NÃO PODE FALTAR

TÉCNICAS DE MODELAGEM DA UML

Maurício Acconcia Dias

DIAGRAMAS UML

A UML é uma linguagem de modelagem visual que apresenta diferentes tipos de diagramas para diferentes tipos de situações e etapas do desenvolvimento.

Fonte: Shutterstock.

Deseja ouvir este material?

Áudio disponível no material digital.

PRATICAR PARA APRENDER

Olá, aluno! Bem-vindo ao início do estudo sobre os diagramas UML. A modelagem é uma fase importante do desenvolvimento de software e deve ser realizada da maneira mais clara, simples e funcional possível. A UML proporciona esses resultados por ser uma linguagem de modelagem visual que apresenta diferentes tipos de diagramas para diferentes tipos de situações e etapas do desenvolvimento. Como o mercado de trabalho solicitará de você, futuro desenvolvedor, um conhecimento sólido sobre esses conceitos, é importante criar uma boa base de conhecimento, para que você seja capaz de resolver, da melhor maneira possível, os problemas que serão enfrentados na profissão.

Recém-formado e ingressante no mercado de trabalho, você foi contratado por uma empresa de desenvolvimento de software. A experiência de iniciar na equipe de desenvolvimento de uma empresa nova já é um desafio, pois cada empresa tem seus métodos de desenvolvimento e seus padrões, e todos os times de desenvolvimento estão adaptados a esse universo, fazendo com que a empresa consiga atingir suas metas e entregar os softwares no tempo determinado, atendendo a todos os requisitos solicitados.

Apesar de todas as técnicas de desenvolvimento de software e modelos serem amplamente conhecidos, você percebeu que sua nova empresa ainda utiliza métodos antigos de desenvolvimento e, em alguns projetos, método nenhum, ou seja, os softwares não apresentam documentação, sendo difícil para um membro novo na equipe trabalhar no código. Você, um profissional recém-contratado, percebe a dificuldade para o entendimento do software desenvolvido e entende ser necessário realizar a modelagem do sistema. Além disso, conhece a importância da documentação, utilizando métodos e linguagens específicas para essa finalidade. Sabe, ainda, que a linguagem mais utilizada é a UML, pois apresenta todas as principais características desejáveis para o problema em questão.

Sendo assim, como seus conhecimentos sobre o histórico de UML, características e benefícios da linguagem estão sólidos, você acredita poder melhorar o processo de desenvolvimento e manutenção de software em seu novo emprego. Todavia, você terá que mostrar a importância da documentação e modelagem de sistemas para si e demais membros do time de desenvolvimento, mencionando a importância considerando o reuso. Você terá que, aos poucos, mostrar os conceitos, vantagens, desvantagens, motivações e justificativas para que sua empresa adote o UML e se beneficie da ferramenta em seu processo de desenvolvimento.

Você já apresentou a linguagem UML para os diretores da sua empresa, que ficaram interessados em conhecer mais acerca do tema. Então, continuando com seu objetivo de melhorar o processo de desenvolvimento da empresa de software que o contratou, você ficou entusiasmado após saber que sua primeira reunião surtiu efeito, e os funcionários querem aprender mais a respeito da linguagem UML. Agora você deve apresentar um pouco mais da teoria da linguagem UML para seus colegas e para os diretores da empresa.

Faça, portanto, um roteiro de apresentação dos diagramas UML contendo, em resumo, as informações sobre a classificação dos diagramas e a definição básica de cada um deles. Lembre-se de que é interessante relacionar exemplos de situações reais a cada um dos diagramas sempre que possível.

Vamos, então, começar a construir uma base sólida de conhecimentos sobre UML e seus diagramas? Bons estudos!

CONCEITO-CHAVE

A linguagem UML é uma importante ferramenta para a modelagem de sistemas, a qual possibilita elaborar modelos abstratos, tendo um visual do sistema e de como os objetos se comunicam, tudo mostrado na forma de diagramas. Cada modelo elaborado representa um aspecto do sistema, ou seja, suas diferentes perspectivas. Esse tipo de ferramenta, quando utilizada durante as etapas de desenvolvimento do software, proporciona uma série de benefícios e melhorias a esse processo, mesmo sendo uma linguagem independente de processos e não uma metodologia de desenvolvimento.

O fato de a linguagem ser tão bem-estruturada e não apresentar custos de utilização faz com que ela seja amplamente utilizada no mercado de trabalho e torna seu conhecimento indispensável para a análise e modelagem de sistemas.

A ferramenta dispõe de uma ampla lista de funcionalidades. No caso da UML, essas funcionalidades são os diagramas, em 14 tipos diferentes. Seria possível começar a abordá-los um a um sem qualquer tipo de diferenciação básica entre eles, porém é importante que se entenda o objetivo principal de sua utilização. Para que torne mais simples encontrar o diagrama necessário para um determinado problema, a linguagem apresenta uma classificação de tipos de diagramas.

DIAGRAMAS

Durante a criação de diagramas de modelagem constatou-se que duas questões eram principais em seu desenvolvimento: a estrutura e o comportamento do que se deseja modelar. Então os diagramas UML foram divididos em dois grandes grupos: **os diagramas UML estruturais** e **os diagramas UML comportamentais**. Há, ainda, os diagramas de integração, que basicamente fazem parte do grupo de diagramas comportamentais. A partir desses diferentes grupos de diagramas podemos ter a visão do sistema em diferentes perspectivas. Essas divisões ficarão mais claras ao longo do estudo individual dos diagramas; por ora é importante saber como podem ser classificados.

A Figura 1.1 apresenta os 14 diferentes diagramas UML como proposto em UML–Diagrams (2016).

Figura 1.1 | Classificação de diagramas UML com relação a sua natureza

Para visualizar o vídeo, acesse seu material digital.

Fonte: elaborada pelo autor.

Se desejar, baixe o texto do objeto.

Embora não seja indicada pelos manuais da linguagem UML a ordem em que devem ser criados e utilizados os diagramas em um determinado fluxo de desenvolvimento de software, normalmente se aborda os diagramas em uma ordem mais didática quando são explicados. Essa ordem é baseada na

complexidade dos diagramas, e usualmente são apresentados os diagramas mais simples primeiro para que o processo de criação seja assimilado de forma mais eficaz.

Seguindo essa ideia, começaremos de forma simplificada com os diagramas estruturais, em seguida iremos para os diagramas comportamentais, que têm uma subdivisão para os diagramas de interação, a qual será também abordada. A abordagem dos diagramas neste momento será superficial, para proporcionar um entendimento da estrutura de forma geral. Ao longo do estudo da linguagem UML cada diagrama será detalhado, de forma que sua estrutura e utilização ficarão mais claras.

DIAGRAMAS ESTRUTURAIS

Os diagramas estruturais apresentam como um determinado sistema é organizado em partes (suas estruturas), seus componentes e os relacionamentos entre esses componentes. Os diagramas estruturais muitas vezes estão associados à modelagem estática, pois mostram a estrutura do sistema. Em geral, os diagramas estruturais são elaborados no momento do projeto da arquitetura do sistema. Eles representam os conceitos significativos do sistema como abstrações, questões de implementação e do mundo real.

Os diagramas estruturais, como apresentado na Figura 1.1 são sete, descritos a seguir, iniciando com o diagrama que provavelmente é um dos mais utilizados até por desenvolvedores que não estão totalmente familiarizados com os conceitos da UML: o **diagrama de classes**.

DIAGRAMA DE CLASSES

Como um dos fundamentos da criação da linguagem UML é o paradigma de programação de Orientação a Objetos, o diagrama de classes remete às classes criadas em um software desenvolvido em uma linguagem orientada a objetos.

O objetivo do diagrama de classes é representar as classes, suas definições e as relações entre elas.

Segundo a definição do paradigma de orientação a objetos, uma **classe** é uma abstração que descreve entidades do mundo real e quando instanciadas dão origem a objetos com características similares. A **abstração classe** é composta por atributos e métodos.

- Os **atributos** são como as variáveis ligadas ao conceito apresentado, por exemplo: considerando uma classe pessoa, tudo que uma pessoa “possui” (nome, endereço, profissão, sexo, naturalidade) são seus atributos.

- Os **métodos** são como as funções, por exemplo: tudo que uma pessoa “pode fazer” (andar, comer, falar, trabalhar).

Além de suas características, as classes também se relacionam de formas variadas. Algumas dessas relações são a herança e o polimorfismo conforme exemplos a seguir:

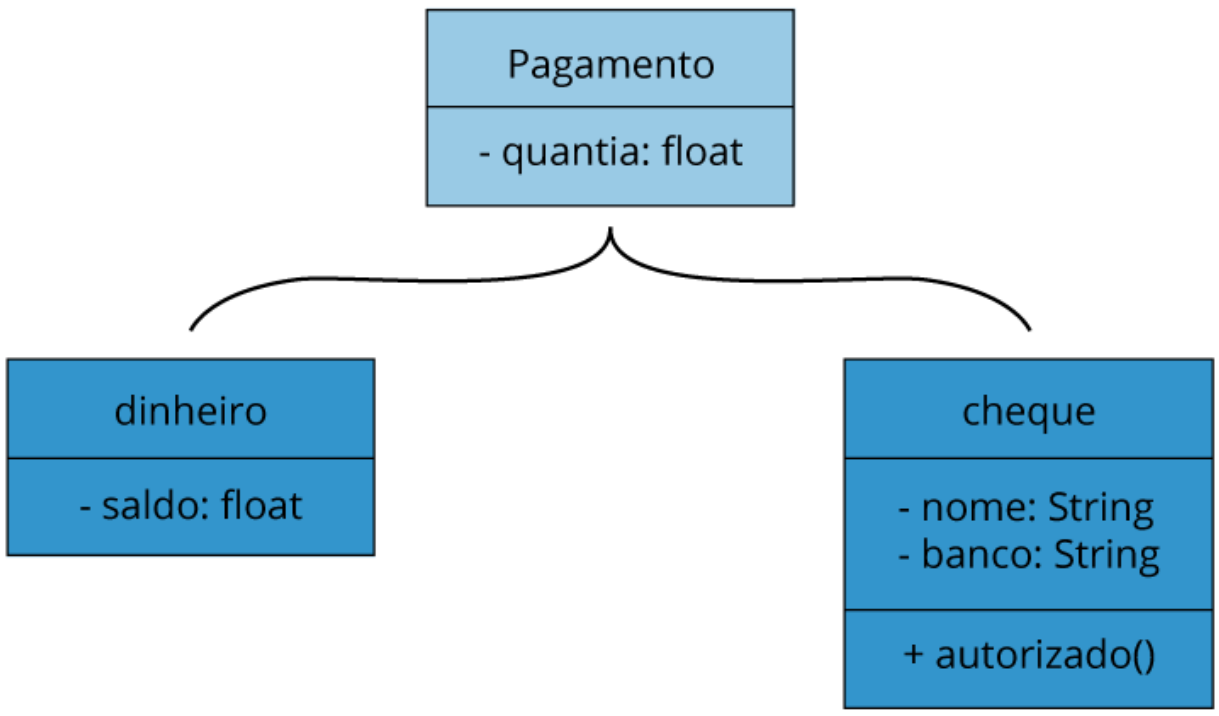
- **Herança:** quando vamos modelar veículos como classes, algumas características são comuns e outras são específicas. Um carro de passeio não tem uma caçamba como um veículo utilitário. Sendo assim, as características comuns são colocadas em uma "superclasse", e as específicas são colocadas em uma "subclasse" que herda as características da superclasse. Então, nesse caso, pode haver uma superclasse veículo que apresenta rodas, portas e motor como atributos, e acelerar, frear e virar como métodos. Também poderia haver a subclasse utilitário, que herda todas essas características da superclasse veículo, incluindo a caçamba nos atributos e o método carregar, por exemplo.
- **Polimorfismo:** imagine que há uma classe que modela um grupo de animais. Todos os animais se locomovem, porém uns andam, outros rastejam e outros ainda voam. O polimorfismo trata esses casos, por exemplo, com a implementação do método andar na superclasse animal e a redefinição desse método para cada subclasse, de acordo com a característica específica do animal a ser modelado.

Os atributos e métodos que definem as classes são também representados no diagrama de classes da UML. Relações representadas no diagrama de classes apresentam como as classes se comunicam e se relacionam.

EXEMPLIFICANDO

Vejamos um exemplo simples do diagrama de classes:

Figura 1.2 | Exemplo de diagrama de classes



Fonte: elaborada pelo autor.

Nesse exemplo, é possível visualizar a classe Pagamento definida como superclasse das classes dinheiro e cheque. A seta saindo das classes dinheiro e cheque e chegando na classe Pagamento indica herança. No caso da classe dinheiro há apenas um atributo **saldo**; já na classe cheque temos o atributo **nome**, outro atributo **banco** e o método **autorizado**. Esse é um exemplo simples que ilustra o diagrama de classes e facilita a familiarização com o conceito.

DIAGRAMA DE PACOTES

O diagrama de pacotes representará os subsistemas contidos no software desenvolvido e as grandes áreas de cada um desses sistemas. Nesse diagrama é possível separar e indicar interfaces de usuário, bancos de dados, módulos de segurança do sistema e pacotes administrativos.

DIAGRAMA DE COMPONENTES

O diagrama de componentes, como diz o nome, modela estruturalmente a relação dos componentes utilizados no software.

Um componente é uma parte do seu software que não foi desenvolvida por você, que já vem pronta para utilização e realiza uma função específica.

Quando vamos utilizar uma função matemática em linguagem C e incluímos a biblioteca *math.h*, por exemplo, ela pode ser classificada como um componente. O diagrama de componentes é importante, pois demonstra em quais partes da arquitetura do sistema foram utilizados componentes, e isso torna mais claro o motivo das decisões (RUBAUGH; JACOBSON; BOOCH, 2004).

DIAGRAMA DE PERFIL

A UML define o modelo de perfis como “leve e de extensão”. Um **perfil** é uma redefinição de uma classe para um determinado domínio ou plataforma. Por exemplo: um perfil determinado de usuário pode ser diferente no caso de um servidor ou de um dispositivo móvel. O diagrama de perfis é utilizado para apresentar os diferentes perfis necessários no desenvolvimento de um software. Antes de sua inclusão na UML, os perfis eram descritos utilizando outros diagramas.

DIAGRAMA DE INSTALAÇÃO

O diagrama de instalação descreve a estrutura de hardware e software necessária para a correta execução do software em desenvolvimento. No diagrama é possível relacionar quais componentes do software desenvolvido serão executados em cada um dos nós de hardware descritos.

DIAGRAMA DE OBJETOS

O diagrama de objetos demonstra os objetos e os seus relacionamentos em tempo de execução. Imagine um software com um menu de seis opções, das quais você sempre utiliza somente cinco. Na prática, a relação expressa pela sexta opção não

impacta o sistema, pois você não a usa. Esse diagrama tem a função de expressar essas relações justamente porque se existe algo modelado que nunca é usado, esse algo precisaria estar na modelagem?

A ideia é que o diagrama de objetos auxilie na análise de multiplicidades de objetos e relações na prática.

Repare que ele é classificado como estático, mas pode necessitar de atualizações, pois à medida que a implementação evolui os objetos e suas relações mudam.

DIAGRAMA DE ESTRUTURA COMPOSTA

A mesma análise pode ser feita com relação aos componentes utilizando o diagrama de estrutura composta, que apresenta relações entre os objetos e componentes do software desenvolvido em tempo de execução. A diferença entre esses diagramas é que o de objetos apresenta apenas os objetos, e o de estrutura composta apresenta os componentes.

Agora que já conhecemos os diagramas estruturais, podemos avançar para os diagramas comportamentais. Esses diagramas são diferentes dos que foram apresentados até este momento, por focarem o comportamento entre os componentes do sistema e não mais em como são organizados.

DIAGRAMAS COMPORTAMENTAIS

Os diagramas comportamentais têm como objetivo mostrar o fluxo de informações e os eventos do sistema longo do tempo. Em outras palavras: apresenta a resposta do sistema a algum evento do seu ambiente, mostrando, assim, o comportamento dinâmico dos objetos em um sistema e como o sistema reage a determinadas ações/eventos.

Os diagramas de interação fazem parte do grupo de diagramas comportamentais, e a partir dele pode-se modelar as interações do sistema ou a interação entre os componentes de um sistema.

DIAGRAMA DE CASOS DE USO

O primeiro diagrama a ser tratado será o diagrama de casos de uso. Ele é um dos primeiros a ser desenvolvido na modelagem de um sistema, por apresentar uma visão geral das funcionalidades do sistema da perspectiva dos usuários. Seria, em outras palavras, a apresentação da forma como o usuário pode usar o sistema que será desenvolvido.

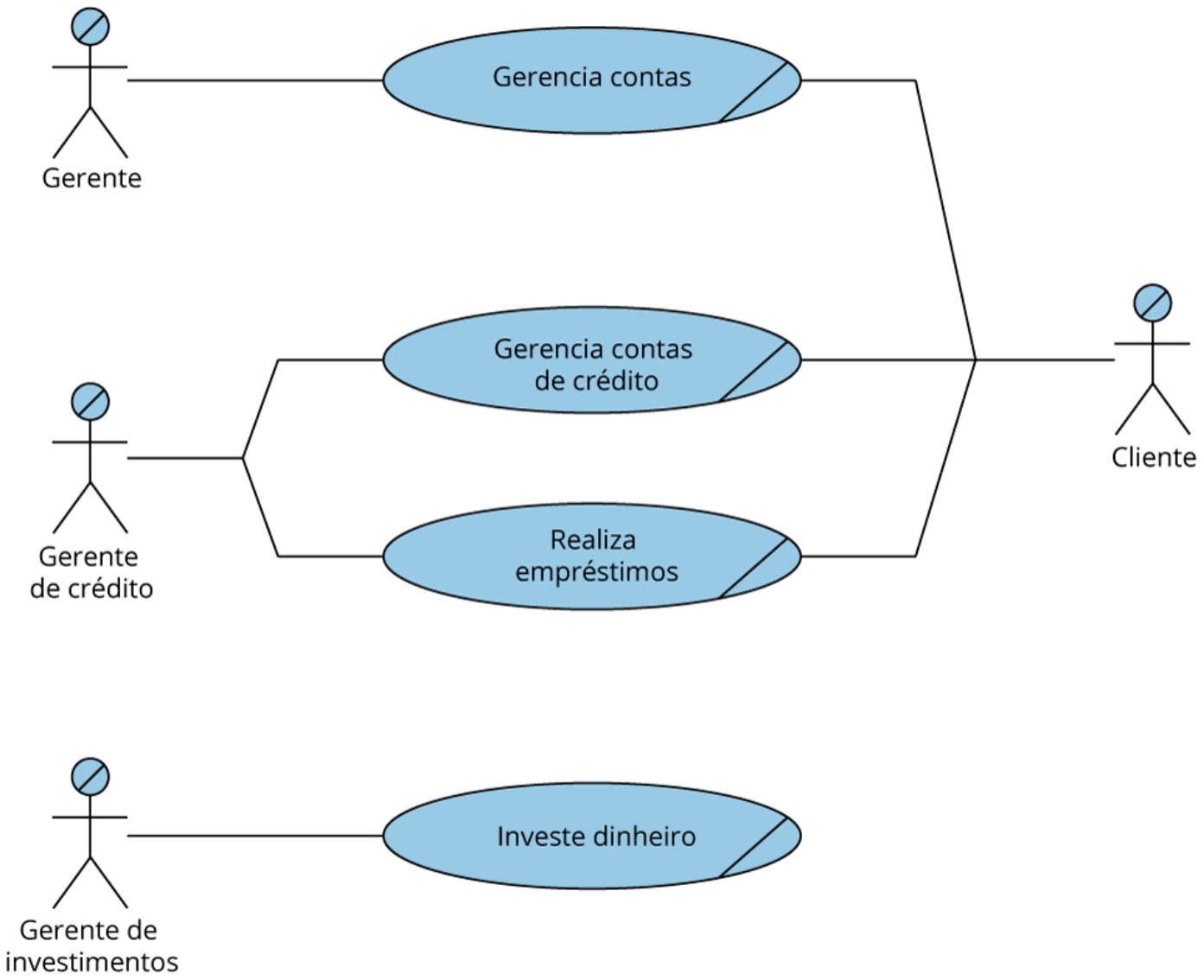
Esse diagrama é muito simples de interpretar e tem grande importância nas fases iniciais do desenvolvimento do software, já que proporciona ao cliente uma visão completa do que será desenvolvido.

Nele, caso alguma funcionalidade estiver faltando ou tenha sido descrita de forma errada, a falha será facilmente identificada e corrigida. A correção na fase inicial do processo de desenvolvimento será de grande auxílio para a diminuição de erros.

EXEMPLIFICANDO

Vejamos um exemplo simples do diagrama de casos de uso:

Figura 1.3 | Exemplo de diagrama de casos de uso



Fonte: elaborada pelo autor.

Nesse diagrama de casos de uso temos as relações que acontecem em um banco com três tipos de gerente e um cliente. Os “bonequinhos” são o que chamamos de atores (por realizarem as ações), os balões são os casos de uso e as setas ou traços representam as comunicações presentes na relação. Portanto, o cliente pode gerenciar as contas, as contas de crédito e realizar empréstimos. O gerente gerencia contas, o gerente de crédito gerencia contas de crédito e realiza empréstimos. Somente o gerente de investimentos pode investir o dinheiro.

DIAGRAMA DE ATIVIDADES

O diagrama de atividades é importante por complementar o diagrama de casos de uso, apresentando os fluxos que ocorrem no sistema como um todo. Para cada possibilidade criada no diagrama de casos de uso (usuário pode abrir um arquivo novo e iniciar sua edição, por exemplo), o fluxo da interação será descrito no diagrama de atividades. O interessante é que não só os fluxos normais são apresentados, mas também os alternativos e as exceções, o que torna o entendimento do sistema como um todo mais completo.

DIAGRAMA DE VISÃO GERAL DE INTERAÇÃO

O diagrama de visão geral de interação apresenta todas as relações que ocorrerão no sistema em alto nível. A importância desse diagrama é que ele mostra como os diferentes diagramas UML se relacionarão e as dependências entre eles. Ele é o último tipo de diagrama comportamental estático.

DIAGRAMA DE TRANSIÇÃO DE ESTADOS

O diagrama de transição de estados demonstrará, essencialmente, o ciclo de vida de determinado objeto em tempo de execução. Esse ciclo de vida mostra todos os estados possíveis para o objeto quando o programa está em execução, e as condições para a mudança entre esses estados. Por exemplo, após fazer o *login* em um sistema, o usuário normalmente tem um conjunto de opções e, para cada escolha, uma sequência de ações pode ser realizada. O mapeamento de todas as opções para todas as escolhas vai gerar o diagrama de transição de estados.

DIAGRAMAS DE SEQUÊNCIA E DE COLABORAÇÃO

A modelagem das interações entre objetos ao longo do tempo é apresentada pelo **diagrama de sequência**. Nesse caso, são representadas as possíveis sequências de trocas de mensagens e informações realizadas entre os objetos ao longo da execução do software. Os **diagramas de colaboração (ou comunicação)** têm o mesmo objetivo do diagrama de sequência, porém a disposição da informação e a maneira como a informação é apresentada são diferentes. O foco do diagrama de sequência é a relação temporal, ou seja, a ordem em que as mensagens são trocadas, enquanto o foco do diagrama de colaboração está na estrutura da relação dos objetos, sem relação com informações temporais.

DIAGRAMA DE TEMPO

Encerrando a análise inicial dos diagramas UML falaremos do diagrama de tempo. Esse diagrama tem por objetivo apresentar a execução do sistema como um todo em períodos de tempo. A importância dessa representação é a possibilidade de ver todos os objetos ativos em um determinado instante de tempo e as relações que podem estar ocorrendo entre eles. Para encontrar a solução de problemas de implementação, esse diagrama é de grande importância.

ASSIMILE

No início da apresentação dos diagramas foram citadas duas categorias para os diagramas: estrutural e comportamental. É possível, após a análise dos sete diagramas estruturais, entender que os diagramas que apresentam alguma estrutura do software são classificados como estruturais. Os outros oito diagramas são focados no comportamento das estruturas do sistema e não em como são organizados.

Também fica mais simples compreender que um diagrama será estático se não apresentar informações que sofram mudanças ao longo do tempo de desenvolvimento do software, como o comportamento em tempo de execução. Já os diagramas dinâmicos apresentam informações sobre o comportamento em tempo de execução e, portanto, mudam à medida que o software evolui e sofre modificações.

REFLITA

Após uma visão geral dos diagramas UML é possível analisar uma questão que fica evidente em sua utilização. Seria realmente necessário construir todos os diagramas para todos os sistemas? A resposta para essa pergunta é complexa e está relacionada com a complexidade do sistema a ser desenvolvido e com o número de pessoas envolvidas no desenvolvimento. Tente refletir sobre essa questão, considerando sistemas de tamanhos diferentes sendo desenvolvidos em empresas de tamanhos diversos.

Ver anotações 0

Conhecer a linguagem UML como um todo é interessante, pois nos permite tentar encontrar situações em que cada um dos diagramas é aplicável, e quais problemas podem solucionar. Portanto, continue seus estudos e descubra de que maneira construir cada um dos diagramas e como utilizá-los no desenvolvimento de software para obter o melhor resultado possível. Bom trabalho!

REFERÊNCIAS

COSTA, A. N.; WERNECK, V. M. B.; CAMPOS, M. F. Avaliação de Ferramentas para Desenvolvimento Orientado a Objetos com UML. **Cadernos do IME**. Série Informática. V. 25. 2008. Disponível em: https://bit.ly/313DUlS. Acesso em: 1 jul. 2020.

OMG – OBJECT MANAGEMENT GROUP. Página inicial. 2020. Disponível em: https://bit.ly/3360Aof. Acesso em: 19 maio 2020.

RUBAUGH, J.; JACOBSON, I.; BOOCH, G. **The Unified Modeling Language Reference Manual**. 2. ed. Pearson Higher Education, 2004.

UML – DIAGRAMS. **The Unified Modeling Language**. 2016. Disponível em: https://bit.ly/3f7qM41. Acesso em: 19 maio 2020.

UNHELKAR, B. **Software engineering with UML**. CRC Press, 2018.