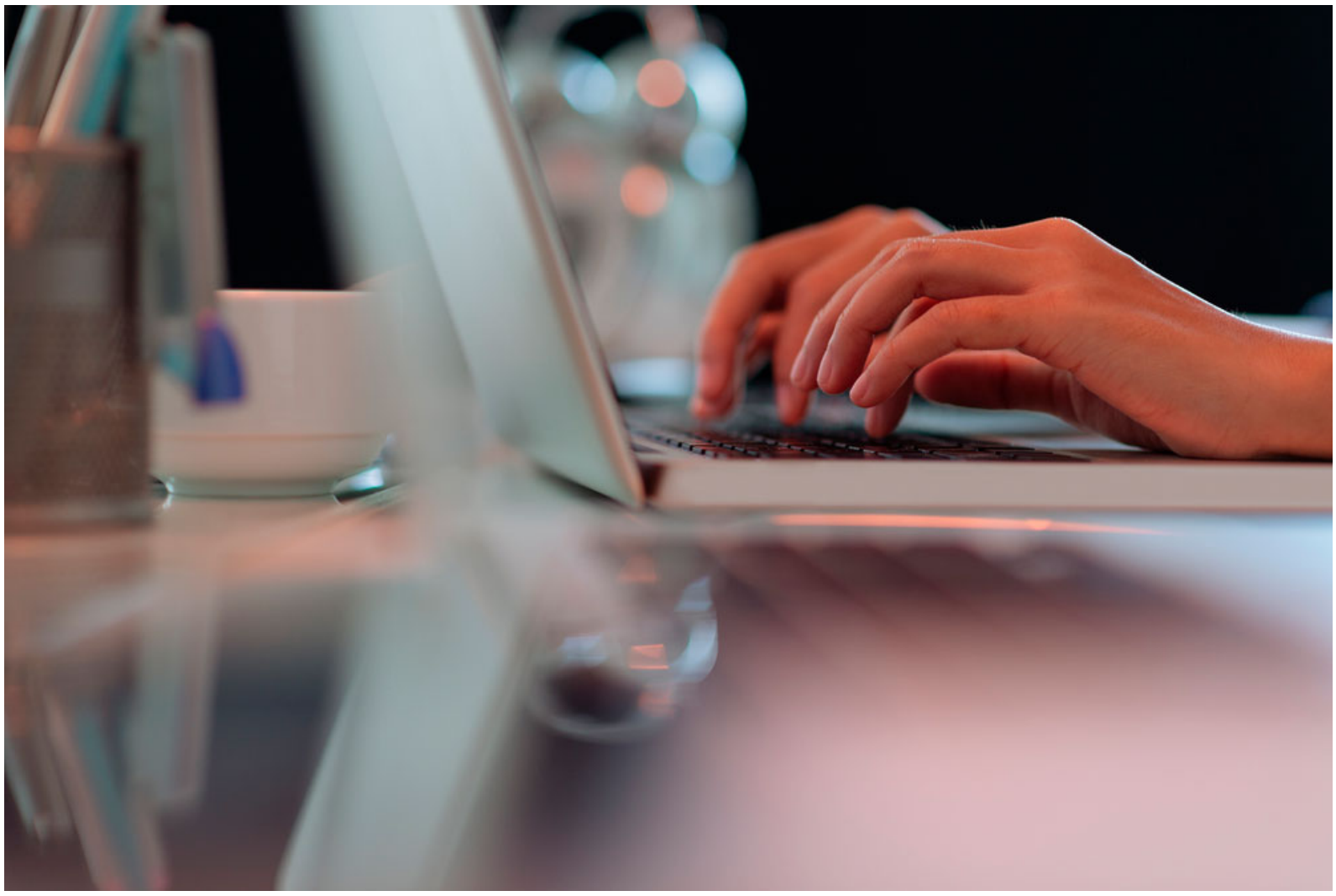
NÃO PODE FALTAR

FUNDAMENTOS DA UML

Maurício Acconcia Dias

A LINGUAGEM UML

A UML é uma linguagem de modelagem visual de soluções de problemas que torna a modelagem de software universal e de simples entendimento.



Fonte: Shutterstock.

Deseja ouvir este material?

Áudio disponível no material digital.

CONVITE AO ESTUDO

Diversas empresas atualmente, independentemente de seu tamanho, necessitam de serviços relacionados ao desenvolvimento de software. Como exemplo é possível citar os sistemas web de divulgação, cadastro de clientes, e-commerce, softwares de controle de estoque, gerenciamento de clientes e inteligência de negócios. Diversos tipos de clientes buscam empresas de desenvolvimento para solucionar seus problemas e nem sempre estes clientes têm o conhecimento necessário para saber o que desejam de verdade gerando um conjunto de requisitos de desenvolvimento complexo.

Ao fazer parte de uma equipe de desenvolvimento de software é necessário resolver diversos problemas para atender todos os requisitos solicitados. Antes de programar, normalmente, modelamos a solução de alguma forma como por meio de um pseudocódigo, seja um fluxograma ou, até mesmo, desenhos sem seguir um modelo padrão. Esta situação é aceitável quando estamos desenvolvendo um software em casa com objetivos acadêmicos ou de aprendizado, porém em uma empresa de desenvolvimento não será possível atender aos requisitos de forma satisfatória se não for utilizada alguma ferramenta de auxílio, já que os problemas são consideravelmente maiores e mais complexos. Além disso, em empresas, trabalhamos em meio a um time de desenvolvedores o que pode ser bem complexo, assim como você já deve ter percebido em seus trabalhos em grupo. Seria então interessante poder utilizar uma forma de modelar os problemas que todas as pessoas envolvidas em um projeto pudessem entender sem precisar de explicações.

Para entender bem como utilizar uma ferramenta é necessário um estudo de todas as suas principais características. O histórico da criação e desenvolvimento da ferramenta será capaz de mostrar o motivo de sua criação e evolução. Entender suas principais ferramentas permite que seja possível identificar quando e como utilizá-las da melhor forma possível.

Esta unidade irá apresentar a linguagem UML que é uma linguagem de modelagem visual de soluções de problemas que atinge o principal objetivo deste tipo de ferramenta: ela torna a modelagem de software uma linguagem universal e de simples entendimento.

A seção 1 apresenta o histórico da linguagem, quando foi criada, as motivações e justificativas. Mostra também como a linguagem evoluiu ao longo do tempo incorporando as necessidades dos desenvolvedores e melhorando seus pontos fracos. Em seguida, na seção 2, os diagramas da linguagem serão apresentados e divididos de acordo com sua aplicação, ou seja, diagramas estruturais (responsáveis por modelar a estrutura do sistema como classes, atributos e operações), comportamentais (que detalham o funcionamento das partes do sistema) e de interação (que modelam as interações entre objetos de uma aplicação). Depois dessa etapa, vamos entender como é fundamentada a linguagem UML. Por fim, na seção 3, será abordada a linguagem em um processo de desenvolvimento e seus mecanismos gerais.

Ao final desta unidade, toda a base para o correto entendimento da linguagem estará formada e será possível, então, evoluir para o estudo em profundidade sobre UML.

PRATICAR PARA APRENDER

Para o desenvolvimento de software para a solução de problemas, se não há um entendimento real do problema em questão, a solução dificilmente é satisfatória. Por muitos anos, logo no início da programação de sistemas de informação para empresas, o desenvolvimento de software era feito sem seguir um padrão, sem a utilização de técnicas ou ferramentas que até então eram inexistentes. Ao longo dos anos, com uma maior exigência e necessidade de resolução de problemas mais complexos, surgiu maior demanda por sistemas mais complexos. Todavia, não existiam grandes preocupações com a manutenção e muitos códigos sequer tinham documentação porque eram produzidos informalmente. Em algumas ocasiões, a etapa de modelagem do sistema não existia, inviabilizando a manutenção nos códigos já desenvolvidos.

Neste momento, você já percebeu que em problemas complexos, sempre que utilizamos a ferramenta correta, é possível obter sucesso considerando um caminho menos árduo. Esta seção irá apresentar o histórico e as principais características de uma das ferramentas mais utilizadas para análise e modelagem de software atualmente: a linguagem UML.

Esta linguagem utiliza conceitos de modelagem visual, tendo como principal objetivo uniformizar a maneira como os sistemas, processos, projetos e negócios são modelados e descritos, solucionando os diversos problemas normalmente encontrados em empresas de desenvolvimento de software, especialmente quando há divergência entre o projeto e a entrega final e, ainda, o retrabalho que pode ocorrer por conta de implementações erradas de módulos do sistema (UML-DIAGRAMS, 2016).

Em seguida são apresentados os princípios, as características e a evolução da UML para que seja possível compreender o contexto relacionado à linguagem e as etapas de desenvolvimento para que ela atingisse a maturidade e aplicabilidade atual. Por fim, a maneira como a UML é utilizada para a modelagem de sistemas encerra esta primeira seção criando a base para o estudo aprofundado dos diagramas existentes na linguagem.

Este início é fundamental, pois irá demonstrar a importância da linguagem, seu impacto na área de desenvolvimento de software, os motivos que levaram a sua adoção em larga escala no mercado de trabalho e os fatores que resultaram em sua evolução.

> **EXEMPLIFICANDO**
>
> Um exemplo de impacto pode ser um software de uma loja de vendas onde o gerente deve ter acesso ao relatório de cada vendedor para tomar decisões sobre a equipe. Se este requisito não for devidamente documentado pode ser que o time de desenvolvimento, quando for programar a geração de relatórios, não implemente uma maneira de obter o relatório de vendas. Quando o software estiver terminado e for entregue ao cliente, logo na primeira reunião, o gerente irá procurar pela funcionalidade e não irá encontrá-la, causando um transtorno para o cliente. Além disso, o software terá de retornar a fase de desenvolvimento para adicionar esta funcionalidade, o que levaria tempo dependendo de como o sistema foi construído. Todo o transtorno e o atraso neste caso poderiam ser evitados caso existisse uma documentação apropriada que descrevesse de forma clara as funcionalidades do sistema. A UML, por ser uma linguagem de modelagem visual, apresenta diagramas de fácil entendimento que de forma eficiente evitam este tipo de ocorrência.

Você, recém-formado e ingressante no mercado de trabalho, foi contratado por uma empresa de desenvolvimento de software. A experiência de iniciar na equipe de desenvolvimento de uma empresa nova é um desafio normalmente, pois cada empresa possui seus métodos de desenvolvimento, seus padrões e todos os times de desenvolvimento estão adaptados a este 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 a empresa para a qual vai trabalhar agora ainda utiliza métodos antigos de desenvolvimento e, em alguns projetos, método algum. Agora que você iniciou seus estudos de UML, já sabe que é necessário desenvolver a modelagem dos sistemas utilizando métodos e

linguagens específicas para esta finalidade. Sabe ainda que a linguagem mais utilizada é a UML por possuir todas as características principais desejáveis para o problema em questão.

Sendo assim, como seus conhecimentos sobre o histórico de UML, suas características e seus benefícios estão sólidos, você acredita poder melhorar o desenvolvimento de software no novo trabalho. Porém, esta tarefa nem sempre é fácil devido à inércia apresentada pelos funcionários que já trabalham na companhia. Você terá que, aos poucos, mostrar os conceitos, vantagens, desvantagens, motivações e justificativas para que a empresa adote a UML e se beneficie da ferramenta em seu processo de desenvolvimento.

Logo nos primeiros dias, você percebeu que existem problemas no processo de desenvolvimento de software e a maioria deles está associada ao fato de que os programadores nem sempre entendem realmente a análise e o projeto dos softwares, o que leva a uma implementação falha. Neste caso, você pediu um espaço para apresentar os conceitos de UML e desenvolvimento de software e conseguiu uma parte de uma reunião. Lembre-se de que a implantação de uma ferramenta nova em uma empresa, independente de qual seja, irá apresentar alguns problemas que ocorrem sempre e estes problemas também são importantes e devem ser considerados.

Apresente aqui como você iria abordar, nesta primeira reunião, o processo histórico de criação da UML, suas principais características e versões e como pode ser utilizada diretamente na modelagem dos sistemas. Descreva quais características você apresentaria a seus novos superiores para convencê-los de que ao fazer esta modificação no processo de desenvolvimento, os resultados obtidos serão melhores e o tempo gasto com seu desenvolvimento poderia ser inclusive diminuído. Faça isso em uma lista que contenha problemas de desenvolvimento, vantagens e desvantagens, se achar conveniente.

Atualmente no mercado de trabalho uma quantidade considerável de empresas utiliza UML para a modelagem de seus sistemas e você, como futuro analista de sistemas, precisa ter este conhecimento para obter êxito em sua jornada profissional. Preparado para aprender sobre UML? Bons estudos!

CONCEITO-CHAVE

ANÁLISE E MODELAGEM DE SOFTWARES

O desenvolvimento de software apresenta diversos desafios e um dos principais é se certificar de que todos os requisitos solicitados pelo cliente serão atendidos e da forma correta. Uma das formas de se obter sucesso com relação a este problema é realizando uma boa análise dos requisitos e a partir da análise elaborar modelos que representem o sistema em formato mais facilmente entendível por todos os membros do time de desenvolvimento. Portanto, antes de desenvolver o sistema e codificar, é importante que os membros do time tenham uma documentação clara com todos os detalhes estruturais e como o sistema ou partes do sistema deverão se comportar dependendo da ação do usuário. Vale destacar que a documentação com esses detalhes é de extrema importância para que o resultado final, o produto de software, não gere insatisfação do cliente, por não desenvolver alguma funcionalidade requerida pelo cliente, por exemplo, ou até gerar retrabalho na programação do software, atrasos no cronograma e custos adicionais que impactam no orçamento da empresa. Para evitar esses transtornos, muitas empresas adotam ferramentas – como a linguagem UML – que irão auxiliar no processo de desenvolvimento.

LINGUAGEM UML

A Linguagem de Modelagem Unificada – UML (do inglês, *Unified Modeling Language*) é uma das principais ferramentas utilizadas para modelagem de negócios e processos similares, análise, design e implementação de sistemas baseados em software. A UML possibilita realizar especificações dos artefatos a serem desenvolvidos, bem como definir as tarefas do grupo e/ou de algum membro do grupo responsável pelo desenvolvimento do sistema. Importa destacar que a UML é uma linguagem de modelagem universal, não um processo de desenvolvimento de software. A análise orientada a objetos está diretamente ligada a UML, uma vez que, na elaboração de modelos, o sistema deve ser descrito com elementos (objetos) que sugerem o que acontece mais concretamente no mundo real. Portanto, vamos entender melhor o surgimento desta relação para, em seguida, ver os conceitos iniciais sobre a linguagem.

PARADIGMA DE PROGRAMAÇÃO ORIENTADA A OBJETOS

O paradigma de orientação a objetos (OO) é um paradigma de programação importante que teve seu início oficial com uma linguagem chamada SIMULA (HOLMEVICK, 1994). Esta linguagem apresentou os conceitos que em seguida viriam a ser engiobados por linguagens como JAVA, C++, C#, PHP. Estes conceitos

são as classes, os objetos e as relações entre eles como herança, polimorfismo, agregação e composição. Em oposição ao paradigma de programação estruturada composto por variáveis, funções e tipos abstratos de dados, a orientação a objetos propôs um modelo novo de programação que permite a modelagem dos problemas a serem solucionados pelos programadores pensando em objetos do mundo real. Este fato impacta diretamente um dos passos importantes do desenvolvimento de software: a análise.

A fase de análise de requisitos é importante para que seja possível modelar e projetar o software corretamente de acordo com o que foi especificado pelo cliente. Este processo (análise) em modelos atuais de desenvolvimento de software é classificado como modelagem e sua definição é: o processo de desenvolvimento de modelos abstratos do sistema, sendo que cada um destes modelos apresenta uma perspectiva diferente do mesmo sistema (SOMMERVILLE, 2011). Considerando esta definição, que especifica a necessidade de diversos modelos abstratos do sistema, pode-se entender a importância de uma ferramenta de modelagem para auxiliar no processo. Esta ferramenta é a UML.

Seria interessante se desde o início do desenvolvimento utilizando OO os desenvolvedores percebessem a necessidade de se modelar o sistema detalhadamente com diferentes perspectivas do mesmo problema a ser resolvido, porém não foi assim que aconteceu o processo de desenvolvimento da linguagem UML.

LINGUAGEM ADA

Dentre as linguagens inicialmente desenvolvidas com o objetivo de permitir a programação de softwares orientados a objetos é possível destacar o caso da linguagem ADA. A linguagem ADA foi desenvolvida pelo Departamento de Defesa dos Estados Unidos e, desde o início, possuía compiladores extremamente rápidos e eficientes para as aplicações-alvo que eram militares em sua maioria (ICHBIAH *et al.*, 1996). O alto desempenho da linguagem ADA fez com que fosse amplamente adotada nos anos 1980 e um pesquisador chamado Grady Booch apresentou um dos primeiros trabalhos relacionados à análise orientada a objetos utilizando a linguagem ADA (BOOCH, 1986).

A partir destes trabalhos iniciais com foco na linguagem ADA, outros projetos também foram apresentados com métodos de análise e modelagem para linguagens orientadas a objetos. Estes métodos tinham duas características

principais que os tornavam não tão atraentes de uma forma geral: ou eram orientados a determinadas linguagens, ou eram orientados a modelos de dados e estruturas de dados específicas. Um exemplo neste sentido é o método OMT criado por James Rumbaugh *et al.* (1991) que apresentava uma solução de descrição de um modelo específico de banco de dados, o modelo entidade-relacionamento.

REFLITA

Neste momento é interessante pensar um pouco sobre a situação descrita em relação às soluções apresentadas para uma linguagem ou aplicação específica. Quando um problema é comum a diversas áreas de pesquisa e desenvolvimento é sempre interessante pensar em uma solução genérica que resolva o problema do maior número possível de pessoas. Considere o grande número de linguagens de programação existentes, cada uma com o seu escopo de aplicação específico, ou seja, umas melhores para desempenho como C++, outras melhores para interfaces com o usuário e portabilidade como JAVA, outras ainda específicas para aplicações WEB como o PHP. Todas as linguagens listadas são linguagens orientadas a objetos utilizados para diferentes aplicações e a questão é: por melhor que seja uma solução de modelagem aplicada a um tipo específico de linguagem ou problema, o quão importante esta solução é para a área de pesquisa? Não seria melhor desenvolver uma solução genérica? Reflita, então, neste momento, sobre os benefícios de se ter uma solução de desenvolvimento aplicável a qualquer linguagem orientada a objetos e tente listar estes benefícios.

PADRONIZAÇÃO

Um número considerável de linguagens e métodos voltados para orientação a objetos com os problemas já listados foi desenvolvido até que, em meados dos anos 1990, o grupo de padronização chamado OMG (do inglês, *Object Management Group*) (OMG, 2020) percebeu o problema e resolveu solucioná-lo. Em 1996 foi aberta uma chamada para um padrão unificado de modelagem pela OMG e isto casou com o que estava sendo feito por três pesquisadores da área, conhecidos como "*three amigos*", Grady Booch, Ivar Jacobson e James Rumbaugh. Eles já haviam apresentado, um ano antes, ideias em relação à padronização da

modelagem na conferência chamada OOPSLA (do inglês, *Object Oriented Systems Languages and Applications*). Neste momento, o objetivo deles era (BÉZIVIN, MULLER, 1999):

- Utilizar conceitos de OO para representar sistemas complexos.
- Estabelecer uma ligação entre a modelagem e a implementação.
- Considerar no modelo fatores de escala que são inerentes a sistemas complexos e críticos.
- Criar uma linguagem que poderia ser processada tanto pelos computadores como pelos programadores.

O resultado deste esforço foi a primeira versão da linguagem UML enviada como resposta à chamada do OMG em 1997. Após esta primeira versão, pessoas envolvidas em versões paralelas também enviadas ao OMG foram convidadas para participar do projeto da versão 1.1. Neste segundo momento, o principal objetivo foi a definição de um metamodelo, ou seja, a definição de quais elementos seriam disponibilizados pela linguagem UML e como utilizá-los. O padrão UML 2.0 lançado em 2005 foi desenvolvido a partir dos anos 2000 pela OMG em um processo de modernização. Apesar de todas as mudanças propostas, até hoje a linguagem possui as características iniciais de sua criação.

CARACTERÍSTICAS DA UML

A linguagem UML apresenta algumas características que a tornam uma linguagem que cumpre de maneira satisfatória o que é esperado para modelagem do sistema de software. Sendo assim, a seguir serão apresentadas algumas importantes características da linguagem em relação a seu processo de unificação (RUBAUGH, JACOBSON, BOOCH, 2004):

- UML combina os conceitos comuns de linguagens orientadas a objetos apresentando uma definição clara para cada um, bem como notação e terminologia. Isto faz com que seja possível representar a maioria dos modelos existentes utilizando UML.
- UML é compatível com o desenvolvimento de software desde os requisitos até as etapas finais do desenvolvimento. Os mesmos conceitos e notações podem ser utilizados em diferentes estágios sem necessidade de tradução dos modelos.
- UML é compatível com diversos escopos, ou seja, é capaz de modelar diferentes linguagens, bancos de dados, documentação organizacional, trabalha com diversos frameworks, engloba inclusive o desenvolvimento de software de controle de hardware (*firmwares*).
- Um dos resultados mais importantes do desenvolvimento da UML, que não era um de seus objetivos iniciais, foi um esforço para modelar os relacionamentos entre os conceitos das linguagens orientadas a objetos tornando os diagramas aplicáveis a várias situações que eram conhecidas para os desenvolvedores e outras que ainda não eram.

OBJETIVOS DA UML

Agora que entendemos os principais conceitos relacionados à linguagem UML é possível apresentar seus objetivos como uma ferramenta de modelagem que já foram atingidos em seu desenvolvimento.

- O primeiro, e mais importante, é ser geral no sentido de modelar diferentes linguagens e situações. A questão de ser uma ferramenta não proprietária, resultado de um acordo realizado com grande parte da comunidade de desenvolvedores, permite que seja utilizada por todos que desejarem modelar seus softwares.
- Outro importante objetivo foi a superação de outros modelos já existentes na época de seu lançamento para modelagem, permitindo que se tornasse realmente um padrão para o desenvolvimento de software.
- O último, porém igualmente importante objetivo, foi ser tão simples quanto possível sem perder a capacidade de modelagem de sistemas complexos.

VERSÕES DA UML

Em relação às versões desenvolvidas da linguagem UML é interessante analisar sua evolução para entender como a linguagem foi desenvolvida (BOOCH, RUBAUGH, JACOBSON, 2005).

- A primeira versão, que foi enviada ao OMG em janeiro de 1997, não estava totalmente completa e precisou de reformulações. A primeira versão realmente lançada como funcional ao mercado foi a 1.1 em novembro de 1997.
- Após o retorno das informações sobre os principais problemas da linguagem, pequenas modificações foram feitas com relação a semântica, notações e metamodelos gerando a versão 1.3.
- Um contínuo processo de desenvolvimento que acrescentou aos diagramas opções de visibilidade, artefatos e estereótipos resultou na versão 1.4, que foi seguida pela versão 1.5 com a adição de procedimentos e mecanismos de *data flow*. Em janeiro de 2003, entre os lançamentos das versões 1.4 e 1.5, a UML foi aceita pela ISO como um padrão.
- Em julho de 2005 foi lançada a UML 2.0 com a inclusão de interações nos diagramas como *object*, *package* e *timing* (AMBLER, 2005). Os diagramas de colaboração foram renomeados para diagramas de comunicação, dentre outras modificações significativas em todos os outros diagramas existentes na linguagem.
- Entre as versões 2.1 a 2.3 apenas correções de erros e pequenas modificações foram realizadas.
- A versão 2.4.1 de julho de 2011 incorporou mudanças em classes, pacotes e estereótipos.
- A versão da linguagem – UML 2.5 – é uma revisão da versão anterior com melhorias na questão da simplicidade da linguagem, com a geração mais rápida de modelos mais eficientes e descartando características de versões anteriores não mais utilizadas.
- Em sua última versão, 2.5.1 de 2017, houve a correção de erros reportados pelos usuários em relação aos diagramas e sua utilização, porém, segundo o controle apresentado no site da OMG, alguns problemas ainda não foram resolvidos.

MODELOS

A linguagem UML é toda baseada em modelos, portanto é necessário saber o que é um modelo, seus propósitos, o que compõe um modelo e, principalmente, seu significado. Um modelo pode ser definido, de forma simples e direta, como uma

representação de algo de alguma natureza (software, problema, sistema matemático) utilizando algo da mesma ou de outra natureza.

EXEMPLIFICANDO

Vamos explicar com mais detalhes o conceito do que é um modelo e sua natureza. Um sistema mecânico, como um motor de carro, pode ser representado por um modelo matemático de natureza diferente. Outro exemplo possível é a simulação de um sistema elétrico em um software de simulação e, neste caso, ao criar um modelo computacional de um sistema elétrico real, estamos trabalhando com objetos de natureza diferente. Um software, programa de computador, pode ser representado por um diagrama UML, que é uma ferramenta visual. Os dois objetos em questão, o software e o modelo UML do software, possuem a mesma natureza computacional, ou seja, tanto o software quanto o modelo são criados e definidos em um computador.

Um modelo captura aspectos importantes e de alguma forma modifica ou omite o restante das informações. A forma como o modelo é apresentado e desenvolvido deve ser escolhida para facilitar tanto sua construção quanto sua interpretação e utilização. O modelo de softwares e sistemas computacionais é normalmente feito em uma linguagem de modelagem e, atualmente em sua maioria, utilizando UML (GUEDES, 2018).

Os propósitos em um desenvolvimento de modelos são vários, porém é possível destacar alguns deles:

- Capturar e definir com precisão os requisitos do software para realmente atender às necessidades de quem contratou o desenvolvimento do software.
- Auxiliar o início do projeto do sistema no sentido de apresentar melhor as características que estariam condensadas em um texto.
- Apresentar uma solução que contenha as decisões de projeto em uma forma que não depende diretamente dos requisitos.
- Melhorar a exploração de diferentes soluções com uma apresentação simples de ser feita e de fácil entendimento.
- E, principalmente, permitir que um sistema complexo seja descrito de forma que possa ser entendido em sua totalidade, possibilitando seu desenvolvimento de forma eficiente.

Modelos podem ser apresentados de várias formas diferentes com diversos objetivos e níveis de abstração. A quantidade de detalhes em um modelo deve ser considerada de acordo com seu objetivo de construção.

O Quadro 1.1 sintetiza os níveis de abstração e a função do diagrama para cada um desses níveis de abstração.

Quadro 1.1 | Resumo dos tipos de abstração e seus diagramas correspondentes

Nível de Abstração	**Objetivo do Diagrama**
ALTO	Ser claro e simples, apresentar os conceitos ao cliente para tomada de decisão
MÉDIO	Guiar o desenvolvimento apresentado, sem detalhar demais, as classes, os objetos e as interações
BAIXO	Demonstrar como deve ser desenvolvido o sistema propriamente dito. Necessita de diagramas e modelos com a especificação completa de cada módulo, interação e outras informações que possam ser necessárias

Fonte: elaborado pelo autor.

Em um nível alto de abstração, a ideia é apresentar diagramas que são construídos no início do desenvolvimento para mostrar aos clientes as possibilidades e auxiliar na tomada de decisão em pontos que precisam de aprovação. Após este nível é necessário detalhar um pouco mais com a abstração da estrutura essencial do sistema, apresentando sem muitos detalhes como serão desenvolvidos os objetos, as classes e como irão interagir. Em seguida, o nível de abstração “desce” ainda mais com o desenvolvimento de diagramas e modelos com a especificação completa de cada módulo, interação e tudo mais que seja necessário para que realmente seja possível construir o sistema propriamente dito. A única maneira de melhorar ainda mais este cenário seria apresentando exemplos possíveis do sistema com certa funcionalidade (protótipos) para que a funcionalidade seja demonstrada de forma completa (GUEDES, 2018).

> **ASSIMILE**
>
> Neste momento é importante entender como a utilização da linguagem UML faz diferença no processo de desenvolvimento de software. A modelagem tem um impacto muito grande no resultado final do software desenvolvido e, portanto, é necessário que seja feita de forma que todos os requisitos sejam atendidos e que todo o time de desenvolvimento do software seja capaz de visualizar os detalhes necessários para o

desenvolvimento. Ao utilizar a linguagem UML e seus diagramas, o resultado da modelagem é visual e pode ser entendido sem problemas por todos os envolvidos em seu desenvolvimento. Esta característica irá resultar em um software com menor número de erros no desenvolvimento e na versão final do software, diminuição no tempo gasto com testes (justamente pelo número de erros ser menor) e com menor probabilidade de atrasos em seu tempo de desenvolvimento.

FLUXO DE DESENVOLVIMENTO

Existem diversos fluxos de desenvolvimento de software, tanto clássicos (cascata, espiral, prototipação) como os atuais (chamados de metodologias ágeis de desenvolvimento) que possuem uma etapa de modelagem prevista. A linguagem UML pode ser utilizada em qualquer um deles por ser **uma linguagem de modelagem independente do fluxo de desenvolvimento de software escolhido** (UML-DIAGRAMS, 2016).

Alguns componentes principais devem estar presentes no modelo e dentre eles é possível destacar:

- A **apresentação visual da semântica do sistema**, ou seja, uma forma de visualizar tudo que o sistema precisa ter e as relações entre as partes do sistema de forma gráfica e simples.
- E o **contexto**, ou seja, o padrão de apresentação para o público no qual o modelo foi desenvolvido.

Informações importantes para o sistema, porém, que estão fora do padrão adotado pela organização em questão, para o público-alvo esperado, devem ser armazenadas em outros locais e apresentadas de outras formas. Um exemplo de quebra de contexto é um diagrama de alto nível de abstração que contém vários trechos de código em uma linguagem específica e é apresentado aos executivos que contrataram o desenvolvimento. Se a empresa desenvolve softwares para diferentes segmentos de mercado, os clientes não irão tirar proveito algum desta informação na maioria dos casos.

Por fim é importante lembrar do real significado de um modelo, que é ser um gerador de potenciais configurações para o sistema em questão em seus diferentes níveis. O fato de se ter diversos diagramas possíveis em UML é para agrupar características diferentes nos modelos certos e só apresentar o que interessa para quem interessa. Isso inclui a escolha de modelos que apresentam o que o sistema faz em detrimento a modelos que apresentam como o sistema realiza determinadas ações, sendo que cada um destes diagramas possui públicos específicos diferentes entre si. Por isso é importante conhecer bem a ferramenta de modelagem e utilizar todo seu potencial para obter o melhor resultado possível no momento do desenvolvimento.

Apesar de todas as facilidades da linguagem UML e do esforço de seus desenvolvedores em torná-la simples, algumas questões são utilizadas como justificativa para a não utilização da ferramenta, como a falsa impressão de que a linguagem é complexa. Isso pode se dar por, basicamente, dois motivos: pouco conhecimento sobre sua utilização e o costume de utilização de um método específico pelo desenvolvedor, que acredita ser esse método mais eficiente que a UML.

Em relação a esses problemas é importante considerar que a UML é uma ferramenta completa, amplamente utilizada e que foi desenvolvida por um grupo de pessoas que são especialistas na área de desenvolvimento para ser simples e eficiente.

Sendo assim é necessário que se aprenda a utilizar a UML e que seu uso se dê em diversos projetos antes de se decidir por outra ferramenta ou nenhuma. Na maioria dos casos, estes “problemas” acabam se mostrando como apenas uma resistência sem fundamento por parte dos desenvolvedores.

Neste ponto você já foi apresentado ao contexto da linguagem UML, sua evolução e algumas de suas principais características. Está familiarizado com diversos conceitos relacionados a modelos e abstração de dados e está preparado para iniciar seu estudo específico dos modelos presentes na linguagem UML. Bons estudos!

REFERÊNCIAS

AMBLER, S. W. **The Elements of UML 2.0 Style**. Cambridge University Press. 2005.

BÉZIVIN, J.; MULLER, P. A. UML: The Birth and Rise of a Standard Modeling Notation. The Unified Modeling Language. **UML’98**: Beyond the Notation, v.1618 of Lecture Notes in Computer Science, pp. 1-8. Springer, 1999.

BOOCH, G. Object-Oriented Development. **IEEE Transactions on Software Engineering**, n. 12, v. 2, pp. 211–221, 1986.

BOOCH, G.; RUBAUGH J.; JACOBSON, I. **The Unified Modeling Language User Guide.** 2. ed. (Addison-Wesley Object Technology Series). Addison-Wesley Professional. 2005.

GUEDES, G. T. A. **Uml 2** – Uma Abordagem Prática. 3. ed. São Paulo: Novatec, 2018.

HOLMEVICK, J.R. Compiling SIMULA: A Historical Study of Technological Genesis. **IEEE Annals of the History of Computing**, n. 16, v. 4. pp. 25–37, 1994.

ICHBIAH, J.D. *et al.* **Rationale for the Design of the Ada Programming Language**. Cambridge: Cambridge University Press, 1986.

OMG. **OMG® Unified Modeling Language® (OMG UML®) version 2.5.1**. 2017. Disponível em: https://bit.ly/30WhGSJ. Acesso em: 19 maio 2020.

OMG. **Object management group page**. 2020. Disponível em: https://bit.ly/3360Aof. Acesso em: 19 maio 2020.

O QUE é o software livre? Free Software Foundation. [S.l. s.n.], 3 jul. 2019. 03/07/2019. Disponível em: https://bit.ly/3k1e62u. Acesso em: 11 maio 2020.

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

RUMBAUGH J. *et al.* **Object-Oriented Modeling and Design**. New Jersey: Prentice Hall, 1991.

SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo: Pearson Universidades, 2011.

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

VISÃO Geral do Sistema GNU. Free Software Foundation, Inc. 29/04/2019. Disponível em: https://bit.ly/309nGbt. Acesso em: 1 jul. 2020.