DESCREVER AS VISÕES, A SÍNTESE GERAL E OS DIAGRAMAS DA UML

O QUE É UML, AFINAL?

No final dos anos 1990, as linguagens de programação orientadas a objeto já eram uma realidade, e cada profissional que desenvolvia atividade de análise e projeto de sistemas criava seus próprios modelos, baseados em suas necessidades e realidade. Ou seja, não havia consenso no mercado sobre os modelos a serem usados para modelagem de sistemas desenvolvidos sob a tecnologia de orientação a objetos.

Três competentes profissionais despontavam com seus modelos naquele momento:

• Ivar Jacobson, idealizador do método OOSE (*Object-Oriented Software Engineering*).

• James Rumbaugh, criador do método OMT (*Object, Modeling Technique*).

• Grady Booch, criador do método que leva seu nome.

A UML foi então adotada pela OMG (*Object Management Group*), em 1997, oriunda da integração dos três métodos anteriormente descritos, como uma **linguagem de modelagem padrão** para sistemas desenvolvidos sob o paradigma orientado a objetos.

UML tornou-se o padrão para modelagem gráfica, não apenas para objetos e, de fato, faz sentido essa afirmativa, pois a UML pode ser a linguagem de modelagem para diagramar sistemas concorrentes e distribuídos.

(FOWLER, 2005.)

Desde sua versão inicial, a UML sofreu mudanças substanciais e atualmente encontra-se em sua versão 2.5.1, de dezembro de 2017.
A UML é, portanto, uma linguagem padrão para construção de projetos de sistemas, voltada para a visualização, a especificação, a construção e a documentação de artefatos de um sistema. Foi projetada para ser **independente do método ou processo de desenvolvimento** utilizado.
A UML é uma linguagem de modelagem, **não é um método de desenvolvimento nem tampouco uma metodologia ou um processo de desenvolvimento de sistemas**, uma vez que não determina a ordem e nem como os diagramas devem ser usados. Simplesmente disponibiliza os diagramas, sob as várias visões necessárias ao desenvolvimento de sistemas, e cada empresa (ou equipe de desenvolvimento) a utiliza da forma como lhe convenha, ou seja, adequando a UML ao seu processo ou metodologia de desenvolvimento de sistemas.

A UML é uma linguagem destinada a:

Visualização

A modelagem gráfica facilita a compreensão do sistema e das decisões tomadas em análise e projeto, além de melhorar a comunicação entre a equipe, permitindo sua interpretação sem ambiguidades.

Especificação

Permite a construção de modelos precisos, não ambíguos e completos sob diferentes visões e atendendo às necessidades de modelagem das diferentes fases do processo de desenvolvimento de softw

Construção

Os diagramas UML podem ser integrados às principais e mais populares linguagens de programação do mercado, tais como Java e C++. Mas, para isso, terá que buscar uma solução integrada de **ferramenta CASE (*Computer-Aided Software Engineering*)** que gere código fonte (para linguagens específicas) a partir de alguns diagramas UML.

Resumindo
A UML é uma linguagem de modelagem padronizada.

• A UML é independente de tecnologia, adequando-se a todo método, metodologia ou processo de desenvolvimento.
• A UML não diz quais diagramas usar e nem em que ordem, pois a metodologia de desenvolvimento ditará essa ordem.
• A UML disponibiliza diagramas sob diferentes visões ou perspectivas.
A UML se tornou não somente a notação gráfica dominante dentro do mundo orientado a objetos, como também uma técnica popular nos círculos não orientado a objetos.

(FOWLER, 2005)

VISÕES DA UML

Assim como vimos nos exemplos do empreendimento imobiliário, as plantas das unidades, a planta elétrica, a planta hidráulica, dentre outras, cada modelo tinha uma perspectiva de compreensão da mesma realidade (unidades residenciais em um empreendimento). No mundo de sistemas computacionais, o mesmo acontece com relação à modelagem de sistemas com UML. Os autores da UML entendem que um sistema deve ser visto sob cinco diferentes perspectivas ou visões, descritas a seguir:

VISÃO DE CASOS DE USO

Assim como a maquete permite uma perspectiva geral do empreendimento imobiliário, sob o ponto de vista externo (visão do comprador), a visão de caso de uso permite olhar o sistema sob o ponto de vista externo, do usuário, descrevendo seu comportamento por conjunto de interações usuário-sistema. Tal qual a maquete, é a primeira perspectiva de um empreendimento; a visão de caso de uso é criada no estágio inicial do

desenvolvimento e guia todas as demais visões, na medida em que captura os requisitos funcionais que definem o sistema.

VISÃO DE PROJETO (OU LÓGICA)

Permite visualizar o sistema sob o ponto de vista de sua estrutura interna e seu comportamento, em resposta às funcionalidades externamente percebidas por seus usuários. Enfatiza os pacotes, as classes, as interfaces, os subsistemas (pacotes) e as colaborações.

VISÃO DE IMPLEMENTAÇÃO (OU DE DESENVOLVIMENTO)

Compreende o gerenciamento das versões do sistema, ou seja, de suas implementações utilizáveis por seus usuários. Compreendem os componentes, subsistemas e arquivos que compõem fisicamente o sistema.

VISÃO DE IMPLANTAÇÃO (OU FÍSICA)

Enfatiza a distribuição física do sistema em suas partes (subsistemas e componentes) e as respectivas conexões entre elas. Enfatiza também a organização física dos computadores e as conexões entre eles (a rede de computadores).

VISÃO DE PROCESSO

Enfatiza aspectos físicos mais peculiares como concorrência, sincronismo entre sistemas e desempenho (performance, confiabilidade, tolerância a falhas e outros aspectos) do sistema, considerando os processos e os processadores.
A UML implementa diagramas que atuam nas cinco visões descritas anteriormente, permitindo ampla modelagem sobre todos os aspectos (visões) relevantes do sistema.

Nem todas as perspectivas ou visões são úteis em todo desenvolvimento, pois dependem das características, tipo e complexidade do sistema. A imagem a seguir mostra a ideia central da visão de casos de uso, influenciando todas as demais.

UML E INTEGRAÇÃO COM PROCESSOS DE DESENVOLVIMENTO

A UML é independente de metodologia e processo de desenvolvimento. Isso é positivo para os fabricantes de software que implementam UML, pois não limita seu mercado.

Sendo assim, cabe a cada empresa ou equipe de desenvolvimento a integração da UML com sua metodologia de desenvolvimento de sistemas computacionais, permitindo uma modelagem eficiente.

A UML pode ser usada de diversas formas, desde esboços manuais para interação com usuários ou equipe, passando pelo uso de ferramentas de diagramação UML até sofisticadas ferramentas CASE, que integram os modelos e geram código em linguagens específicas.
A UML então pode ser adaptada em qualquer processo, seja ele:

EM CASCATA

Com e sem retroalimentação, onde as fases se sucedem, a seguinte inicia quando a anterior termina. A desvantagem é que, se for sem retroalimentação, não há opção de retorno à fase anterior. Por exemplo, se na fase de projeto identificamos novos requisitos, estes não podem ser considerados, pois a fase de análise congelou os requisitos identificados. Se a retroalimentação for permitida, esse problema pode ser minimizado, mas não solucionado. O grande problema é que o usuário interage com a equipe de desenvolvimento no início do processo e ao final, quando o sistema é entregue.

ITERATIVO

Onde o sistema é dividido em subconjuntos de funcionalidades (com mínimo de dependência com os demais conjuntos), e as atividades de análise, projeto, implementação, teste e implantação são realizadas a cada subconjunto. Isso significa que a cada subconjunto haverá a implantação de uma parte do sistema, permitindo que ajustes das partes encerradas ocorram em paralelo com o novo subconjunto de funcionalidades que está sendo construído.

ÁGIL

Os processos são ditos ágeis porque compartilham um conjunto de valores e princípios definido pelo manifesto ágil. O foco aqui é permitir modificação sempre que preciso e no desenvolvimento de código. A modelagem existe, mas em menor escala e com ênfase na comunicação com usuário e equipe de desenvolvimento. Visa a pouca formalidade nos processos ágeis. Mais usados: Extreme Programming (XP), Scrum, FDD (Feature Driven Development).

RUP (*RATIONAL UNIFIED PROCESS*)

O mercado muito fala da integração UML e RUP, mas, como os demais processos, o RUP é independente da UML. O RUP, na verdade, é uma estrutura de processo, que vai usar casos de desenvolvimentos (processo a ser usado), a maioria deles iterativos. O RUP não se adapta a processo em cascata.

VISÃO GERAL DOS DIAGRAMAS UML

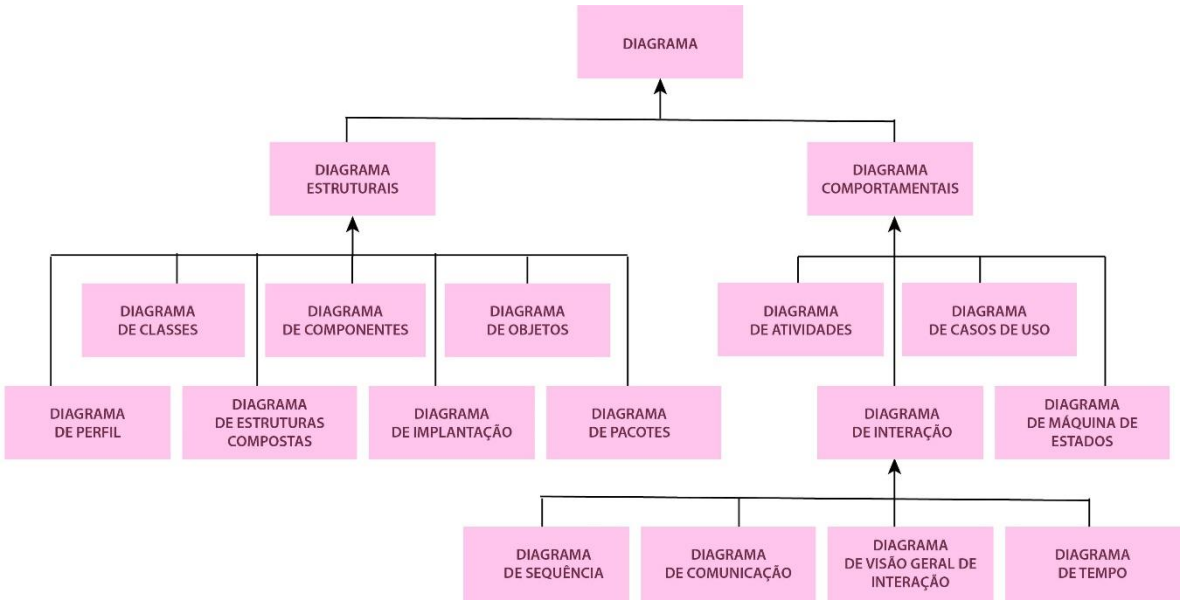
Vamos nos ater aqui à versão mais recente da UML, a 2.5.1, que traz 14 diagramas divididos em estruturais e comportamentais, conforme imagem a seguir.

Os diagramas estruturais

Dizem respeito às estruturas estáticas necessárias ao sistema, como os pacotes, as classes, os objetos, os componentes e a estrutura de nós (estruturas computacionais). Também são chamados de estruturas estáticas.

Os diagramas comportamentais

Evidenciam o comportamento (funcionamento) de parte de um sistema ou processo de negócio relacionado ao sistema, segundo determinada perspectiva. Dizem respeito às funcionalidades do sistema, aos estados de um objeto em seu ciclo de vida, às interações entre os objetos, dentre outros aspectos. Também são chamados de diagramas dinâmicos.



Diagramas da UML

A seguir, uma breve descrição de cada diagrama da UML:

Diagrama	Especificação
Diagrama de perfil	Permite a definição de tipos padronizados, como estereótipos, restrições e valores rotulados. O foco é a adequação aos modelos UML para diferentes plataformas, por exemplo.
Diagrama de classes	Descreve, para cada classe, suas propriedades (atributos e métodos) e seus

	relacionamentos com as demais classes. Classe é a base estrutural dos sistemas orientados a objetos.
Diagrama de estruturas compostas	Possibilita a descrição de colaborações internas de classes, interfaces ou componentes, visando a especificação de uma funcionalidade.
Diagrama de componentes	Apresenta a estrutura e as conexões entre os componentes de um sistema.
Diagrama de implantação	Especifica o ambiente computacional sobre o qual o sistema vai operar, além de distribuir os artefatos (pacotes, classes e componentes) nos nós (elementos computacionais).
Diagrama de objetos	É um diagrama de classes instanciado, ou seja, mostra exemplos de instâncias de classes.
Diagrama de pacotes	Descreve estruturas hierárquicas que englobam outros elementos, como outros pacotes, casos de uso, classes. Usado para modularizar logicamente um sistema em suas partes relevantes (subsistemas).
Diagrama de atividades	Descreve o comportamento de procedimentos, processos de negócios e fluxos de trabalho, suportando processamento sequencial e paralelo.
Diagrama de casos de uso	Mostra como os usuários (atores) interagem com o sistema, do ponto de vista externo, evidenciando as funcionalidades com as quais interagem.
Diagrama de estados	Mostra como os eventos que afetam o sistema alteram o estado dos objetos ao longo de seus ciclos de vida.
Diagrama de sequência	Mostra como os objetos interagem, evidenciando a linha de tempo (a sequência das interações).
Diagrama de comunicação	Mostra como os objetos interagem, evidenciando as conexões e colaborações entre eles.

Diagrama de visão geral de interação	Um mix de diagrama de sequência e de atividades. Uma especialização do diagrama de atividades para interações.
Diagrama de tempo	Diagrama de interação, com foco nas restrições de temporização, como, por exemplo, para evidenciar as mudanças no estado de um objeto ao longo do tempo.

DIAGRAMAS UML E SUA UTILIZAÇÃO NAS FASES

O descrito a seguir não é recomendado ou estipulado pela UML, pois a UML não determina quais diagramas devem ser usados e tampouco em que ordem devem ser elaborados. São dicas práticas.

Dificilmente usamos todos os diagramas da UML em um único sistema. Há um conjunto de 4 a 8 diagramas que são os mais usados: pacotes, **casos de uso, classes, sequência** (ou comunicação), **estados**, atividades, componentes e **implantação**. Todos os mencionados são relevantes, mas os em negrito são os essenciais.

No módulo 1, descrevemos as atividades das fases de análise (levantamento e análise de requisitos) e projeto independentemente do processo de desenvolvimento por entender que ambas estarão presentes em qualquer processo que usemos. Agora, partindo desse mesmo pressuposto, vamos descrever os diagramas UML que podem ser usados em cada uma dessas três fases. Citaremos apenas os diagramas mais usados.

Diagramas UML na atividade de análise

Levantamento de requisitos	Análise de requisitos
Esboço inicial do **diagrama de pacotes**, para grandes sistemas, evidenciando a necessidade de particionamento	Detalhamento e aprofundamento do **diagrama de pacotes**
Esboço inicial do **diagrama de casos de uso**	Detalhamento e aprofundamento do **diagrama de casos de uso**
Esboço inicial do **diagrama conceitual de classes**, em alto nível (sem detalhes)	Detalhamento e aprofundamento do **diagrama conceitual de classes**
	Diagrama de estados, para os objetos mais complexos durante seu ciclo de vida
	Diagrama de atividades, para elucidar um processo ou fluxo de trabalho relevante ou ainda para elucidar um caso de uso mais complexo

Nota: o diagrama conceitual de classes evidencia as classes (com principais atributos e inicialmente sem métodos) e os principais relacionamentos (em geral apenas associações, que são os relacionamentos de classes mais elementares).
Os diagramas usados no levantamento de requisitos são, em geral, esboços para a própria pessoa ou para debater com colegas da equipe. Já na fase de análise de requisitos, alguns diagramas, em geral pacotes e casos de uso, são usados para conversas com usuários.

Diagramas UML na atividade de projeto

Adição de detalhes ao **diagrama conceitual de classes**, que passa a ser o **diagrama de classes de projeto**.
Diagrama de sequência ou diagrama de comunicação, para cenários de uso mais relevantes, o que ajuda a identificar métodos das classes envolvidas na interação.
Detalhamento dos **diagramas de estados** elaborados na atividade de análise, podendo modelar de outros objetos percebidos.
Diagrama de componentes, caso o software use algum já pronto ou a ser construído.
Diagrama de implantação, evidenciando as unidades computacionais e alocando os componentes nelas.
Diagrama de atividades, esclarecendo métodos de classes mais complexos ou que usem processamento paralelo.

Nota: o diagrama de classes de projeto deriva do diagrama conceitual de classes, agregando novos atributos, todos os métodos necessários, identificando os corretos relacionamentos entre as classes (e não apenas associações), adicionando as multiplicidades e outros elementos relevantes da UML.