****

**INSTITUTO FEDERAL DO NORTE DE MINAS GERAIS**

**ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

LUÍS FERNANDO FERREIRA DOS SANTOS

# Backlog 27-02-23

Almenara/MG

2023

LUÍS FERNANDO FERREIRA DOS SANTOS

# Backlog 27-02-23

# Backlog 27-02-23

Orientador: Prof. Pablo.

Almenara/MG

2023

**Modelos de Software**

“All models are wrong, but some models are useful. So the question you need to ask is not is the model true? (it never is) but is the model good enough for this particular application?” – George Box

E assim começa a discussão acerca dos modelos de software, que desde o começo muito se foi pensado se os modelos eram realmente úteis, ou se poderia se fazer software livremente, diferente de todos os outros produtos que já eram produzidos industrialmente, feitos em série, seguindo regras e padrões já muito bem estabelecidos, conhecidos e validados pelas diversas empresas do mercado.

Na visão de Box, os modelos, no princípio da discussão eram apenas os matemáticos, estavam totalmente errados, porque apesar de serem uma aproximação da realidade, eles estão muito distantes daquilo que se pretende realmente fazer, ou representar. E, por outro lado, numa perspectiva ainda mais técnica, o software não tem uma modelagem tão simples quanto uma representação de um modelo matemático, ou uma hipótese química, física ou biológica, devido às diversas camadas e a complexidade envolvida na concepção, no design, no desenvolvimento e na manutenção do software depois de pronto.

“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.”

Nesse contexto, o uso da UML gerou um padrão em que todos os desenvolvedores, ao baterem os olhos podiam saber como aquele sistema seria desenvolvido, baseado nos requisitos que estavam especificados no sistema de caso de uso, e os modelos, que eram feitos em tabelas com a representação das classes, métodos e etc, já que nessa época já havia começado o conceito de orientação a objetos.

As UML tem outros usos, como a engenharia reversa, em que, depois do sistema estar pronto, pode-se usar o diagrama para explicar uma funcionalidade mais facilmente, sem lidar diretamente com as linhas no código-fonte. Além de que, uma representação visual bem-feita é muito proveitosa para facilitar como cada componente deve se comportar em meio ao software em si.

Já sobre os diagramas de classe, temos uma leve impressão de que eles são um pouco desatualizados, embora sejam um conceito muito útil baseado em UML, que pode ser adotado independentemente da natureza do seu projeto.

O diagrama é feito em mini tabelas, em que são descritas as classes, os métodos, as variáveis e seus tipos. Já o relacionamento entre as tabelas é feito com as flechas, que vão de uma a outra para poder representar as relações de dependência entre as mesmas.

Os atributos podem ser públicos ou privados, e são indicados pelo sinal - antes de cada um. O tipo de cada um também deve ser informado, como int, float, double, char, etc. Já os métodos também têm uma representação, se são públicos ou privados, com o sinal de + (público), ou - (privado). No caso dos métodos, temos as variáveis de entrada informadas, e o valor de retorno que esperamos daquele método.

Baseado de novo no conceito de orientação a objetos, temos a representação da herança que vem de uma flecha não preenchida, e a relação de dependência também, a qual vem de uma seta tracejada da qual vem a classe dependente da outra.