Arquitetura de Sistemas

Aula 9: Provisionamento e Construção – Parte I

INTRODUÇÃO

A construção de componentes é o último processo da metodologia apresentada por esta disciplina e visa construir os componentes definidos pelo arquiteto de sistemas para a solução do problemas apresentados pelas partes interessadas.

Aqui nós veremos, dentro deste processo, o passo a passo para que seja feita uma construção limpa e dentro dos padrões, seguindo as boas práticas de arquitetura de sistemas.

Nesta aula, então, apresentaremos a maneira como os programadores devem lidar com a construção de componentes e a sua relação com os resultados a serem alcançados. Dessa forma, ficará evidente a importância de sua aplicação no contexto da arquitetura de sistemas.

OBJETIVOS

Reconhecer a importância do provisionamento e a construção de componentes para o melhor resultado na arquitetura de sistemas;

Analisar os elementos que compõe a etapa de construção de componentes como parte integrante da metodologia apresentada na disciplina;

Identificar a relação entre a construção de componentes e os outros processos da metodologia apresentada na disciplina.

COMPONENTES

Um dos conceitos mestres na arquitetura de sistemas baseada em componentes (glossário) é que um componente é definido e construído para fornecer um certo nível de serviço.

Como vimos nas aulas anteriores, teremos os componentes separados em função dos grupos de serviços que eles oferecem. Teremos componentes de hardware, de sistema operacional, de SGDB, de aplicativos comerciais e de sistemas da própria corporação já desenvolvidos.

O arquiteto de sistemas, baseado nos requisitos do novo sistema, vai executar o design da nova aplicação, identificando todos os componentes necessários e aplicando reuso aos componentes que já existirem. Somente serão construídos os componentes que não existirem.

É importante que o arquiteto de sistemas conduza a definição e a construção desses novos componentes, de maneira que eles passem a fazer parte do conjunto de componentes a serem reutilizados no futuro.

AMBIENTE DE COMPONENTES

Um ambiente componente é um meio ambiente de objetos distribuídos.

Os componentes devem estar em conformidade a um conjunto de regras padrão para que se possa operar nesse ambiente, e um conjunto de serviços de infraestrutura (suporte a transações, segurança, concorrência e assim por diante), na qual o componente de aplicação pode depender.

Atenção

,

Além disso, vamos nos limitar a esses ambientes de componentes que fornecem os serviços de infraestrutura declarativa, usando uma abordagem por framework (por vezes chamado de “programação baseada em atributo”), em vez de usarmos uma invocação explícita dentro da própria lógica da aplicação.
Fazemos isso pois os frameworks são os mais indicados para fornecer uma base mais sólida para a próxima geração de aplicativos baseados em componentes distribuídos, em escala empresarial.

Nesse contexto, temos as seguintes opções:

Ambiente de Componentes	Dependência de Plataforma	Dependência de Linguagem
Microsoft COM+	Windows	Nenhuma
Enterprise JavaBeans (EJB)	Nenhuma	Java

Nesta disciplina, não vamos fazer análise de qual ambiente é melhor e em que condições, tampouco vamos direcionar qual dos dois devem ou não ser usados. Aqui vamos nos limitar a relatar os conceitos de uso dos frameworks sem a preocupação de detalhar particularidades de cada um deles.

FRAMEWORK CCM – CORBA COMPONENT MODEL

O CCM é um framework de componentes do lado do servidor, cuja finalidade é facilitar o desenvolvimento e a instalação de aplicações distribuídas que utilizam a arquitetura de sistemas por componentes.

Esses componentes podem ser de diversos tipos, como vimos nas aulas anteriores.

Níveis de componentes do CCM

O CCM é dividido em dois níveis de componentes:

NÍVEL BÁSICO

Provê uma forma simplificada de distribuir um objeto CORBA como componente.

NÍVEL ESTENDIDO

Provê um conjunto maior de ações, como as portas de comunicação que representam os elementos de conexão entre os componentes.

Cinco tipos de modelos do CCM

O CCM é também estruturado em cinco tipos de modelos:

MODELO ABSTRATO

Define os atributos, portas de comunicação e home dos componentes.

MODELO DE PROGRAMAÇÃO

Composto pela CIDL (Component Implementation Definition Language) e pelo CIF (Component Implementation Framework).

MODELO DE EMPACOTAMENTO

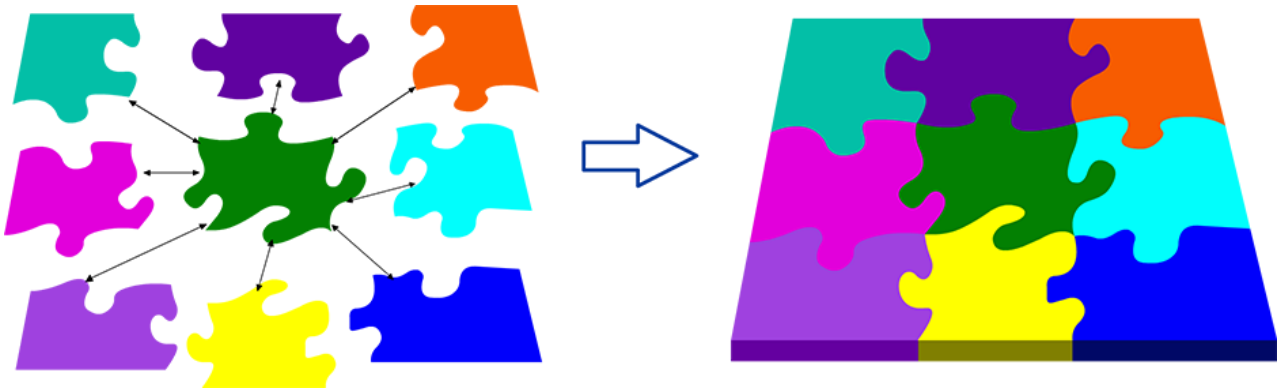
Especifica como os componentes e suas implementações devem ser empacotados.

MODELO DE INSTALAÇÃO

Define um mecanismo padrão para a instalação de aplicações.

MODELO DE EXECUÇÃO

Define o ambiente de execução para as instâncias do componente.



ARQUIVOS CIDL

Os seguintes elementos são especificados em um arquivo CIDL, no contexto do desenvolvimento na arquitetura de sistemas por componentes:

Categoria	CORBA USAGE MODEL	TIPO DE API DO CONTAINER	INTERFACE BASE CHAVEADA	EXEMPLO
Service	Stateless	Session	Não	Componentes do Sistema
Session	Conversational	Session	Não	Interfaces de Interação
Process	Durável	Entity	Não	Regras de Negócio
Entity	Durável	Entity	Sim	Objetos de Negócio

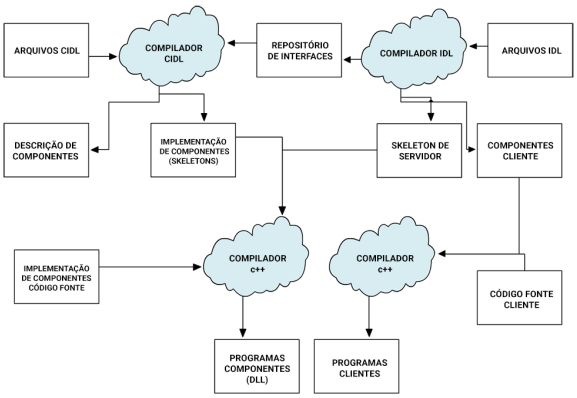
ARQUIVOS CIF

O CCM define um grande número para suportar a estrutura e funcionalidade dos componentes.

As implementações de muitas dessas interfaces podem ser geradas automaticamente. É nesse contexto que apresenta-se o CORBA - Component Implementation Framework (CIF).

CCM define uma linguagem declarativa, Component Implementation Definition Language (CIDL), para descrever implementações e persistência de estado de componentes e home.

CIF usa a CIDL para gerar skeletons que automatizam tarefas básicas, como navegação, ativação e gerenciamento de estado.



CONTAINERS

Fonte da Imagem:

Os componentes são empacotados em DLLs e executados em servidores de componentes.

As implementações dos componentes dependem dos conceitos da programação orientada a aspectos para encaminhar requisições de clientes para os elementos de servidor.

Os componentes não precisam saber como tratar problemas, como a criação de hierarquia de POAs, e localizar serviços do CCM.

Funcionalidades:

Para isso, foram definidos os containers, com as seguintes funcionalidades:

- Ativação/desativação de implementações de componentes, preservando recursos (como memória).
- Fornecimento de camada de adaptação com os serviços de transação, persistência, segurança e notificação.
- Fornecimento de camada de adaptação para call-backs.
- Gerenciamento de políticas do POA.

Gerenciamento do ciclo de vida

O gerenciamento do ciclo de vida dos componentes de servidor é feito através de políticas que controlam o momento de ativação/desativação dos componentes:

Method

Ativação/desativação a cada chamada de método, limitando o uso de memória ao tempo de duração da operação, mas acrescentando o custo de ativação e desativação do componente.

Transaction

Ativação/desativação a cada transação. Memória permanece alocada durante a transação.

Component

O container ativa o componente, quando for feita a primeira chamada a alguma de suas operações, e desativa, quando explicitamente requisitado pela aplicação, deslocando a memória utilizada pelo componente.

Container

O componente será ativado quando for feita a primeira chamada a alguma de suas operações e, ao final da execução da mesma, será desativado. Entretanto, a memória permanecerá alocada até que o container decida deslocá-la.

EMPACOTAMENTOS E DISTRIBUIÇÃO

Fonte da Imagem:

Em sistemas distribuídos, componentes podem ser implantados em diversos servidores e sistemas operacionais.

Além disso, um componente pode depender de outros componentes, tornando o processo de empacotamento e distribuição bem mais complicado do que se imagina.

CCM descreve componentes e suas dependências usando Open Software Description (OSD), que é um XML Document Type Definition (DTD) definido pelo consórcio www.

Componentes são empacotados em DLLs. Package descriptors são documentos XML em conformidade com o OSD e DTD, descrevendo o conteúdo da DLL e suas dependências.

CCM e OSD também definem component assembly descriptors, que descrevem instruções de implantação e topologia dos componentes, e têm como objetivo o suporte à implantação automática dos componentes.

Exercício

1. Na construção dos componentes de um domínio do problema, há três passos principais. Assinale a alternativa que corresponde à ordem correta de criação desses passos:

☐
A) Instalar componentes, gerar código dos componentes e modelar componentes.
☐
B) Gerar código dos componentes, modelar componentes e instalar componentes.
☐
C) Modelar componentes, gerar código dos componentes e instalar componentes.
☐
D) Modelar componentes, instalar componentes e gerar código dos componentes.

○
○
○
○

Justificativa

---

2. A etapa “modelar componentes” é dividida em três passos: definir domínio do problema, especificar componentes e projetar componentes. Assinale a alternativa que não corresponde à especificação de cada um dos passos dessa etapa:

☐
A) No passo “definir domínio do problema”, obtêm-se as especificações UML do domínio do problema a partir estudo do conhecimento do domínio do problema.
☐
B) No passo “especificar componentes”, obtêm-se os componentes especificados usando técnicas de DBC.
☐
C) No passo “projetar componentes”, faz-se o projeto interno dos componentes, usando técnicas de DBC do método Catalysis [D Souza, 1999].
☐
D) No passo “especificar componentes”, as atividades são realizadas no nível de abstração do projeto interno de componentes.

○
○
○

Justificativa

Glossário

COMPONENTES

Que compõe, ou entra na composição de alguma coisa; que contribui para formar; constituinte.