem coordena os esforços de reutilização entre times e decide que componentes são desenvolvidos/terceirizados/comprados
  - Quem faz os componentes
  - Quem monta aplicações com componentes
  - Quem instala e configura componentes em ambientes finais.

Componentes

Não durma no ponto.



Exercício de fixação.