UNINOVE

Arquitetura de Software

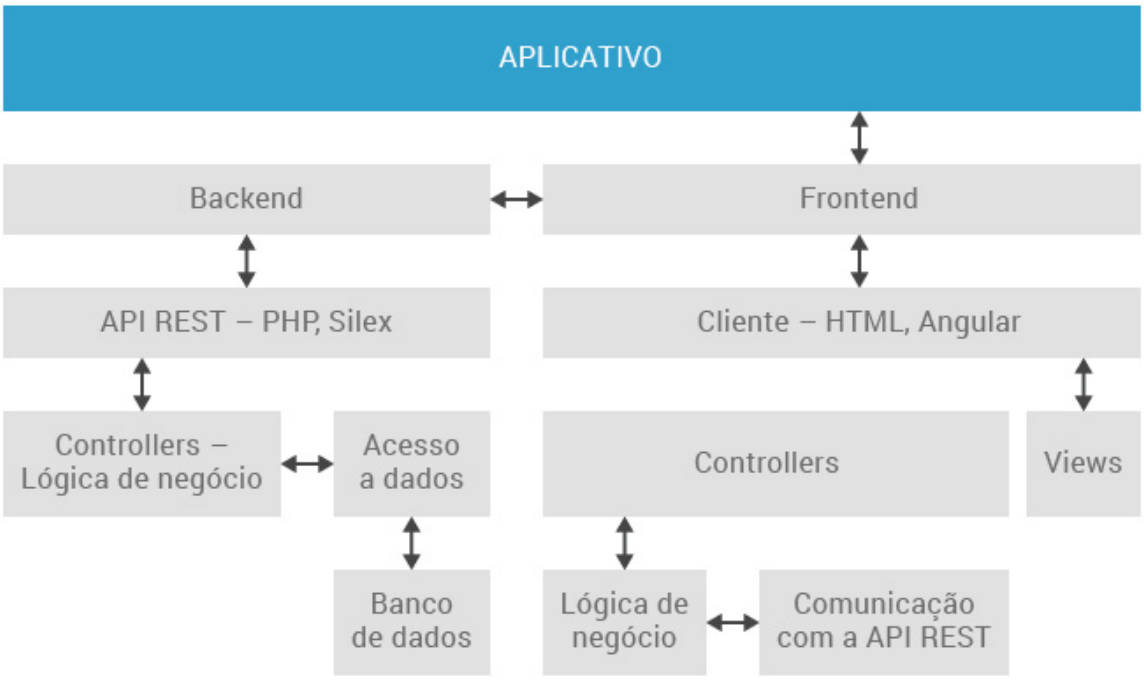
APRESENTAR OS CONCEITOS BÁSICOS RELACIONADOS À ARQUITETURA DE SOFTWARE, E COMO A UML PODE SER USADA PARA REPRESENTAR OS DETALHES ESTRUTURAIS DE UM SOFTWARE.

AUTOR(A): PROF. PAULO RICARDO BATISTA MESQUITA

Os primeiros softwares desenvolvidos ao longo dos anos 1950 a 1980, em sua maioria, automatizavam tarefas bastante específicas, e consequentemente, o software a ser implementado tinha requisitos mais simples, e muitos aspectos como interoperabilidade, escalabilidade, armazenamento de dados e segurança na proteção à informação, não eram considerados, ou não eram de grande relevância. Ao longo dos anos 1990, houve um sensível aprimoramento das tecnologias de comunicação e da informática, incluindo o uso comercial da Internet e o surgimento e popularização das telecomunicações através de telefones celulares. Como resultado, os sistemas automatizados alcançaram novos níveis de uso e de possibilidades, e os requisitos se tornaram muito mais complexos.

Essa evolução no uso e desenvolvimento dos aplicativos aumentou a complexidade do software, principalmente quando é necessário modificá-lo. O principal problema relacionado às modificações no software é determinar o quanto elas podem afetar o que já está pronto, e o quanto essas modificações podem afetar negativamente o software, exigindo novos trabalhos para corrigir eventuais problemas causados pelas modificações.

Durante os anos 1960 e 1970, os desenvolvedores perceberam a necessidade de manter documentada a organização de um software, tanto para iniciar as atividades de desenvolvimento, quanto para as atividades de manutenção. Como isso era algo análogo ao que acontecia na construção civil, usaram o nome arquitetura de software. Os primeiros cientistas a tratarem do assunto através de publicações, foram Edsger Djikstra e David Parnas, em suas pesquisas sobre processamento concorrente e desenvolvimento modular.



Legenda: FIGURA 1 - EXEMPLO DE ARQUITETURA DE SOFTWARE DE UMA API.

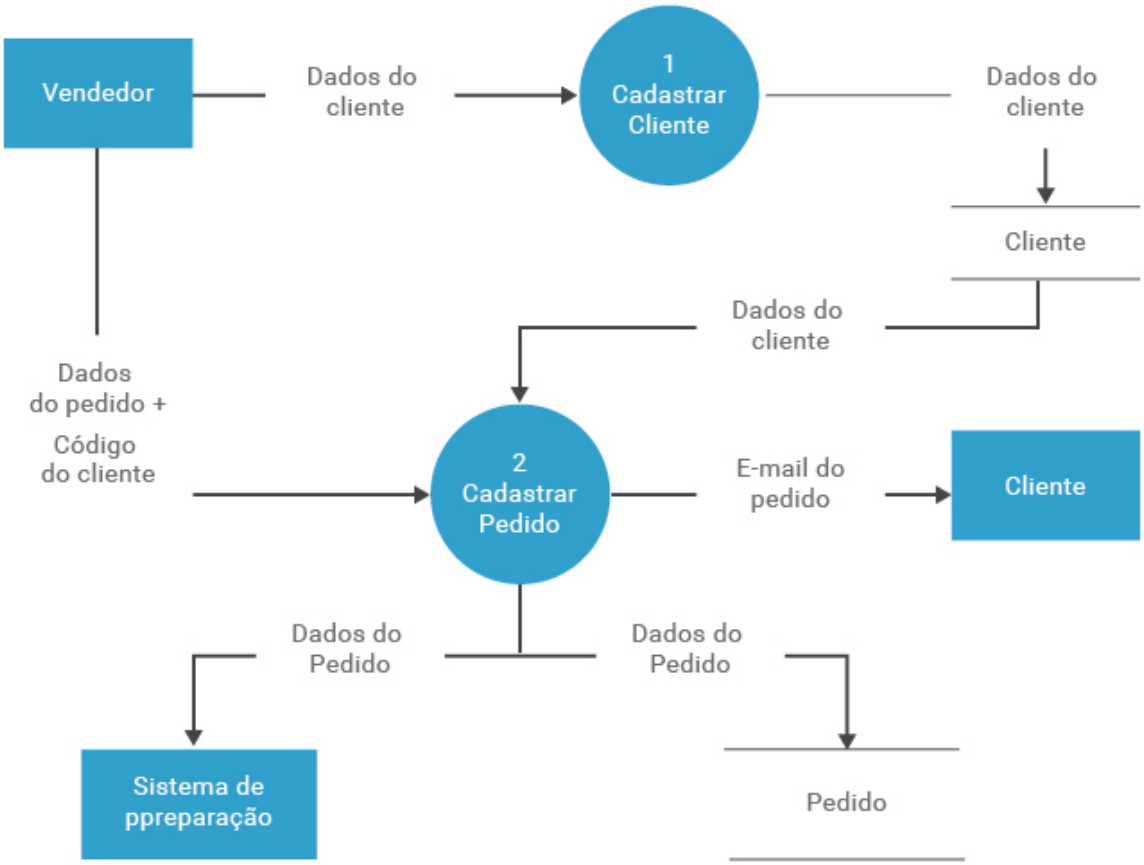
Inicialmente, a intenção da arquitetura de software era mostrar como estavam associados a escolha das estruturas de dados, os algoritmos escolhidos, de acordo com o que o processo que um determinado software estava automatizando. No início, não haviam técnicas definidas, por isso, cada desenvolvedor fazia a seu modo, mas durante os anos 1990, começaram a se tornar populares o uso de modelos de arquitetura, padrões de projetos, padronização dos estilos de codificação, melhores práticas de programação, frameworks. A popularização da orientação a objetos ajudou a popularizar esses conceitos, uma vez que as principais linguagens de programação apresentavam modelos já baseados nesses conceitos.

Muito desse trabalho está centrado no SEI, sigla do Instituto de Engenharia de Software da Universidade Carnegie Mellon, acessível através do link http://www.sei.cmu.edu. Apesar de não haver uma determinação formal quanto a isso, muito do que é proposto em trabalhos publicados através do SEI, é bastante usado como padrão no desenvolvimento de software.

Para projetar um software, há dois modelos principais, que são a análise estruturada, e a análise orientada a objetos. Os dois modelos orientam o desenvolvedor sobre o que ele deve considerar na hora de projetar um software, e de como ele deve ser documentado.

A análise estruturada se tornou popular durante os anos 1980, e ela orienta o desenvolvedor a projetar o software, considerando os processos que estão sendo automatizados, e na forma como esses processos modificam uma determinada informação. Assim, a partir do conceito operacional de um sistema, devem ser identificados com que dados um processo é iniciado, onde eles são armazenados, e quais módulos de software devem ser desenvolvidos para processar esses dados. Esse modelo de análise foi desenvolvido

no auge dos aplicativos estruturados na arquitetura cliente-servidor, então, ao aplicar esse modelo de análise na documentação de um software, era necessário indicar quais módulos seriam executados pelo servidor e pelo cliente. Um dos principais métodos da análise estruturada é o método apresentado por Edward Yourdon, conhecido como YSM, sigla para o Método Estruturado de Yourdon (tradução de Yourdon Structured Method), mas há vários outros, como o IDEF0, que é usado por vários órgãos do governo norte-americano, ou o SSADM (Método de Análise e Projeto de Sistemas Estruturados), que é usado para documentar os sistemas usados por alguns órgãos do governo do Reino Unido.

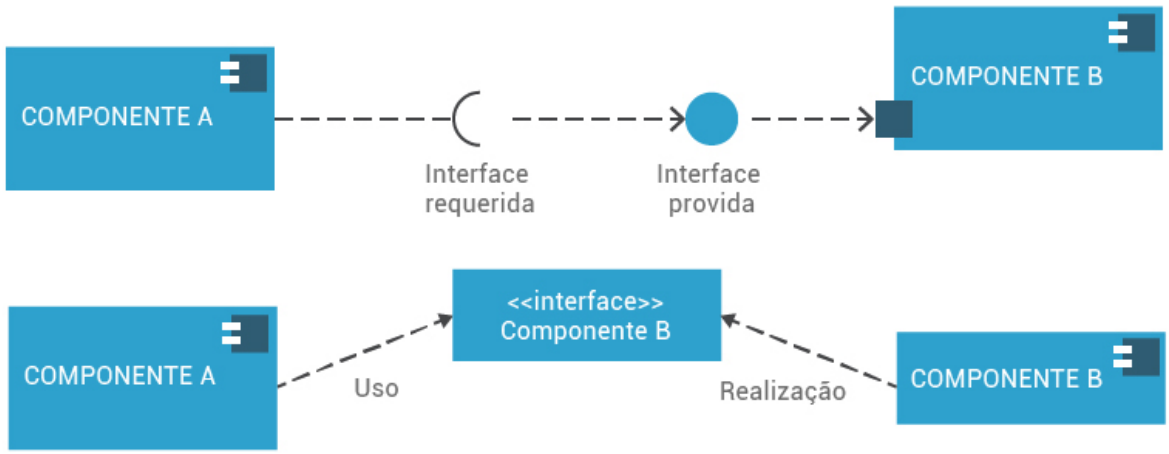


Legenda: FIGURA 2 - EXEMPLO DO DIAGRAMA DE FLUXO DE DADOS PROPOSTO PELA METODOLOGIA DE EDWARD YOURDON

Com o advento da Internet, e das linguagens de programação orientadas a objetos durante os anos 1990, a análise orientada a objetos se tornou bastante popular, mas ao contrário do que aconteceu com a análise estruturada, em que vários métodos eram amplamente usados, o método criado por Ivar Jacobson, Grady Booch e James Rumbaugh, se sobressaiu aos demais, e foi adotado como padrão da análise orientada a objetos.

O método de trabalho deles consistia em determinar, a partir do conceito do sistema, quais seriam os objetos que interagiriam com o sistema, e como cada um deles é afetado, ou se ele afeta, o sistema no qual está inserido. O método de trabalho deles era apoiado por uma linguagem de documentação, que pode ser usada para documentar várias partes do software, como as entidades (objetos) que interagem com o sistema, os processos que precisam ser automatizados, como os objetos interagem entre si em relação ao tempo, ou relação a prioridades para processas dados, organização do software em termos de componentes de software, etc.

Essa linguagem recebeu o nome de UML, sigla para Linguagem Unificada de Modelagem, e recebeu esse nome, pois a comunidade de desenvolvedores percebeu a facilidade em ter um método padronizado, ao contrário de vários métodos concorrentes entre si, como aconteceu com os métodos da análise estruturada.



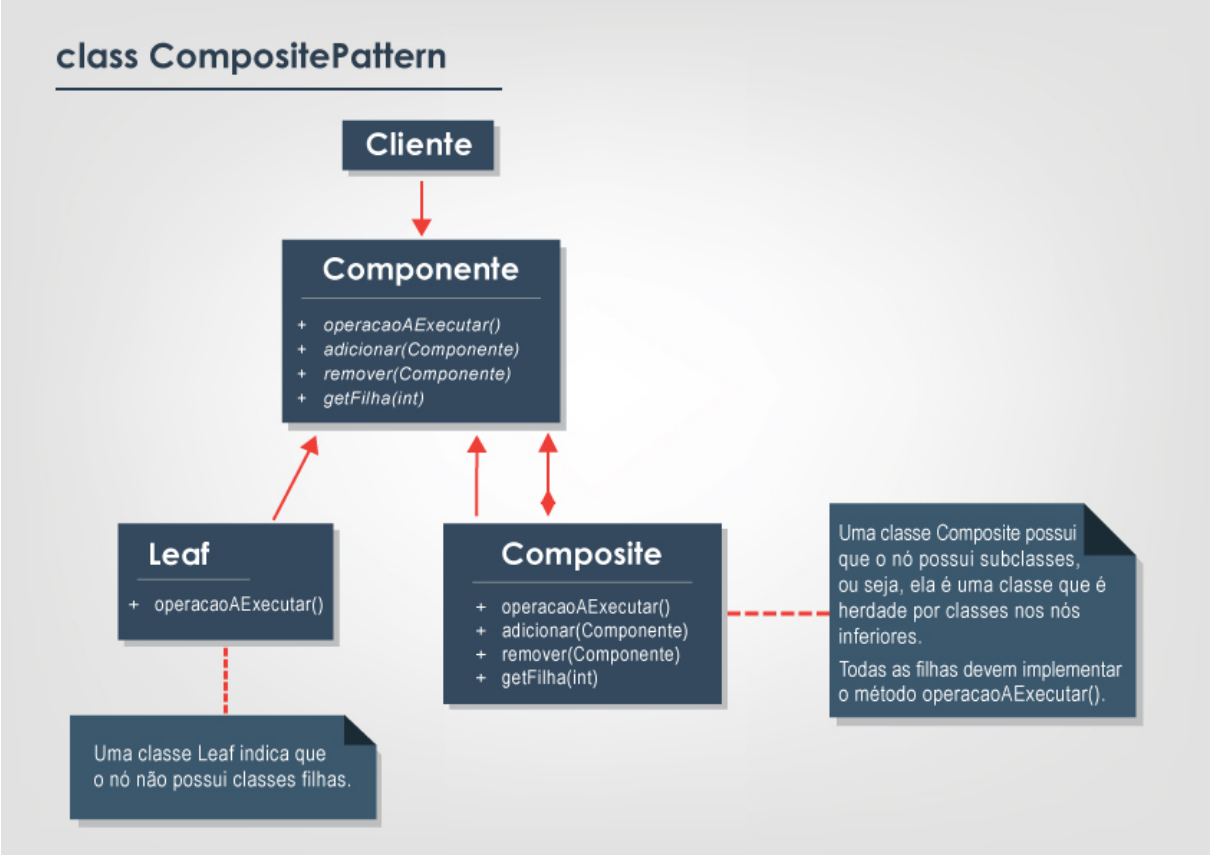
Legenda: FIGURA 3 - ELEMENTOS DE UM DIAGRAMA DE COMPONENTES UML, USADO PARA REPRESENTAR A ARQUITETURA DE SOFTWARE.

A UML continua sendo atualizada, e os trabalhos estão concentrados em um consórcio, que mantém um site (http://www.uml.org (http://www.uml.org/)) para manter contato com a comunidade dos desenvolvedores, disponibilizando atualizações, divulgando melhores práticas, e ouvindo as necessidades dos desenvolvedores. Assim, a UML se atualiza durante o tempo, e possui várias versões de especificações.

Entre os vários diagramas que a UML apresenta, os mais indicados para a arquitetura de software são: diagramas de classes; diagramas de componentes; diagramas de instalação (ou implantação); diagrama de pacotes e os diagramas de estrutura composta.

Diagrama de Classes

Esse é um dos diagramas mais significativos da UML, e são usados para representar a estrutura e relações das classes usadas para construir o software. O diagrama de classes é conceitual, e por isso, ele pode ser usado para representar o sistema inteiro, ou partes dele. Além disso, as classes podem ser representadas sob a forma de estereótipo, representando atores, telas, relatórios ou tabelas na base de dados, etc.



Legenda: FIGURA 4 - DIAGRAMA DE CLASSES UML QUE REPRESENTA A ORGANIZAçãO DE UM SOFTWARE QUE EXECUTA UMA FUNçãO BASTANTE ESPECíFICA.

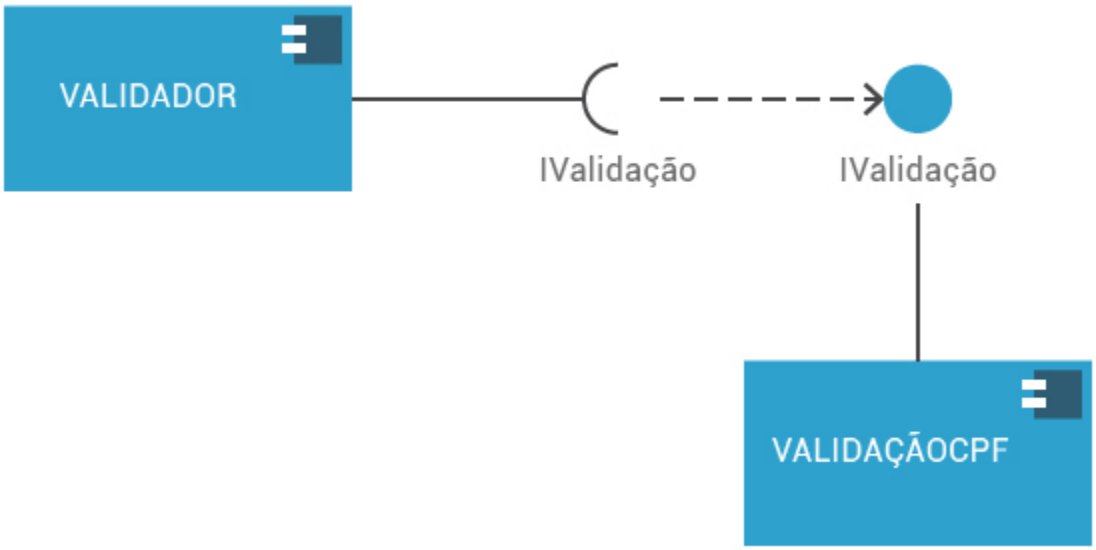
Como o diagrama mostra o relacionamento entre as classes, este diagrama também se pode ser utilizado para representar mapas objeto/relacional, que são usados por algumas tecnologias para representar a conexão entre as tabelas, e outros tipos de entidades, existentes numa base de dados, e os objetos de um sistema orientado a objetos.

Diagrama de Componentes

O diagrama de componentes mostra como as classes deverão ser organizadas de acordo com a noção de componentes de software. A UML define um componente de software como um programa que executa uma tarefa dentro do sistema. Por exemplo, um sistema pode ter um componente de software que valide

o CPF de um determinado usuário. O diagrama de componentes mostrará como ele se relaciona com outros componentes de software, quais os dados que ele precisa para realizar a validação do CPF, e quais serão as respostas que ele dará aos componentes com os quais ele se relaciona.

Esse tipo de ligação acaba sendo demostrado pelo diagrama de componentes, que demonstra a distribuição dos componentes, quais devem ser desenvolvidos, e quais já estão prontos e podem ser reusados na produção de um novo aplicativo. Além da distribuição dos componentes, o diagrama também destaca as tarefas que os componentes de software executam, os parâmetros necessários para realizar seus processamentos, e quais os resultados que serão gerados pelos componentes de software.



Legenda: FIGURA 5 - DIAGRAMA DE COMPONENTES UML, ILUSTRANDO OS COMPONENTES DE SOFTWARE QUE IMPLEMENTAM A FUNçãO DE VALIDAçãO DO CPF.

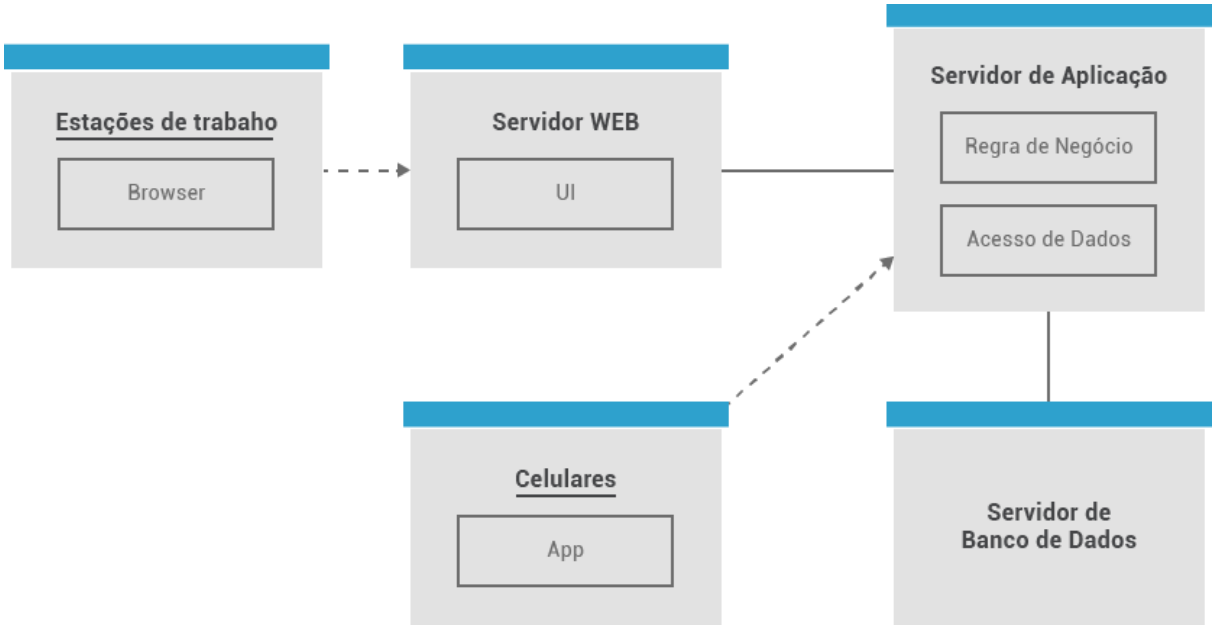
Por essas características, também é possível usar os diagramas de componentes para mostrar a distribuição de componentes de acordo com os modelos de arquitetura usados para desenvolvimento, como por exemplo, o SOA (Arquitetura Orientada a Serviços) e o MVC (Modelo-Visão-Controle).

Isso é possível porque a UML não define regras sobre como devem ser projetados os componentes de software, ou sobre como eles devem ser distribuídos. Ela também não define como eles devem ser implementados internamente.

Normalmente, não se costuma usar diagramas de componentes para aplicações de baixa complexidade, a não ser que os processos de desenvolvimento de software determinem que seja necessário fazer esses diagramas nessa situação. O uso mais comum desses diagramas é para os sistemas de alta complexidade.

Diagramas de Instalação (ou Implantação)

Este diagrama é usado para apresentar a arquitetura do sistema. Ele mostra a distribuição dos componentes de hardware e software e sua interação com outros elementos que processam os dados usados no sistema. Esse diagrama destaca a função de cada componente de hardware ou software, a forma como eles se ligam, o modelo de arquitetura aplicado, os relacionamentos de todos eles.

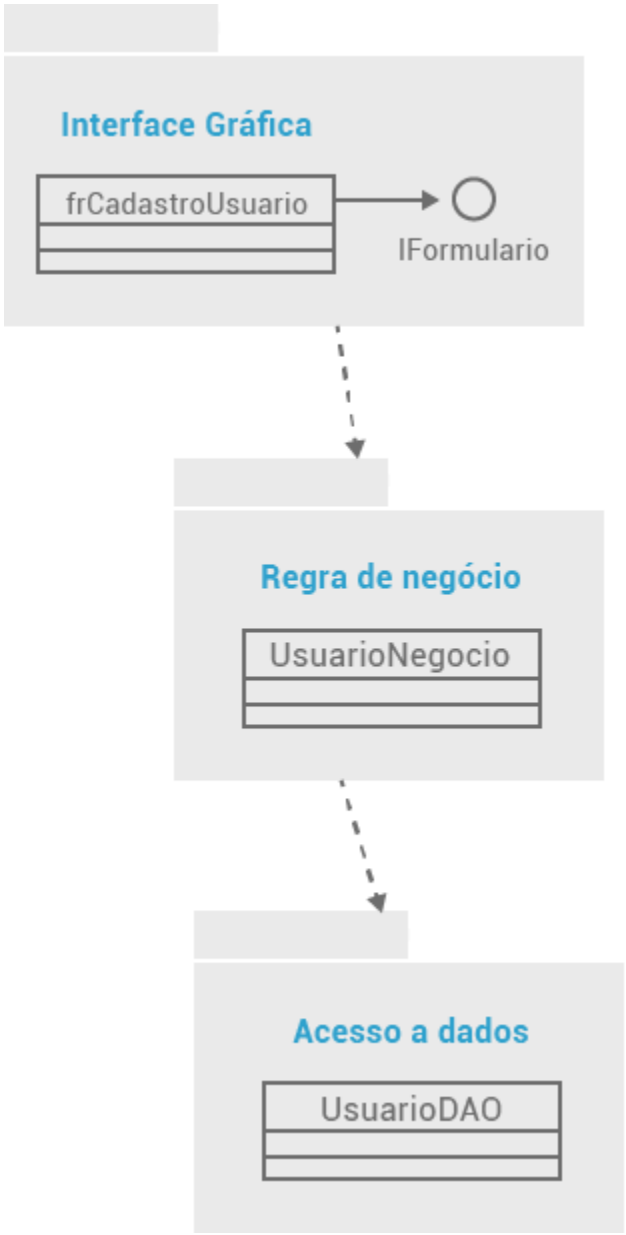


Legenda: FIGURA 6 - DIAGRAMA DE IMPLANTAçãO UML QUE ILUSTRA A INTERAçãO ENTRE OS COMPONENTES DE SOFTWARE E HARDWARE DE UM SISTEMA COM VáRIOS TIPOS DE EQUIPAMENTOS.

O diagrama de instalação não é tão usado no desenvolvimento de software. Ele costuma ser mais usado nas fases finais do desenvolvimento, apresentando a versão final do sistema, quando ele está pronto para ser instalado, ou quando se precisa ligar um novo software a um sistema já existente, e é necessário analisar o impacto sobre o que já existe, e o que será necessário para executar os novos componentes de software.

Diagrama de Pacotes

Os diagramas de pacotes ilustram como as classes que implementam um sistema estão divididas sob a forma de pacotes, que são agrupamentos lógicos usados para organizar o software a ser implementado. Normalmente, os pacotes são definidos para agrupar as classes de acordo com as funcionalidades que elas executam para o sistema. Assim, ao usar um pacote para representar uma funcionalidade do software, este diagrama ajuda a ilustrar como essas funcionalidades estão associadas, e qual o relacionamento entre elas.



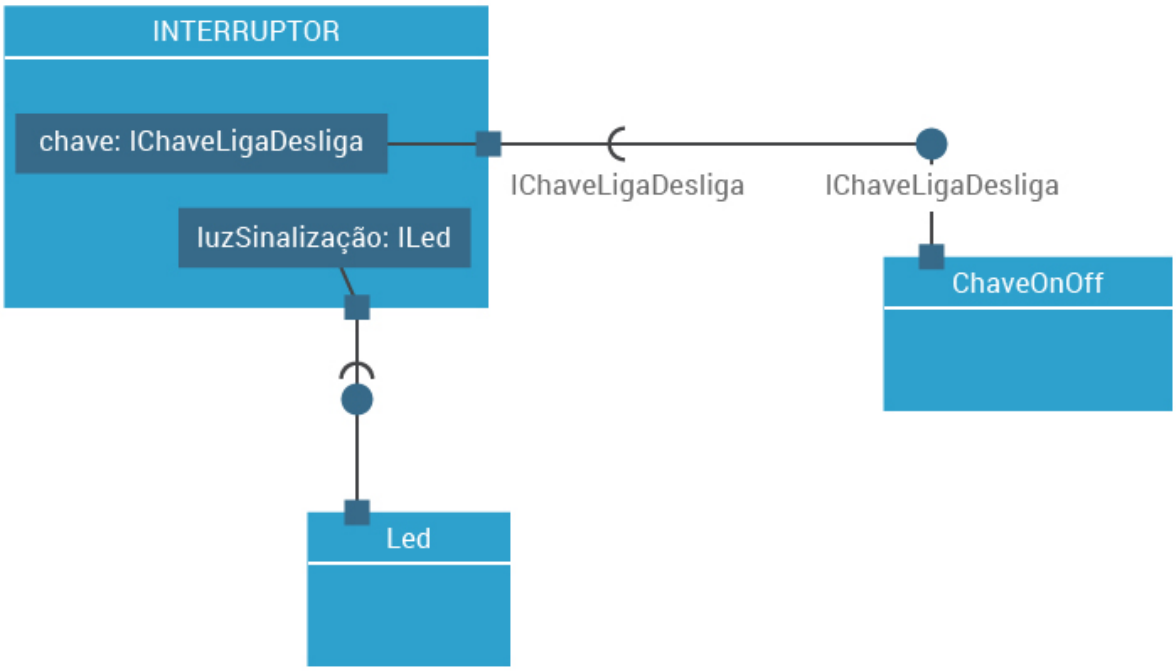
Legenda: FIGURA 7 - DIAGRAMA DE PACOTES UML QUE ILUSTRA A ORGANIZAçãO DAS CLASSES DE UM SOFTWARE, DE ACORDO COM SUAS FUNçõES

Por isso, é um dos diagramas que pode ser facilmente aplicado, para representar como um software está sendo projetado para se adequar aos modelos de arquitetura selecionados para o desenvolvimento do projeto.

Diagrama de Estrutura Composta

Este diagrama costuma ser mais usado para demonstrar a arquitetura interna de alguma funcionalidade do software, ilustrando como os elementos que implementam o software se relacionam entre si, descrevendo a colaboração interna de classes, interfaces ou componentes para especificar uma

funcionalidade do software.



Legenda: FIGURA 8 - DIAGRAMA DE ESTRUTURA COMPOSTA EM UML, ILUSTRANDO O CONTROLE DE UMA LâMPADA.

ATIVIDADE FINAL

Analisar um sistema, sob o ponto de vista de seus processos, e de como esses processos afetam os dados que trafegam pelo sistema, para identificar os módulos que poderão ser desenvolvidos para automatizar esses processos, é o modelo da:

A. Análise Orientada a Objetos

B. Programação Orientada a Objetos

C. Análise Estruturada

D. Programação Estruturada

E. DFD

Analisar um sistema, com o objetivo de determinar como as entidades que interagem com o sistema, afetam, ou são afetados, pelas informações que trafegam no sistema, é característica de:

A. Análise Orientada a Objetos
B. Programação Orientada a Objetos
C. Análise Estruturada
D. Programação Estruturada
E. UML

REFERÊNCIA

YOURDON, E. *Análise Estruturada Moderna*. São Paulo: Ed. Campus, 1990. 824 p