UNINOVE

Arquitetura em 3 Camadas

APRESENTAÇÃO DOS CONCEITOS BÁSICOS QUE ENVOLVEM O DESENVOLVIMENTO DE SOFTWARE BASEADO EM 3 CAMADAS, E SUA EVOLUÇÃO A PARTIR DO MVC.

AUTOR(A): PROF. PAULO RICARDO BATISTA MESQUITA

Há vários modelos para arquitetura de sistema e de software disponíveis, e muitas deles voltadas para problemas bastante específicos no desenvolvimento de software, e às vezes, mesmo que sejam aplicativos com finalidades bastantes específicas, como por exemplo, processamento de imagens digitais ou captura de imagens de exames de medicina.

De acordo com a UML, uma das formas mais interessantes para o desenvolvimento de aplicativos, é dividir o aplicativo em componentes de software. Entretanto, componente de software é um termo cujo significado varia bastante na literatura, e por quem os usa, e consequentemente, causando muita confusão às vezes. Para estabelecer um entendimento comum, nós definimos um componente de software como sendo a parte de uma aplicação que é responsável pela execução de determinada tarefa ou serviço, como por exemplo, verificação se o número de CPF informado num sistema de vendas on-line, é válido ou não.

De acordo com [1], um componente de software pode conter uma ou várias classes que executam a tarefa atribuída a ele. Mas para executar essa tarefa, ele precisa receber algumas informações para efetuar sua tarefa, e assim, poder informar um resultado para quem requisitou seu uso. Os componentes de software são usados para estruturar um aplicativo de software, como é o caso dos frameworks, entretanto, componentes de software são mais complexos, uma vez que um componente de software pode ser usado como um aplicativo completo, dependendo da tarefa que ele executa. Ou seja, apesar de serem mais independentes, eles não são tão flexíveis quantos os frameworks, mas podem ser reutilizados em outros softwares, desde que seja possível acoplar o componente de software, sem modificar a aplicação já existente.

Uma aplicação ter seu próprio conjunto de componentes, bem como pode usar desenvolvidos por outras pessoas, e que já são usados em outras aplicações. Por exemplo, uma máquina em que um cliente passa o cartão de débito/crédito necessita acessar o sistema da operadora do cartão para verificar a confirmação da operação de crédito. Quem está fazendo o sistema de vendas, ao invés de desenvolver um componente que executa a operação de pagamento, ele usa um componente de software desenvolvido, e fornecido pela própria operadora de cartão.

Ainda neste exemplo, se acontecer algum problema com o sistema da operadora de cartões, a máquina de cartão não pode ficar esperando indefinidamente uma resposta. Por isso, um componente de software deve ser capaz de executar sua tarefa sem depender do resultado de outros componentes. No caso deste exemplo, se a máquina de cartão não receber uma resposta do sistema da operadora, ela deve ser capaz de parar o que está fazendo e ficar pronta para um novo uso, o que nem sempre ocorre com o uso de frameworks;

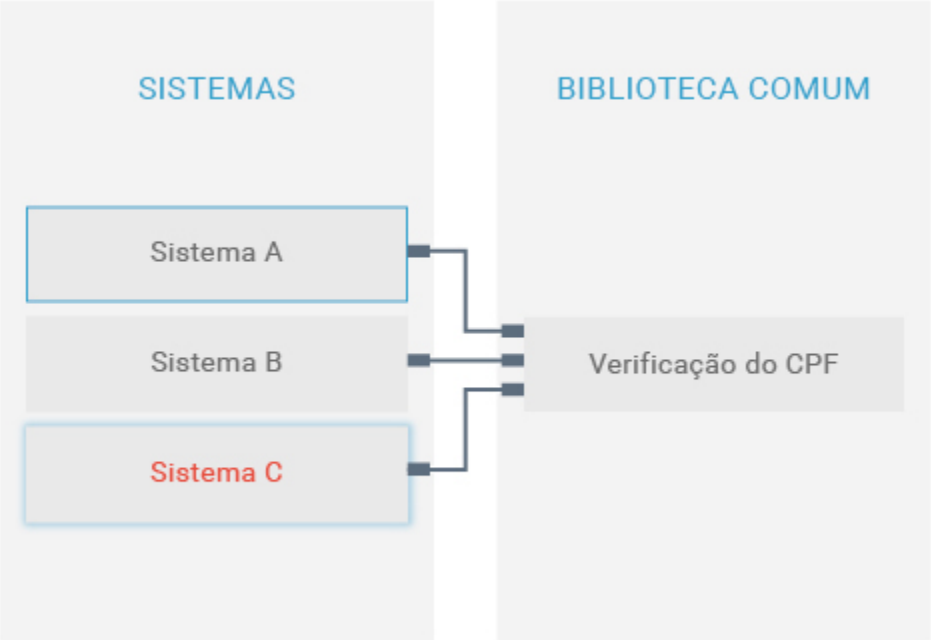
Porque usar Componentes de Software?

Nos últimos 30 ou 40 anos, a indústria de software evoluiu de sistemas chamados de "monolíticos", pois os computadores eram apenas uma interface para que um usuário pudesse digitar e ler dados armazenados em sistemas de bancos de dados. Com a popularização do uso dos PCs em residências e empresas de pequeno e médio porte, os sistemas evoluíram para outro modelo de arquitetura, pois os trabalhos começaram a ser realizados em paralelos, e muitas vezes, mais de um usuário necessitava ter acesso aos dados que acabaram de ser atualizados por outro usuário do sistema. Assim, começaram a surgir propostas para dividir o processamento das informações entre os PCs e os servidores.

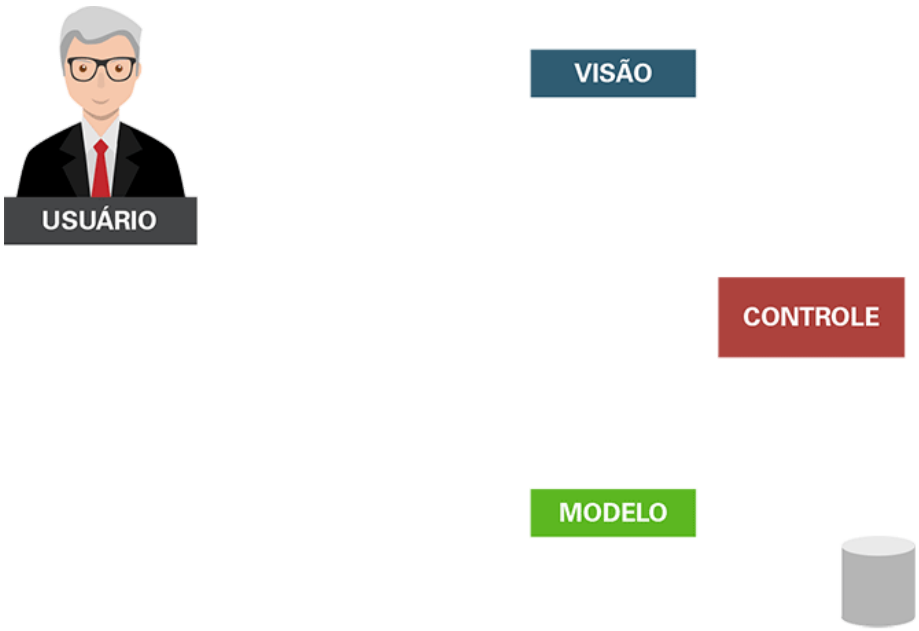
A partir desse momento, as aplicações de software que antes eram executadas isoladamente num único servidor, começaram a serem executadas em servidores interligados, o que possibilitou interligar as filiais de grandes corporações e, em alguns casos, interligando diretamente seus sistemas aos sistemas de fornecedores e/ou clientes. Com o advento da web, as empresas viram mais um canal de interação com clientes e fornecedores, e a Internet passou a ser vista como uma plataforma de serviços, assumindo o papel que antes era dos aparelhos de fax e telefones.

Diante deste cenário em constante evolução, rapidamente os arquitetos de software e de sistemas perceberam que a inflexibilidade e a interoperabilidade das antigas aplicações operando em duas camadas dificultavam as tarefas de integração e, até mesmo, o desenvolvimento dos novos aplicativos, pois a tecnologia daquele momento inviabilizava o reuso de determinada funcionalidade. Além disso, era difícil alterar aquelas aplicações, impedindo que uma funcionalidade pudesse ser adaptada a um novo uso. Assim, era comum ter um aplicativo replicado em vários servidores, sendo que cada um deles executava a mesma tarefa, causando uma ocupação inútil de espaço de armazenamento e de recursos para processamento.

Após tentativas bem-sucedidas e frustradas, chegou-se à conclusão de que se um sistema fosse projetado sob a forma de componentes de software, ele seria flexível para ser facilmente alterado, além disso, ela seria mais facilmente integrada a outros sistemas. Outra vantagem dos componentes de software é que eles podem ser adaptados para funcionar segundo vários modelos de arquiteturas. Essa facilidade fez com que alguns desenvolvedores optassem por agrupar componentes de software com funções semelhantes, dentro de camadas de serviços, para facilitar o trabalho de modelagem da arquitetura.



Legenda: FIGURA 1 - EXEMPLO DE SISTEMAS BASEADOS EM COMPONENTES DE SOFTWARE.



Legenda: FIGURA 2 - FUNCIONAMENTO DO MODELO MVC

Modelo - Visão - Controle (MVC)

Um dos primeiros padrões de projeto a surgir, mesmo antes dos componentes de software, o padrão de projeto denominado MVC, também conhecido como Model-View-Controller, orienta o desenvolvedor a organizar seus componentes em 3 camadas:

- Modelo: esta camada agrupa os componentes de software que executam as regras de negócios, gerenciamento de conexão com as bases de dados, além de lógica e funções necessárias para manter o software sendo executado;
- Visão: esta camada agrupa os componentes que permitem ao usuário do software informar, ou ter acesso, aos dados que o sistema tráfega;
- Controle: esta camada agrupo os componentes de software que controlam a troca de informações entre os componentes que estão instalados nas outras duas camadas;.

A interação destas camadas ocorre conforme demonstrado na Figura 2. Os componentes da camada *Controle* enviam comandos para os componentes da camada *Visão* atualizarem o que é apresentado ao usuário, ou da camada *Modelo*, para atualizar os estados de objetos quando houver alteração em seu estado na memória. Os componentes da camada *Modelo* notificam os componentes das camadas *View* e *Controle* a respeito das mudanças que ocorrem no estado dos objetos. Assim, essas duas camadas podem atualizar a interface de usuário que mostra o resultado dessa alteração e processar essas alterações. Os componentes da camada *View* requisitam aos componentes nas camadas *Controller* e *Model* sobre possíveis atualizações na interface de usuário para mostra-las aos usuários.

Modelos em 3 Camadas

O MVC surgiu durante a década de 1970, para um determinado cenário que havia no desenvolvimento de software, e o seu uso vem sendo aperfeiçoado até os dias atuais, pois surgiram novas tecnologias, como linguagens de programação, além de novos procedimentos, frameworks e outros padrões de projeto, além dos ambientes de desenvolvimento, que melhoraram bastante.

Atualmente, usuários móveis são uma realidade, e não apenas uma possibilidade, e devem ser considerados quando um novo software é desenvolvido. E num primeiro momento, eles seriam os componentes que estariam na camada Visão do modelo. Entretanto, o aplicativo que é executado no dispositivo móvel também possui seu próprio modelo de arquitetura, adaptado para a linguagem de programação em que ele foi desenvolvido e para o ambiente em que ele foi projetado para ser executado.

Isso fez com que apesar de ainda se usar o termo MVC, os softwares não estão organizados apenas nas 3 camadas iniciais propostas pelo MVC, pois os componentes que estavam agrupados na camada Model precisaram ser redistribuídos em outras subcamadas, e cada camada com uma função específica. Assim, começamos a ter as camadas de rede, que agrupam todos os componentes de software que executam o gerenciamento do uso da rede de dados, a camada de dados, que agrupa todos os componentes de software que executam as transações nos bancos de dados, e assim vai.

Ou seja, na maioria das aplicações, a camada Modelo passou a executar várias funções, e ficou mais simples dar a ela os nomes das subcamadas, de modo a deixar mais claro o que um determinado conjunto de componentes de software faz, ao contrário do que acontecia quando era usada apenas a camada Model. Assim, alguns arquitetos começaram a usar os termos arquitetura em 3 camadas ou arquitetura em N camadas, pois um sistema poderia ter várias camadas para estruturar o software, e não mais as três camadas iniciais propostas pelo MVC.

ATIVIDADE FINAL

A diferença básica de um modelo de arquitetura em N camadas para o modelo de arquitetura MVC, pode ser entendido como definir os nomes das camadas propostas no MVC, cada qual com uma função específica, já definida pelo modelo, para nomes que representem funcionalidades do software a ser desenvolvido, de acordo com as necessidades do desenvolvedor, que normalmente atribui à camada o nome de uma função do software, e ali agrupa todas as partes que desempenham aquela função.

A. Verdadeiro.
B. Falso.

Assinale a sentença que possui a relação correta entra a camada do MVC e a sua respectiva função.

A. Modelo - Interação com o usuário
B. Controle - Interação com o usuário
C. Controle - Regras de negócio do aplicativo.
D. Modelo - Executa a infra-estrutura necessária para executar o aplicativo.
E. Modelo - Conexão com a camada Visão

Usar componentes de software facilitam projeto de partes do softwara que podem ser reutilizados por diversas partes do mesmo software, ou que podem ser reutizalizados em novos projetos de software, sem haver

a necessidade de duplicar esses componentes.

A. Verdadeiro.

B. Falso.

A implementação de software, de acordo com os modelos de arquitetura MVC ou em 3 ou N camadas, pode ser simplificado, se o software não for baseado em componentes de software.

A. Verdadeiro.

B. Falso.

REFERÊNCIA

LARMAN, Craig. Applying UML and Patterns: Na Introduction to Object-Oriented Analysis and Design. United States of America: Prentice Hall, 1997.

FOWLER, Martin. UML Essencial: Um breve guia para a linguagem-padrão de modelagem de objetos 3. ed. Porto Alegre, Bookman, 2005.