**Resumo**

Para preencher o vazio entre os requisitos e o código fonte, os Engenheiros de Software investiram na criação de modelos de software. Esses modelos são mais detalhados que os requisitos e menos complexos que o código do sistema. Modelos são usados também em outras engenharias, entretanto modelos de software são considerados menos eficazes já que o motivo é que ao abstrair detalhes eles também descartam as complexidades que são essenciais para os sistemas modelados.

Ao refletir sobre o uso prático dos modelos, George Box diz que “todos os modelos são errados pois são simplificações ou aproximações da realidade. ”. Desse modo, o objetivo principal é avaliar se, apesar das simplificações, um modelo continua sendo útil para estudo de propriedades ou algo que ele modela. Ademais, modelos de software são representaçẽos gráficas de um sistema de software.

No âmbito de desenvolvimento de software, o desenvolvimento de modelos tem suma importância na fase de projeto/design. Inicialmente, no levantamento de requisitos, a atenção é voltada para a definição de problemas a serem resolvidos. No próximo passo, nas atividades de projeto, o problema já deve estar devidamente entendido e as atenções se voltam para a concepção de uma solução capaz de resolvê-lo. Após isso, a solução deve ser desenvolvida usando linguagens de programação, frameworks, etc.

UML é uma linguagem que se originou por volta da década de 80 e define um conjunto de diagramas para auxiliar no desenvolvimento dos sistemas orientados a objeto, na década de 80 eram desenvolvidos seguindo o Modelo Waterfall, que percorre uma longa fase de design. A proposta de UML era que nessa fase fossem criados modelos gráficos, que depois seriam repassados para os programadores, para serem convertidos em código fonte.

A primeira versão de UML foi proposta em 1995, por Grady Booch, Jim Rumbaugh e Ivar Jacobson - três Engenheiros de Software. Na mesma época surgiram algumas ferramentas para desenhar esses diagramas, chamadas ferramentas CASE (*Computer-Aided Software Engineering*). Em 1997, UML passou a ser um padrão em desenvolvimento de software gerenciado pela OMG, uma organização de padronização financiada por indústrias de software.

Em seu livro UML Distilled: A Brief Guide to the Standard Object Modeling Language, Martin Fowler, classifica UML em três formas de uso: UML como blueprint, UML como linguagem de programação e UML como esboço.

UML como blueprint trata-se de um uso propagado por seus criadores na década de 90. 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.

UML como linguagem de programação trata-se do uso de UML propagado pela OMG. Nesse tópico 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 Model Driven Development ou MDD.

UML como esboço trata-se do uso de UML para construir diagramas leves e informais de partes de um sistema. Esses diagramas são usados em 2 situações principais:

* **Engenharia Avante** (*Forward Engineering*): quando os desenvolvedores usam modelos UML para discutir e analisar alternativas de design, antes que exista qualquer código.
* **Engenharia Reversa** (*Reverse Engineering*): quando os desenvolvedores usam modelos UML para analisar e discutir uma funcionalidade que já se encontra implementada no código fonte.

Diagramas de UML são classificados em 2 tópicos:

* Diagramas Estáticos, que modelam a organização e estrutura do sistema. Isso inclui classes, pacotes e métodos.
* Diagramas Dinâmicos, que modelam os eventos que ocorrem durante a execução do sistema.

Os diagramas UML mais usados atualmente são os diagramas de classes. Eles representam graficamente um conjunto de classes e informações como seus atributos, métodos e relacionamentos. Esses diagramas são desenhados utilizando retângulos, para representar as classes, e setas, para os relacionamentos. Os retângulos são divididos em 3 partes: nome da classe, atributos e métodos. As setas são usadas para representar 3 tipos de relacionamentos: associação, herança e dependência.

Quando uma classe A possui um atributo b de um tipo B, dizemos que existe uma associação de A para B, a qual é representada por meio de uma seta, também de A para B. Relações de herança são representadas por meio de setas com a extremidade não preenchida. Essas setas são usadas para conectar subclasses à sua classe base. Diz-se que existe uma dependência de uma classe A para uma classe B, representada por uma seta com uma linha tracejada de A para B, quando a classe A usa a classe B, porém esse uso não ocorre por meio de associação ou herança.