UML

UML é uma notação gráfica para modelagem de software. A linguagem define um conjunto de diagramas para documentar e ajudar no design de sistemas de software, particularmente sistemas orientados a objetos. As origens de UML datam da década de 80, quando o paradigma de orientação a objetos estava amadurecendo e vivendo seu auge. Assim, surgiram diversas linguagens orientadas a objetos, como C++ e também algumas notações gráficas para modelagem de software. Lembre-se que os sistemas na década de 80 eram desenvolvidos segundo o Modelo Waterfall, que prescreve uma grande e longa fase de design. A proposta de UML era que nessa fase seriam criados modelos gráficos, que depois seriam repassados para os programadores, para serem convertidos em código fonte.

Na verdade, UML é o resultado de um esforço para unificar as notações gráficas que surgiram no final das décadas de 80 e início da década de 90. Especificamente, a primeira versão de UML foi proposta em 1995, como resultado da unificação de notações que estavam sendo desenvolvidas de forma independente por três Engenheiros de Software conhecidos na época pelos três amigos : Grady Booch, Jim Rumbaugh e Ivar Jacobson. Nessa época, surgiram também ferramentas para desenhar diagramas UML, as quais foram chamadas de ferramentas CASE (Computer-Aided Software Engineering). O nome é inspirado em ferramentas CAD (Computer Aided Design), usadas para criar modelos para produtos de Engenharia tradicional, como casas, pontes, automóveis e aviões. Por isso, era importante ter uma padronização de UML, de forma que um diagrama criado em uma ferramenta CASE pudesse ser aberto e editado em uma outra ferramenta, de uma empresa diferente. De fato, em 1997, UML passou a ser um padrão gerenciado pela OMG, que é uma organização de padronização financiada por indústrias de software. Desde o início, o desenvolvimento de UML foi comandado por consultores influentes e por grandes empresas de ferramentas ou consultoria, como a Rational, que depois viria a ser comprada pela IBM.

Como usar UML?

Martin Fowler, em seu livro sobre UML, propõe uma classificação sobre formas de uso dessa linguagem de modelagem. Segundo ele, existem três formas de uso de UML: como blueprint, como linguagem de programação ou como esboço. Vamos descrever cada uma delas nos próximos parágrafos.

UML como blueprint corresponde ao uso de UML vislumbrado por seus criadores, ainda na década de 90. Nessa forma de uso, defende-se que, após o levantamento de requisitos, seja produzido um conjunto de modelos — ou plantas técnicas (blueprints) — documentando diversos aspectos de um sistema e sempre usando diagramas UML. Esses modelos seriam criados por analistas de sistemas, usando-se ferramentas CASE e, depois, repassados a programadores para codificação. Logo, UML como blueprint é recomendado quando se emprega processos de desenvolvimento do tipo Waterfall ou quando se adota o Processo Unificado (UP). Na verdade, UP foi proposto por pessoas com forte ligação com UML. No entanto, como já discutimos no Capítulo 2, o uso de UML na construção de modelos detalhados e completos é cada vez mais raro. Por exemplo, com métodos ágeis não existe uma longa fase inicial de design (big design up front). Em vez disso, decisões de design são tomadas e refinadas ao longo do

desenvolvimento, em cada uma das iterações (ou sprints). Por isso, não iremos neste capítulo nos aprofundar no uso de UML como blueprint.

UML como linguagem de programação corresponde ao uso de UML vislumbrado pela OMG, após a padronização da linguagem de modelagem. De forma ambiciosa e pelo menos durante um período, vislumbrou-se a geração de código automaticamente a partir de modelos UML. Em outras palavras, não haveria mais uma fase de codificação, pois o código seria gerado diretamente a partir da compilação de modelos UML. Essa forma de uso é conhecida como Desenvolvimento Dirigido por Modelos (Model Driven Development ou MDD). Para que MDD fosse viável, UML foi expandida e ganhou novos recursos e diagramas. Foi a partir desse momento que a linguagem ganhou a reputação de ser pesada e complexa. Porém, mesmo com adição de complexidade extra, o uso de UML para geração de código não se tornou comum, pelo menos na grande maioria dos sistemas.

Resta então o terceiro uso, **UML como esboço**, que corresponde à forma que vamos estudar neste capítulo. Nela, usamos UML para construir diagramas leves e informais de partes de um sistema, vindo daí o nome esboço (sketch). Esses diagramas são usados para comunicação entre os desenvolvedores, em duas situações principais:

Engenharia Avante (Forward Engineering): quando os desenvolvedores usam modelos UML para discutir e analisar alternativas de design, antes que exista qualquer código. Por exemplo, suponha que uma história tenha sido alocada para o sprint corrente. Antes de implementar a história, os desenvolvedores podem se reunir e fazer um esboço das principais classes que deverão ser criadas no sistema, bem como dos relacionamentos entre elas. O objetivo é validar a proposta de tais classes antes de começar a codificar.

Engenharia Reversa (Reverse Engineering): quando os desenvolvedores usam modelos UML para analisar e discutir uma funcionalidade que já se encontra implementada no código fonte. Por exemplo, um desenvolvedor mais experiente pode desenhar alguns diagramas UML para explicar para um desenvolvedor recém-contratado como uma funcionalidade está implementada. Normalmente, é mais fácil conduzir essa explicação usando modelos e diagramas gráficos do que analisar e explicar cada linha de código. Ou seja, aplica-se aqui o ditado segundo o qual uma figura vale mais do que mil palavras.

Nas duas situações, o objetivo não é gerar modelos completos e detalhados. Por isso, não se considera o uso de ferramentas complexas e caras, como ferramentas CASE. Muito menos se cogita a geração automática de código a partir desses esboços. Muitas vezes, os diagramas são desenhados em um quadro e, depois, fotografados e apagados. Adicionalmente, usa-se apenas um subconjunto dos diagramas UML.

Como os esboços são pequenos e informais, pode-se questionar a necessidade de uma linguagem padronizada nos cenários que mencionamos. No entanto, consideramos que é melhor usar uma notação existente há anos, mesmo que de forma parcial, do que inventar uma notação própria. Especificamente, o emprego de UML como esboço contribui para evitar dois extremos. Por um lado, ele não assume o emprego rígido, detalhado e sistemático de UML. Por outro lado, evita-se o uso de uma notação informal e ad hoc, cuja semântica pode não ser clara para todos os desenvolvedores. Além disso, UML costuma ser usada em livros, tutoriais e documentos que explicam o uso de frameworks ou técnicas de programação.

Sintetizando a descrição que acabamos de fazer, modelos de software, como diagramas UML, são usados para comunicação entre desenvolvedores. Ou seja, eles são escritos por e para desenvolvedores. Trata-se de uma diferença importante para documentos de requisitos, que, conforme vimos no capítulo anterior, são escritos por desenvolvedores, mas de forma que eles possam ser lidos e verificados pelos usuários finais do sistema.

Mundo Real: No segundo semestre de 2013, Sebastian Baltes e Stephan Diehl — ambos pesquisadores da Universidade de Trier, na Alemanha — pediram a 394 desenvolvedores para responder um questionário sobre o emprego de esboços (sketches) em atividades de projeto de software (link). Esses desenvolvedores estavam distribuídos por mais de 32 países, embora a maioria fosse da Alemanha (54%). A análise das respostas obtidas revelou resultados interessantes sobre o uso de esboços em atividades de projeto e desenvolvimento de software, conforme descrito a seguir:

- 24% dos desenvolvedores que participaram da pesquisa criaram o último esboço no mesmo dia em que responderam ao questionário e 39% no intervalo de tempo máximo de uma semana, antes da resposta. Portanto, esses percentuais indicam que esboços são criados com frequência por desenvolvedores de software.

- 58% dos últimos esboços criados pelos participantes foram depois arquivados, seja em papel (6%), digitalmente (42%) ou de ambas as formas (10%). Isso sugere que os desenvolvedores consideram que os esboços carregam informação importante, que talvez seja útil no futuro.

- 40% dos esboços foram feitos em papel, 18% em quadros e 39% em computadores.

- 52% dos esboços foram feitos para ajudar no projeto (design) da arquitetura do sistema, 48% para ajudar no projeto de novas funcionalidades, 46% para explicar alguma tarefa para um outro desenvolvedor, 45% para analisar requisitos e 44% para ajudar no entendimento de uma tarefa. A soma dos percentuais ultrapassa 100% porque os participantes podiam marcar mais de uma resposta.

- 48% dos esboços continham algum elemento de UML e 9% eram integralmente baseados em diagramas UML. Portanto, esses percentuais reforçam a importância de estudar UML, não como notação para documentação detalhada de sistemas (blueprints), mas para ajudar na construção de modelos informais e parciais.

Diagramas UML

Os diagramas UML são classificados em dois grandes grupos:

Diagramas Estáticos (ou Estruturais) modelam a estrutura e organização de um sistema, incluindo informações sobre classes, atributos, métodos, pacotes, etc. Neste capítulo, vamos estudar dois diagramas estáticos: Diagramas de Classes e Diagramas de Pacotes.

Diagramas Dinâmicos (ou Comportamentais) modelam eventos que ocorrem durante a execução de um sistema. Por exemplo, eles podem modelar uma sequência de chamadas de métodos.

Para entender melhor a diferença entre esses grupos de diagramas, diagramas estáticos lidam apenas com informações que estão disponíveis, por exemplo, quando da compilação do código resultante dos modelos. Essa visão é estática porque ela não muda, a não ser que sejam realizadas mudanças nos modelos. Já os diagramas dinâmicos fornecem uma visão de tempo de execução. Eles são dinâmicos porque é comum ter execuções diferentes de um mesmo programa. Por exemplo, os usuários podem executar o programa com entradas diferentes, selecionar opções e menus diferentes, etc. Em resumo, se estiver interessado em modelar a estrutura de um programa, você deve usar diagramas estáticos. Se seu interesse for modelar o comportamento de um programa — isto é, o que pode acontecer durante sua execução, quais métodos são de fato executados, etc. — você deve usar algum diagrama dinâmico da UML.

Aviso: Existem diversas versões de UML. Vamos usar a versão de UML que é adotada na 3a edição do livro UML Distilled, de Martin Fowler. Esse livro foi o primeiro trabalho a discutir o uso de UML como esboço (sketches). Na verdade, vamos estudar um pequeno subconjunto da versão 2.0. Para se ter uma ideia do nível de detalhe alcançado por UML, a especificação da versão mais recente da linguagem — versão 2.5.1, no momento da escrita deste capítulo — possui 796 páginas. Ela pode ser encontrada no site da OMG.

FONTE: Capítulo 4 do LIVRO Engenharia de Software Moderna de Marco Tulio Valente, 2020.