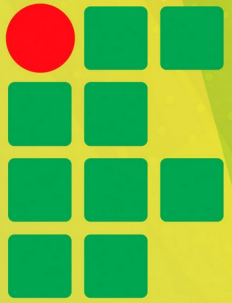
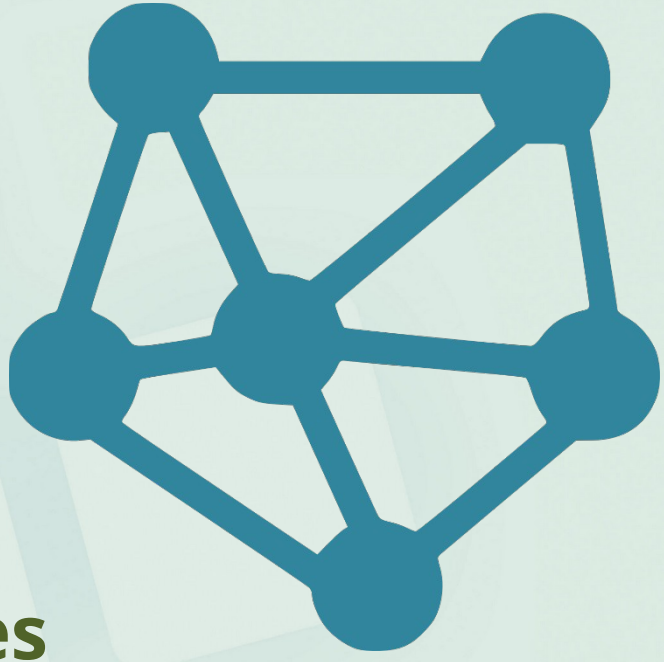


INSTITUTO FEDERAL

Sertão Pernambucano

Sistemas Distribuídos



Desenvolvimento Baseado em Componentes

Prof. Heraldo Gonçalves Lima Junior

INSTITUTO FEDERAL
Sertão Pernambucano

1. Introdução

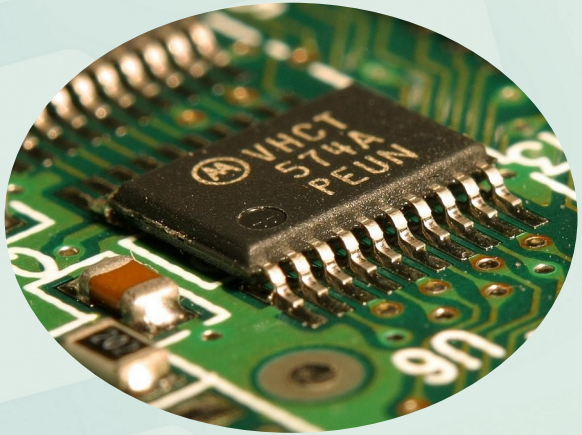
- O Desenvolvimento Baseado em Componentes (DBC) surge como uma nova **perspectiva de desenvolvimento de software caracterizada pela composição de partes já existentes**.

2. Motivação

- Vamos examinar mercados bem sucedidos para ver o que eles fazem diferente do que é feito no mercado de software.
- **Como mercados maduros dão boa relação qualidade/preço?**

2. Motivação

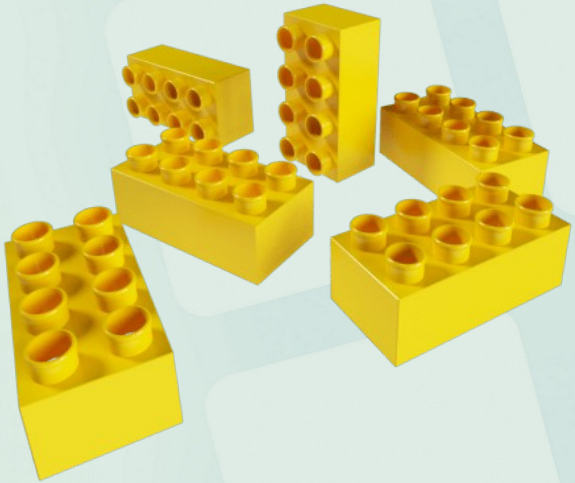
- Porque a manufatura usa Componentes Pré-Fabricados?
- **Pense em carros, bicicletas, hardware (com circuitos integrados)**



2. Motivação

- **Por que usar componentes é bom?**
  - A empresa não tem que fazer tudo;
  - Ela se concentra no que ela é boa;
  - Ela compra componentes de outros especialistas;
  - É a boa e velha "terceirização“;
  - Desta forma, pode-se focar melhor apenas no que falta.

2. Motivação



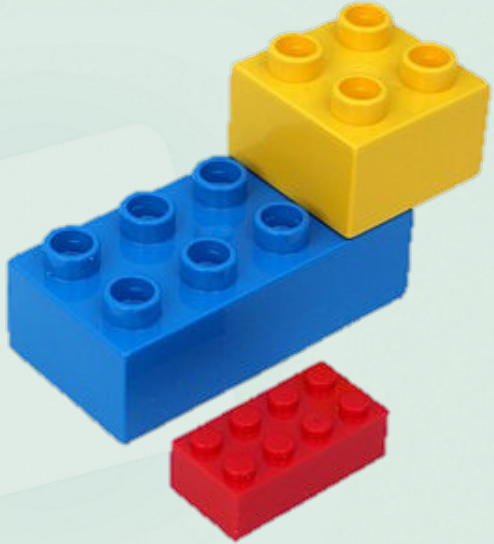
- **Resultados**
  - Componentes **reutilizáveis**;
  - Compre, não construa, porque deve ser mais barato;
  - Pare de escrever aplicações do zero cada vez que inicia um projeto;
  - Muda a ênfase da programação (construção) para a composição (montagem);

2. Motivação

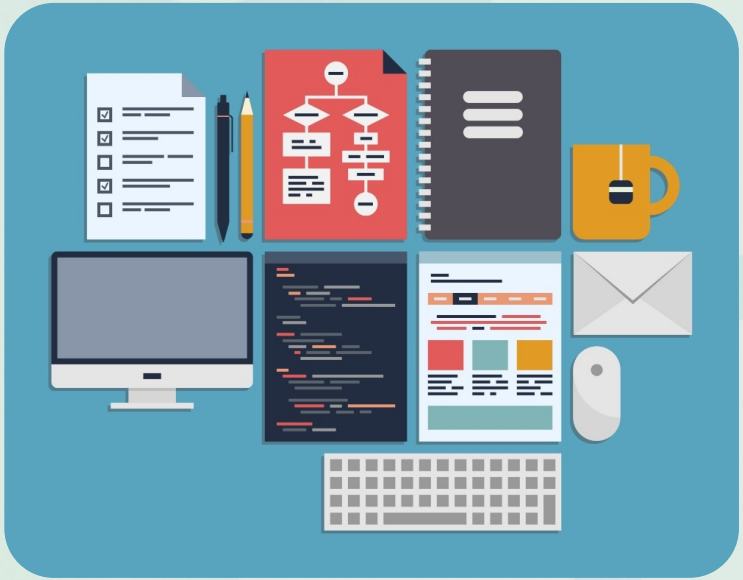
- **Precisamos de uma forma simples de dinamicamente conectar os componentes entre si mas não em tempo de compilação ou link-edição**
  - Quero usar uma ferramenta visual para
    - Configurar os componentes
    - Interconectar os componentes (estabelecer associações)
- **Estou "programando" sem codificar**: Attribute programming, Visual programming, Interactive programming, Rapid Application Development (RAD).

3. O que são componentes?

- Muitas definições foram oferecidas ao longo dos últimos anos.
- Vamos iniciar com a definição mais geral de D'Souza
  - Componente geral: "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."



3. O que são componentes?



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

3. O que são componentes?

- Mas nós queremos falar apenas de componentes de implementação.
- Definição de D'Souza, mais uma vez:
  - Componente de implementação: "Um **pacote coerente de implementação de software** que (a) pode ser **desenvolvido independentemente e entregue como unidade**; (b) **tem interfaces explícitas e bem definidas** para os serviços que oferece; (c) tem interfaces explícitas e bem definidas para os serviços que requer; e (d) **pode ser composto com outros componentes**, talvez após a customização de algumas propriedades mas sem modificar os componentes em si."

3. O que são componentes?

- Examinando esta definição (e as outras), podemos ver o que componentes têm de diferente comparados a bibliotecas ou outros artefatos de software
- São 3 características básicas:
  - Construção de aplicações por montagem;
  - Um componente explicita suas interfaces;
  - Um componente é uma unidade de empacotamento (packaging), entrega (delivery), implantação (deployment), e carga (loading).

3.1. Construção de aplicações por montagem

- **Essa é a diferença principal!**
- Um componente deve permitir que todo o trabalho (ou quase) seja feito pela composição de pedaços existentes.
- Isso significa que, no processo de desenvolvimento, novas etapas podem surgir.
  - Design time (ou assembly time) para juntar componentes e montar aplicações

3.1. Construção de aplicações por montagem

- **Composição de Terceiros:**
  - **Um "terceiro" é alguém que não tem acesso a detalhes de construção e não possui o código fonte**. Portanto, componentes podem ser modificados ao incluí-los na aplicação, mas sem ter código fonte
  - A modificação pode ser de atributos, por exemplo ("Attribute programming")
  - A modificação (configuração) do componente usa frequentemente uma ferramenta visual ("Visual Programming")

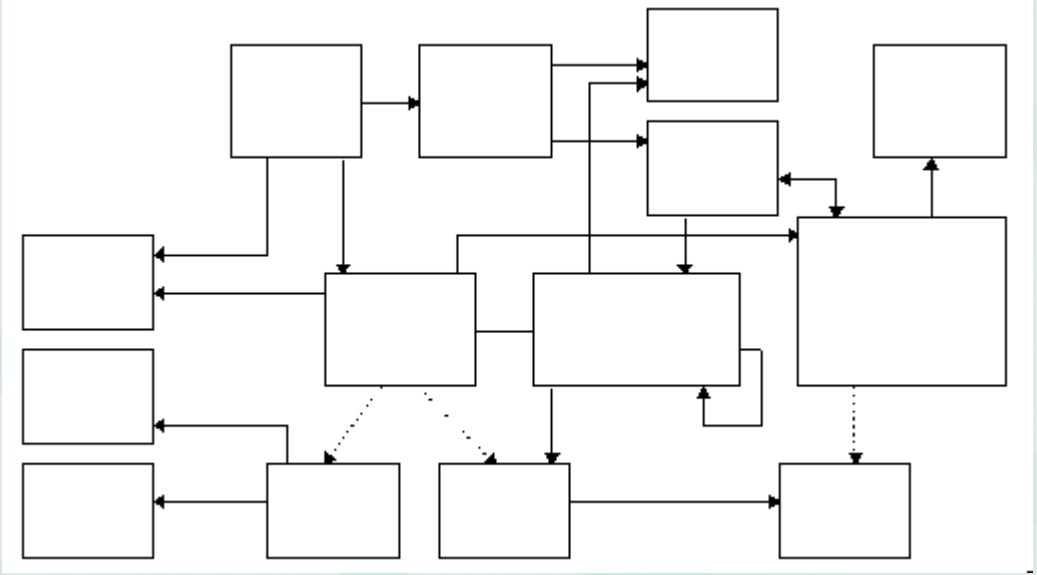
3.2. Um componente explicita suas interfaces

- Para ser plugável em vários contextos ("Plug-Replaceable"), temos que usar **interfaces padronizadas**
- Uma interface é um **contrato que especifica a funcionalidade** do componente
- Um componente tem duas interfaces:
  - Dos serviços que ele oferece (**export interface**)
  - Dos serviços que ele requer de outros componentes (**import interface**)

3.2. Um componente explicita suas interfaces

- As linguagens de programação têm se concentrado apenas em especificar as interfaces de serviços oferecidos e não de serviços usados pelos componentes.
- Se juntarmos um pequeno grupo de objetos funcionando conjuntamente e isolá-los com uma interface de componente, a complexidade (o acoplamento) vai diminuir, **reduzindo a chance de Objetos Hyperspaghetti**.

3.2. Um componente explicita suas interfaces



3.2. Um componente explicita suas interfaces

- Para permitir que o ambiente (frequentemente visual) de composição funcione, **as interfaces devem ser auto descritivas**:
  - **Deve ser possível descobri-las em tempo de execução**, sem ter tido conhecimento algum do componente antes;

3.3. Um componente é uma unidade de empacotamento, entrega, implantação, e carga

- **Unidade de empacotamento:** Inclui tudo dentro dele: a especificação de suas interfaces, legível em tempo de execução, implementação, imagens, outros recursos, ...

- **Unidade de entrega:** Um componente não é entregue parcialmente quando vendido. Ele é entregue num formato que é independente de outros componentes com os quais ele venha a ser composto

3.3. Um componente é uma unidade de empacotamento, entrega, implantação, e carga

- **Unidade de implantação:**
  - Um componente não é implantado parcialmente
  - Durante a implantação, seus atributos podem ser alterados (configurados)
  - Também significa que é a unidade de troca na manutenção (a aplicação inteira não precisa ser trocada)

3.3. Um componente é uma unidade de empacotamento, entrega, implantação, e carga

- **Unidade de carga:** Na aplicação final um componente é carregado por inteiro: "Quero um desses!"
  - Não posso dizer: "Me dê esse componente, mas só carne magra!"

4. Modelo e Framework de Componentes

- Existe em DBC um relativo consenso quanto à impossibilidade de separar componentes e arquitetura de software. **Um componente não pode ser visto de forma completamente independente dos outros componentes com os quais se relaciona e de seu ambiente**.
- Desta forma, a arquitetura de software assume um importante por ser a partir dela que é possível **especificar de forma mais detalhada como se dá a interconexão entre os componentes**.

4. Modelo e Framework de Componentes

- Um **modelo de componente** representa um elemento da arquitetura do sistema na qual **são definidos os padrões e convenções impostas aos componentes do sistema**, de modo a descrever a função de cada um e como eles interagem entre si. Com isso, busca-se expressar restrições de projeto arquitetural ou global.

4. Modelo e Framework de Componentes

- Elementos definidos pelo Modelo de Componente.

Item	Descrição
Interfaces	Especificação do comportamento e propriedades
Nomeação	Nomes globais únicos para as interfaces e componentes
Metadados	Informações sobre os componentes, interfaces e seu relacionamento
Interoperabilidade	Comunicação e troca de dados entre componentes de diferentes origens, implementados em diferentes linguagens
Customização	Interfaces que possibilitam a customização dos componentes
Composição	Interfaces e regras para combinar componentes no desenvolvimento de aplicações e para substituir e adicionar componentes as aplicações já existentes
Suporte a evolução	Regras e serviços para substituir componentes ou interfaces por versões mais novas
Empacotamento e utilização	Empacotar implementações e recursos necessários para instalar e configurar componentes

4. Modelo e Framework de Componentes

- Já **Framework de Componente**, é a implementação de serviços que dão suporte ou reforçam o modelo de componentes. A função do framework é gerenciar os recursos compartilhados pelos componentes e prover um mecanismo que possibilite a comunicação (interação) entre eles.

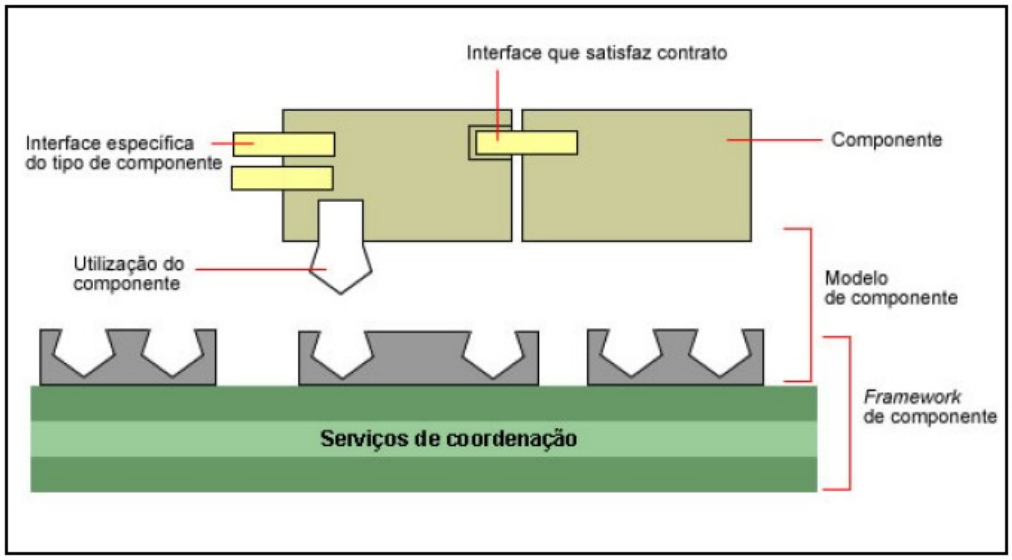
4. Modelo e Framework de Componentes

- Categorias de serviços presentes na infraestrutura de componentes.

Item	Descrição
Empacotamento	- definição de uma forma padrão que possibilite a infra-estrutura de componentes saber quais serviços o componente disponibiliza e a assinatura dos métodos que invocam estes serviços; - definição de uma forma padrão para solicitações de serviços a componentes externos;
Distribuição	- ativação e desativação das instâncias dos componentes; - gerenciar a alocação das instâncias para processos remotos; - prover uma transparência de localização, em que o cliente não precise saber onde está a instância do componente que fornece o serviço, bem como a
	instância do componente não precise ter conhecimento da origem das solicitações que recebe;
Segurança	- serviços de controle de acesso, além de serviços que proporcionem conexões seguras quando da transmissão de informações; - níveis de isolamento para garantir segurança e conexões confiáveis entre os componentes;
Gerenciamento de transação	- prover o gerenciamento de transações distribuídas, de modo a controlar e coordenar as interações complexas com os componentes, visando garantir a consistência dos dados;
Comunicação assíncrona	- suporte a comunicação assíncrona entre componentes, que normalmente ocorre através de alguma forma de enfileiramento das solicitações

4. Modelo e Framework de Componentes

- Relacionamento entre componentes, modelo e framework de componentes.



4. Modelo e Framework de Componentes

- **De que consiste um componente, tipicamente?**
  - Várias classes (código binário)
  - Definições de interfaces, usando algum mecanismo utilizável em tempo de execução
  - Possivelmente objetos (factories para criar objetos através do componente)
  - "Recursos" (podem ser arquivos de dados contendo formulários, strings, parâmetros, imagens, etc.) que são usados para configurar o componente
    - Alguns desses recursos podem ser mudados em tempo de execução.

4. Componentes x Objetos

- As diferenças geram controvérsia
- **Instanciação:**
  - **A instanciação de um componente não gera outro componente mas um "objeto" criado a partir de um protótipo** (o componente)
  - Um componente frequentemente é visto como um factory de objetos e não um objeto em si

4. Componentes x Objetos

- Em termos de linguagem OO, **um componente pode conter vários objetos**
  - Componentes têm granularidade maior, frequentemente

- Objetos não são empacotados como componentes
  - Componentes são sujeitos à composição em tempo de design

4. Componentes x Objetos

- **Se o componente obedece às suas interfaces, ele pode ser implementado em qualquer linguagem** (mesmo sem ser OO!) e rodar em qualquer plataforma
  - Objetos são manipulados com linguagens OO apenas

5. Categorias de Componentes

- Podemos classificar componentes sob várias dimensões

- **Escopo**
  - Componentes de Especificação (diagramas, documentos)
  - Componentes de Implementação (classes, ...)

5. Categorias de Componentes

- **Objetivo**
  - Domínio (voltado ao problema)
  - Tecnologia (para o suporte, infra-estrutura)

- **Abstração**
  - Geral (aplicação em muitos domínios: horizontal)
  - Específico (aplicação em um único ou poucos domínios: vertical)

5. Categorias de Componentes

- **Granularidade**
  - Fina (um widget GUI)
  - Grossa (shopping-cart, agência bancária, fatura, contas a receber)
    - As oportunidades de reuso podem ser menores mas tais componentes são cruciais para melhorar a produtividade no desenvolvimento

5. Categorias de Componentes

- **Localização**
  - Cliente
  - Servidor (Middle tier)

5. Categorias de Componentes

- **Serviços providos:**
  - Gerência de configuração e de propriedades
  - Notificação de eventos
  - Acesso a metadados e reflexão
  - Persistência
  - Controle de transação e concorrência
  - Segurança
  - Licenciamento
  - Controle de versão
  - Auto testes
  - Auto instalação

Obrigado!
Vlw! Flw!

INSTITUTO FEDERAL
Sertão Pernambucano