O que é UML?

A **Unified Modeling Language**, ou **Linguagem Unificada de Modelagem**, é um modelo de linguagem para modelagem de dados orientado a objetos, usada para especificar, construir, visualizar e documentar um sistema de software.

Com ela podemos fazer uma modelagem visual de maneira que os relacionamentos entre os componentes do sistema sejam melhor visualizados e compreendidos e documentados .
Tecnicamente dizendo, a UML (Unified Modeling Language) é a junção das três mais conceituadas linguagens de modelagem orientados a objetos (Booch de Grady, OOSE de Jacobson e o OMT de Rumbaugh).

Como seu próprio nome indica, ela é uma linguagem de notação utilizada para modelar e documentar as diversas fases do desenvolvimento de sistemas orientados a objetos. Não é linguagem de programação.

Para isso, ela define uma série de elementos gráficos — como retângulos, setas, balões e linhas — que são usados em diferentes diagramas para representar os componentes de uma aplicação, suas interações e mudanças de estados.

Trata-se de uma linguagem de modelagem única, cujo papel é auxiliar a equipe de desenvolvimento a visualizar os diversos aspectos da aplicação, facilitando a compreensão do seu funcionamento.

Histórico da UML

As linguagens de modelagem orientadas a objetos surgiram entre a metade da década de 1970 e o final da década de 1980, à medida que o pessoal envolvido com metodologia, diante de um novo gênero de linguagens de programação orientadas a objeto e de aplicações cada vez mais complexas, começou a experimentar métodos alternativos de análise e projeto. A quantidade de métodos orientados a objetos aumentou de pouco mais de 10 para mais de 50 durante o período de 1989 a 1994. Muitos usuários desses métodos tiveram dificuldades para encontrar uma linguagem de modelagem capaz de atender inteiramente às suas necessidades.

Destacaram-se algumas linguagens como o Booch, o OOSE (Object-Oriented Software Engineering) de Jacobson, e o OMT (Object Modeling Technique) de Rumbaugh. Podemos citar outros métodos importantes como Fusion, Shlaer-Mellor e Coad-Yourdon. Todos eram métodos completos, alguns se destacavam em algum ponto, porém tinham suas limitações. O método Booch destacava-se durante as fases de projeto e construção de sistemas, o OOSE fornecia excelente suporte para captura de requisitos, a análise e o projeto em alto nível; o OMT-2 era mais útil com a análise e sistemas de informações com uso de dados[Booch, 2000].

Na metade da década de 1990, Grady Booch (Rational Software Corporation), Ivar Jacobson (Objectory) e James Rumbaugh (General Electrics) criadores de métodos orientados a objetos, começaram a pegar as melhores idéias e partiram para a criação de uma linguagem unificada de modelagem. Com isso esperavam fornecer ao mercado uma linguagem mais concreta e madura com os quais os desenvolvedores de ferramentas pudessem criar uma ferramenta mais utilizável. Usando técnicas orientadas a objeto criaram uma linguagem que iria desde o conceito até o sistema executável, não somente a sistemas complexos mas também a sistemas menores e também a outros problemas que não fossem sistemas de informação, podendo ser utilizado por seres humanos e máquinas[Furlan, 1998].

A criação da UML iniciou oficialmente em outubro de 1994, quando Rumbaugh se juntou a Booch na Rational. O foco inicial do projeto era a unificação dos métodos Booch e OMT[Furlan, 1998]. O esboço da versão 0.8 do Método Unificado foi lançado em outubro de 1995. Mais ou menos na mesma época Jacobson se associou à Rational com a finalidade de incorporar o OOSE no escopo inicial da versão 0.8, resultando o lançamento da versão

0.9 da UML em junho de 1996[Booch, 2000]. Foi então aprovada pela comunidade de engenharia de software em geral. Muitas empresas ficaram interessadas, foi então criada um consórcio com várias empresas interessadas em dedicar recursos com o propósito de trabalhar uma definição mais forte e completa da UML.
Empresas que contribuíram para a definição da UML 1.0, Digital Equipment Corporationm Hewlett-Packard, I-Logix, Intel-licorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational, Texas Instruments e Unisys. Resultando uma linguagem de modelagem bem definida , expressiva, poderosa, e que poderia ser aplicada a uma grande variedade de tipos de problemas[Booch, 2000]. A UML foi oferecida para a OMG (Object Management Group) em janeiro de 1997, em resposta à solicitação do próprio OMG de propostas para uma linguagem padrão de modelagem[Furlan, 1998].
Entre janeiro a julho de 1997, o grupo original se expandiu, passando a incluir virtualmente todos os participantes e colaboradores da resposta inicial ao OMG, entre os quais se encontravam Andersen Consulting, Ericson, Object Time Limited, Platinum Technology, Ptech, Reich Technologies, Softeam, Sterling Software e Taskon. Um grupo foi formado, liberado por Cris Kobryn da MCI Systemhouse e administrado por Ed Eykholt da Rational, com o propósito de formalizar a especificação da UML e de integrar a linguagem a outros esforços de padronização.

A versão 1.1 foi entregue a OMG em julho de 1997. Em setembro do mesmo ano, essa versão foi aceita pela ADTF (Analysis and Design Task Force) e pelo Architecture Board do OMG e, posteriormente submetida a votação de todos os membros da OMG. A versão 1.1 foi adotada pela OMG em 14 de novembro de 1997[Booch, 2000].
A manutenção da UML foi então assumida pela RTF (Revision Task Force) do OMG, sob a responsabilidade de Cris Kobryn. A RTF lançou uma revisão editorial, a UML 1.2., em junho de 1998. No final do mesmo ano, a RTF lançou a UML 1.3[Furlan, 1998].

Aceitação

Os criadores da UML procuraram desenvolver uma linguagem unificada padrão que pudesse ser de fácil entendimento a todos. Preocuparam- se em deixá-la aberta aos desenvolvedores, onde os mesmos pudessem criar seu próprio método de trabalho. Empresas desenvolvedoras de ferramentas estão livres para criarem uma ferramenta adequada ao uso da UML. Devido a necessidade de criação da UML

empresas e profissionais liberais da área estão desenvolvendo estudos para melhor aplicá-la.

Padronização OMG

Quando se iniciaram os trabalhos para criação da UML, os criadores tinham como intenção fazer sua aceitação com a distribuição da linguagem a vários desenvolvedores.

A OMG (Object Management Group) fez um requerimento por uma linguagem de modelagem padrão. Então houve interesse dos criadores da UML em padronizá-la, para isso foi preciso que os mesmos aprimorassem a qualidade da linguagem para tal. Pois para serem realmente utilizadas por empresas era necessário sua padronização.

Para que servem os diagramas UML?

Você já parou para pensar em como seria difícil desenvolver um sistema complexo, com dezenas de funcionalidades e requisitos, sem o apoio de uma documentação para guiar todo esse processo? Certamente, seria uma tarefa problemática e as chances de obter um resultado de baixa qualidade seriam altíssimas.

Por isso, os diagramas UML são essenciais para evitar dois problemas comuns no desenvolvimento de software: os erros das fases de especificação do projeto e a comunicação entre as diferentes partes envolvidas, como gerentes, pessoas desenvolvedoras, analistas, Scrum Master e Product Owner, por exemplo.

Com o uso desses diagramas, é possível obter uma visão clara e única do sistema, deixando todas as entidades envolvidas no projeto a par do que será desenvolvido e evitando erros de implementação. Afinal, trata-se de uma linguagem padrão, objetiva e eficiente que facilmente será entendida por toda a equipe.

Quais são os tipos de diagramas UML?

Existem diversos tipos de diagramas UML e eles são divididos em duas categorias: os diagramas estruturais e os comportamentais. Cada um deles é usado para especificar, documentar, modelar e visualizar aspectos específicos de uma aplicação. Veja!

Diagramas estruturais

Os diagramas estruturais são usados para modelar os aspectos estáticos do sistema. Entre eles, podemos citar a arquitetura, as classes, as interfaces, os métodos, os componentes, etc.

Diagrama de classes

O diagrama de classes é um dos modelos mais importantes no processo de engenharia de software e serve de base para outros diagramas. Isso porque ele é utilizado para mapear o sistema por meio da modelagem dos seus métodos, atributos e classes — além dos relacionamentos definidos entre elas, como herança, composição, associação e dependência.

Material complementar:
https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-classe-uml

Diagrama de objetos

O diagrama de objetos é usado para visualizar instâncias específicas das classes que foram definidas no diagrama de classes. Ou seja, ele mostra os objetos, seus valores e relacionamentos em um determinado momento da execução do programa.

Material complementar:
https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-objetos-uml

Diagrama de componentes

Esse diagrama é utilizado para indicar como o sistema será implementado, quais são as suas interfaces, pacotes e artefatos, além dos relacionamentos entre os seus diferentes componentes. Nele, é possível destacar a função de cada módulo da aplicação, o que facilita a sua reutilização.

Material complementar:
https://www.lucidchart.com/pages/pt/diagrama-de-componentes-uml

Diagrama de implantação

Também conhecido como diagrama de instalação, esse modelo mostra a relação entre os recursos de infraestrutura e os artefatos do sistema. Ele ainda representa os requisitos mínimos de hardware, mapeando as necessidades do software que será implementado.

Material complementar:
https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-implementacao-uml

Diagrama de pacotes

O diagrama de pacotes é usado para representar os subsistemas existentes em uma aplicação. Nesse modelo, cada pacote agrupa elementos que estão relacionados — como classes, diagramas e até outros pacotes — mostrando sua organização, disposição e relação com outros elementos.

Material complementar:
https://www.lucidchart.com/pages/pt/diagrama-de-pacotes-uml

Diagrama de estrutura

Por meio deste diagrama é possível representar as colaborações internas que ocorrem entre classes, componentes, instâncias ou interfaces para a execução de uma tarefa específica.

Material complementar:
https://www.lucidchart.com/pages/pt/diagrama-de-estrutura-composta-uml

Diagramas comportamentais

Como o próprio nome indica, os diagramas comportamentais são utilizados para especificar como o sistema se comporta diante de determinadas interações.

Material complementar:
https://www.lucidchart.com/pages/pt/o-que-e-uml

Diagrama de casos de uso

Usado principalmente na fase de especificação dos requisitos, o diagrama de casos de uso documenta as funcionalidades do sistema e as interações da pessoa usuária com cada uma delas. É um diagrama simples e não exige conhecimentos técnicos para ser compreendido.

Material complementar:
https://medium.com/operacionalti/uml-diagrama-de-casos-de-uso-29f4358ce4d5

Diagrama de máquina de estados

Esse diagrama é usado para descrever como uma aplicação responde aos eventos internos e externos. Nele, é possível demonstrar as mudanças de estado que um objeto apresenta durante a execução de determinado processo

Material complementar:
https://dtic.tjpr.jus.br/wiki/-/wiki/Governan%C3%A7a-TIC/Modelo+de+M%C3%A1quina+de+Estados/pop_up.

Diagrama de atividades

No diagrama de atividades é mostrado o fluxo percorrido até a conclusão de uma atividade. Por meio dele, pode-se representar as operações feitas entre os objetos durante cada atividade.

Material complementar:
https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-atividades-uml

Diagrama de interação

O diagrama de interação é dividido em quatro tipos diferentes: diagrama de visão geral, diagrama de sequência, diagrama de comunicação e diagrama de tempo. O primeiro mostra o fluxo principal das interações dentro do sistema.

Já o de sequência descreve a interação entre os objetos de um caso de uso. Por sua vez, o diagrama de comunicação complementa o anterior, mostrando os vínculos entre cada objeto. Por fim, o diagrama de tempo é usado para descrever o comportamento das instâncias de uma classe durante um intervalo específico.

Material complementar:
https://www.lucidchart.com/pages/pt/diagrama-de-interacao-uml

Quando usar os diagramas UML?

Os diagramas UML devem ser usados em diversas situações, por exemplo:

- quando é preciso oferecer uma visão padronizada do projeto e de suas especificações para todas as pessoas envolvidas no desenvolvimento;
- para documentar e visualizar o funcionamento do software;
- comunicar os protocolos de interfaces que serão consumidas por outros sistemas;
- auxiliar a fase inicial de especificação dos requisitos do sistema;
- Quando é preciso documentar os desejos de algum cliente sobre as funcionalidades do software.

Como foi possível perceber, a linguagem UML e os seus diagramas são ferramentas imprescindíveis para a programação. Portanto, atente-se a elas e busque conhecer cada vez mais os diferentes processos e tecnologias envolvidas no desenvolvimento de software. Assim, você estará um passo mais perto de atingir o sucesso na sua carreira!