Desenvolvimento baseado em componentes

*Nota: Pesquisa realizada para a matéria de Engenharia de Software no curso de S.I. da UFV-CRP sob a orientação do professor Leandro Furtado.

1st Gabriel Oliveira
Matricula: 5203
Estudante UFV-CRP
Rio Paranaiba-MG, Brasil
gabriel.matheus@ufv.br

***Abstract*—In this paper, we will bring some aspects of the development of software based on components. We will discuss what are these components, some models, differences between them and objects, we will also discuss the technology used and how to apply them. We will also explain what are their advantages over the normal development of a system**

***Index Terms*—componentes, objetos vs componentes, desenvolvimento de software, vantagens componetização, JavaBeans, .NET, CCM**

I. Introdução

Nesse trabalho, irei trazer alguns aspectos de desenvolvimentos de um sistema computacional baseado em componentes. Explicarei o que são esses componentes, alguns modelos, diferença entre eles e objetos e abordaremos tambem sobre a tecnologia que se usa e como aplica-los.

II. Conceito

Componente é aquilo em que algo se faz parte da composição de um todo. Trata-se de elementos que, atraves de algum tipo de associação ou relação, dão lugar a um conjunto sem desilgualdades.

Como por exemplo, podemos trazer a seguinte situação, em um todo, por exemplo um console playstation, podemos encontrar diversos componentes, como leitor de mídia, memória de sua placa, manetes, cooler de ventilação e outros.

Pode-se dizer então, que o desenvolvimento baseado em componentes permite que esse sistema final, no caso o console, seja tratado como vários sistemas em menores escalas, necessarios para o funcionamento claro, diminuindo sua complexidade e permitindo que cada componente tenha foco em apenas uma funcionalidade ou um conjunto de funcionalidades semelhantes. No caso do desenvolvimento de software, adota-se a mesma linha de raciocínio para a criação do mesmo.

III. Características de componentes

Para o desenvolvimento de um software acontecer, é relevante citar algumas das características que os componentes devem apresentar, visto que as empresas de software hoje em dia procuram meios de desenvolvimento para alcançar uma alta produtividade com alto retorno sobre o investimento. Com isso, a margem para erros deve ser pequena e o produto entregue deve atender grande parte das solicitações do cliente.

Esta busca pela alta produtividade passa por muitos itens. Dentre eles:

- a contratação de profissionais experientes ou realização de treinamentos
- utilizar ferramentas mais modernas
- adequar os processos e normas de desenvolvimento

Com base nisso, uma estratégia de melhoria de produtividade é a reutilização de artefatos já produzidos em outras ocasiões. Logo, essa reutilização se dá pela construção do produto através do re-uso de seus componentes. Pode-se dizer que os componentes tem de:

- Permitir o desenvolvimento de sistemas por montagem: um componente deve permitir que parte do (ou todo) desenvolvimento do sistema seja feito pela composição de partes existentes.
- Possuir interfaces explícitas: o funcionamento do componente não pode depender do contexto onde estará presente, a não ser através de interfaces, sejam de exportação (dos serviços requeridos) ou de importação (dos serviços requeridos). Essa característica possibilita a diminuição do acoplamento do sistema.
- Ser uma unidade autocontida: isso significa que o componente deve funcionar independentemente de outro componente.

IV. Desenvolvimento de software baseados em componentes

Em primeiro lugar, deve-se realçar que a utilização de componentes para a fabricação de um software requer uma nova forma de pensar da equipe do projeto. Arquitetos, analistas e programadores devem se preocupar à partir daí com a integração do sistema e com os componentes que serão utilizados, necessitando assim de um atentamento maior na definição de interfaces internas e externas.

Outro aspecto, é que esses pontos podem gerar mudanças no processo de desenvolvimento utilizado. Essas mudanças são necessárias porque novas etapas surgem ao desenvolver um sistema utilizando componentes.

Essas novas etapas são basicamente três:

- Seleção: pesquisar e selecionar os possíveis componentes que poderão ser utilizados no desenvolvimento do sistema. Nessa etapa, deve se preocupar em selecionar os componentes que atendam, total ou parcialmente, os requisitos da arquitetura do sistema. Além disso, deve-se garantir que esses componentes permitem adaptações.

- Adaptação: após selecionado, o componente deve ser adaptado às especificações do sistema. Essa etapa é necessária pelo fato de que o componente se encaixar perfeitamente no sistema em desenvolvimento é algo bem raro.
- Composição: é o momento no qual o componente se conecta ao sistema. Logo é a etapa principal do desenvolvimento de sistema por componentes, pois gera a interface de comunicação entre eles e o sistema propriamente dito.

V. Tecnologia de componentes

O desenvolvimento de software baseado em componentes, como também o desenvolvimento dos próprios componentes, são viáveis através do uso de diferentes tecnologias. Estas por sua vez surgiram devido à dificuldade de alcançar objetivos primários no desenvolvimento, juntamente com a instalação e montagem independentes dos componentes, caso os mesmos em um sistema sejam desenvolvidos de forma isolada em relação a outros, devido a conflitos de interfaces, formas de combinações, entre outros. Com o intuito de evitar complicações, as tecnologias seguem certos padrões, como a construção de interfaces fornecidas ou requisitadas pelos componentes.

Entre diversas tecnologias industriais, destacam-se os modelos Enterprise JavaBeans da Sun Microsystems, o COM+ e o .NET da Microsoft e o CCM (Corba Component Model) da OMG (Object Managment Group). Os modelos em questão destacam-se por abordar problemas práticos e descrevem-se em termos técnicos. Existem vários detalhes de implementação nesses modelos, porém os usuários têm dificuldades para entender conceitos e princípios das aplicações. Vale ressaltar que estas tecnologias atuam nas fases finais do desenvolvimento de software baseado em componentes, abrangendo implementação, montagem e execução.

A. Enterprise JavaBeans

A plataforma J2EE (Java 2 Plataform, Enterprise Edition) aborda componentes para o projeto, desenvolvimento, composição e utilização de aplicações corporativas, como também um conjunto de APIś que está definida através de extensões javax.ejb.

Porém, EJB não se caracteriza apenas como uma API, mas também como uma arquitetura de componentes para desenvolvimento e aplicações corporativas, com base em componentes distribuídos. Levando em conta uma arquitetura cliente/servidor, os componentes EJB são posicionados ao lado do servidor e acessados por diferentes aplicações pelo cliente. Logo, os componentes são desenvolvidos para executarem tarefas típicas ao lado do servidor, como algoritmos de diferentes complexidades ou controlarem e executarem um grande volume de transações.

Com o uso do EJB, torna-se possível desenvolver aplicações distribuídas escaláveis e protegidas, sem a preocupação de implementar todos os procedimentos pertinentes para prover distribuição. EJB é definido como uma arquitetura ou um modelo de componentes. Dito isto, é compreensível que a ferramenta aborda a especificação de componentes flexíveis, fornecendo um conjunto de convenções, definições e regras de modo a implementar os componentes ao lado do servidor, utilizando a linguagem Java.

Em relação a compatibilidade dos componentes EJB, os mesmos possuem um relacionamento entre a infraestrutura e os componentes, como também entre os próprios componentes, sendo este realizado em termos de contrato que devem ser obedecidos, garantindo assim que as aplicações possam fazer uso de EJBś de diferentes origens, pois os mesmos devem obrigatoriamente estar em conformidade com as definições estabelecidas.

O compreendimento de EJB sustenta-se nas três principais entidades da arquitetura:

- Enterprise Bean - nome dado aos componentes EJB, que se denominam como componentes distribuídos e hospedados em containers.
- Container EJB - é o espaço onde os beans são armazenados e disponibilizados para posterior uso, tendo como função gerenciar os modos do bean ao longo de sua execução, por exemplo, acesso remoto, segurana, transações, concorrência e acesso a recursos. Embora a existência do container seja transparente para o bean, o mesmo não pode trabalhar fora dele. Faz parte da função de um container intermediar a comunicação entre cliente e bean, como também entre bean e servidor EJB.
- Servidor EJB - mesmo não existindo uma distinção marcante entre container e servidor EJB, o servidor tem como objetivo apresentar diversos containers, sendo um para cada tipo de bean, também prover serviços que favoreçam o gerenciamento dos beans por parte dos containers. Muitas implementações têm desenvolvido containers e servidores como o mesmo produto. Na atualidade, há um leque de produtos comercias de domínio público para servidores/containers EJB, como por exemplo o WebSphere Application Server da IBM e o BEA WebLogic Server da BEA Systems. Já nas opções de domínio público exitem jBoss da Tekel, JOnAS da Evidian e OpenEJB da Intalio.

A figura a seguir mostra a relação entre bean, container EJB e servidor EJB. Mesmo que a figura ilustre apenas um de cada, é possível um container abranger mais de um bean, como também um servidor pode disponibilizar vários containers.

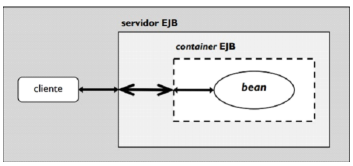

Figura 1. Relação entre eles

B. CCM

Este modelo de componente foi implementado baseado na experiência de uso do Corba, JavaBean e do EJB. Integrar várias características utilizando engenharia de software é

uma das principais vantagens do modelo CCM, entretanto, sua especificação é complexa, levando a descrição de uma aplicação de software baseada em diferentes formalismos. As fases de empacotamento e provisionamento não são abrangidas pela especificação CCM, mesmo essa sendo bastante extensa. As seguintes portas formam a interface do componente:

- Facetas: interfaces fornecidas pelo componente para relacionamento com o cliente.
- Receptáculos: locais de conexão responsáveis por descrever a habilidade de usar uma referência fornecida por algum agente externo.
- Fontes de eventos: locais de conexão que emitem eventos de determinados tipos.
- Receptores de eventos: locais de conexão que podem receber um evento.
- Atributos: valores nomeados e expostos por operações.

O conceito de conexão é definido por CCM como uma referência de objeto. Os componentes são interligados através da conexão entre facetas e receptáculos e entre fontes e receptores de eventos. Um conjunto de segmentos de códigos executáveis providos em qualquer linguagem de programação definem a implementação de um componente, visto que este conjunto implemente ao menos uma porta. Um container é utilizado pelos componentes Corba para implementar o acesso de componentes diversos aos serviços do sistema.

C. .NET

Diversas tecnologias para implementação de componentes são apresentadas pela Microsoft, como o COM, específico para linguagem C/C++, contando também com convenções de inter-operabilidade binária e interfaces. A Microsoft também fornece o DCOM, que estende o COM usando distribuição, e o MTS, responsável por estender o DCOM através de serviços de persistência e transação. A união de todos estes forma o modelo COM+.

Como modelo de componente mais recente, a Microsoft elaborou também o .NET. Ao contrário dos demais, ele não é baseado no COM, devido a limitação da inter-operabilidade binária existente no mesmo. O .NET trabalha com inter-operabilidade de linguagem e introspecção. Como causa disso, é definida uma linguagem interna para o modelo, chamada de MSIL (Microsoft Intermediate Language), muito similar ao Java Byte Code.

A abordagem de linguagem de programação representa a programação de componentes no .NET. Isso demonstra que a aplicação contém as informações associadas dos relacionamentos com outros componentes e o compilador tem a função de gerar essas informações em tempo de execução. O modelo conta com um elo dinâmico específico, responsável por estabelecer a conexão entre recursos fornecidos e requeridos. O componente no .NET consiste também em módulos, que são arquivos tradicionais executáveis ou DLLs (Dinamic Link Libraries). Ao compilar o módulo principal, a linha de comando fornece a lista de módulos que compõe uma montagem.

VI. Framework de componentes

O framework de componentes diz respeito à base na qual estes padrões e modelos de componentes são utilizados. Dito isso, framework e modelo de componentes são considerados conceitos complementares e fortemente relacionados. O framework deve suportar as definições estabelecidas pelos modelos de componentes, como também deve regular e respeitar as mesmas. Outra vantagem do uso de frameworks é a despreocupação dos desenvolvedores em torno de implementar em suas aplicações inúmeros serviços complexos, tais como troca de mensagens, passagem de dados e ligação entre componentes.

Framework de componentes define-se como uma infraestrutura, sendo responsável também por ao menos uma destas categorias de serviços: distribuição, empacotamento, gerenciamento de transações, segurança e comunicação assíncrona. A seguir demonstra-se uma breve definição dos serviços de estruturas para 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

Figura 2. Tabela das definições

A relação entre modelo, componentes e framework pode ser identificado na figura a seguir:

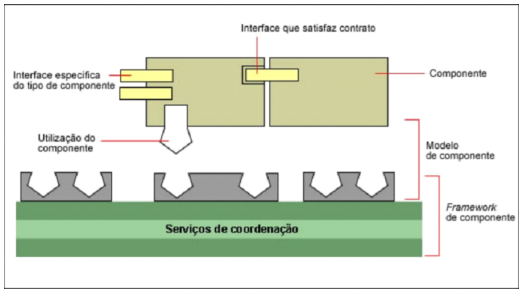

Figura 3. Tabela das definições

Com esta representação é possível identificar o framework como uma infraestrutura de suporte a comunicação e ligação dos componentes, via fornecimento de serviços de

coordenação. As aplicações definidas através de frameworks não reutilizam apenas o código fonte, como também o projeto, que é considerada a característica mais relevante dos frameworks. Há uma independência significativa entre framework e a forma como os componentes são implementados. Ele apenas requer que os componentes sejam compatíveis com conexões originais, como também possam ser substituídos por outros mais específicos.

Além destas definições, frameworks podem também ser utilizados como componentes, devido ao fato dele apenas cobrir uma parte do domínio da aplicação e se puderem ser arbitrariamente combinados, tornando-os componentes concretos e sendo possível o desenvolvimento de uma variedade de produtos através da seleção e combinação desses componentes.

Quanto a tipos de componentes, formas de interação e recursos necessários, são estas definições que identificam os modelos de componentes. Os componentes estipulados para executar determinadas funcionalidades devem estar de acordo com as definições do modelo de componentes, pertencendo a um dos possíveis tipos e respeitando as maneiras de interação pré-definidas, além de usufruírem dos serviços disponibilizados. Por fim, temos que a utilização de componentes caracteriza a relação de modelo e framework de componentes.

VII. Orientação a objetos e desenvolvimento baseado em componentes

É de conhecimento geral, que a orientação a objetos é um modo de analisar, modelar e programar sistemas de software que buscam resolver um problema, decompondo-o em partes menores, a fim de obter a solução completa. Se torna possível por meio da aplicação dos conceitos de abstração de dados e modularização.

Como podemos observar, os princípios do desenvolvimento baseado em componentes é bem parecido a orientação a objetos, entretanto, a orientação a objetos simplesmente, não apresenta todas as abstrações necessárias para o desenvolvimento baseado em componentes. Ela necessita de se especificar no contexto onde um componente pode ser considerado uma peça de reposição e também alguns mecanismos da orientação a objetos dificultam a reutilização futura dos componentes, como por exemplo, o fator da herança. Com isso, faz-se necessária a modificação do processo para enfatizar a reutilização dos modelos ja criados.

VIII. Vantagens do desenvolvimento baseado em componentes

Como já visto, na engenharia de software visamos o desenvolvimento de um software na busca de alcanar alguma finalidade, o desenvolvimento baseado em componentes se mostra como uma alternativa muito viável no processo de desenvolvimento, devido a que este método se centra na reutilização de determinadas classes de software. Classes de software, mecanismos tais como segurança, troca de mensagens e controle de versões são elementos comuns em qualquer desenvolvimento de produtos de software, facilmente encontrados e manipulados através do uso de componentes, o que acarreta numa vantagem significativa em relação às tecnologias de desenvolvimento tradicional.

O desenvolvimento através de componentes utiliza classes de software pré-existentes, extinguindo a necessidade dos desenvolvedores de implementar o software do zero, possibilitando o desenvolvedor selecionar as classes necessárias para o desenvolvimento.

O tempo de desenvolvimento reduz relevantemente, tendo por consequência a disponibilização mais rápida do software no mercado, adiantando o faturamento com o mesmo e aumentando a produtividade da empresa. Em termos de produtividade, o desenvolvimento baseado em componentes permite a empresa assumir mais projetos em relação ao desenvolvimento tradicional. Tendo como base um componente já testado e com a qualidade assegurada, o produto final também proverá de mais qualidade, excluindo as possíveis correções de erros que seriam necessárias no desenvolvimento tradicional. Além disso, o custo torna-se menor, pelo fato de não ser necessário o desenvolvimento de uma parte do software, podendo este valor ser investido no desenvolvimento de novos componentes, que podem ser utilizados posteriormente ou comercializados.

IX. Conclusão

A componetização portanto, é uma maneira eficaz e segura de desenvolvimento de softwares, pois as atividades são levadas a componentes já existentes, entretanto, necessita-se precisamente da atividade de adaptação, devido ao fato de que os componentes não são desenvolvidos para atender todos os seus usos. Assim, pode-se criar e reutilizar destes componentes, vale lembrar também que diversas tecnologias industriais podem fazer parte do processo de componetização e elas sempre atuam nas fases finais do desenvolvimento. Por fim, a relação do desenvolvimento por componentes com a orientação a objetos nos mostrou que o processo de desenvolvimento componetizado é mais efetivo, por não depender de fatores que travam a pesquisa de acordo com as etapas que já foram desenvolvidas.

Referências

[1] Paula Marques Donegan, ”Geração de famílias de produtos de software com arquitetura baseada em componentes”, dissertação de mestrado apresentado ao Instituto de Ciências Matemáticas e Computação, USP, São Carlos, 2008.

[2] Spagnoli, Luciana, and Karin Becker. ”Um estudo sobre o desenvolvimento baseado em componentes.”Relatório Técnico do Programa de Pós-Graduação em Ciência da Computação da PUCRS, Porto Alegre, Brasil (2003).

[3] Werner, Cláudia Maria Lima, and Regina Maria Maciel Braga. ”Desenvolvimento Baseado em componentes.”XIV Simpósio Brasileiro de Engenharia de Software. João Pessoa, Brasil (2000).

[4] OLSCHOWSKY, Caio Borges. ”DESENVOLVIMENTO BASEADO EM COMPONENTES.”

[5] Silva Junior, Moacir Caetano da. ”COSMOS: um modelo de estruturação de componentes para sistemas orientados a objetos.”(2003).

[6] Emmerich, Wolfgang, and Nima Kaveh. ”Component technologies: Java beans, COM, CORBA, RMI, EJB and the CORBA component model.”ACM SIGSOFT Software Engineering Notes. Vol. 26. No. 5. ACM, 2001.

[7] Gimenes, Itana Maria de Souza, and Elisa Hatsue Moriya Huzita. ”Desenvolvimento baseado em componentes: conceitos e técnicas.”Rio de Janeiro: Ciência Moderna (2005).