ANÁLISE E PROJETO DE SISTEMAS

Backlog 27-02-23

RESUMO

Os requisitos descrevem o que um sistema tem como objetivo, já o código fonte é de uma certa forma o comportamento de um determinado sistema. Desse modo existe uma lacuna entre esses dois, que é somente preenchida com os modelos, que são criados com o intuito de ajudar a entender e analisar um sistema. Geralmente eles são mais detalhados que os requisitos e menos complexos que o código fonte de um sistema.

E é no desenvolvimento de um software, que a criação de modelos tem uma participação bem importante nesse processo. Durante o levantamento de requisitos o foco será na definição dos problemas que serão resolvidos pelo sistema.

UML é uma linguagem de notação utilizada para modelar softwares. Essa linguagem fornece diversos conjuntos de diagramas para ajudar no design dos sistemas. Ele foi desenvolvido por volta da década de 80 no auge da orientação a objetos e de diversas linguagens famosas até hoje, como o C + + por exemplo.

A proposta do UML era de criar modelos gráficos que seriam passados aos desenvolvedores para que eles tivessem um melhor entendimento do que seria desenvolvido e pudessem converter aquele modelo em código fonte.

O UML possui três formas de utilização que são:

**UML como blueprint** essa forma de uso defende que após o levantamento de requisitos, deve ser criado um conjunto de modelos que documentam diversos aspectos de um sistema e utilizando de diagramas UML. esses modelos seriam feitos por analistas de sistemas que utilizam de ferramentas case e depois passariam esses modelos para os programadores para a codificação.

**UML como linguagem de programação** é a utilização do UML de uma forma que não seria mais necessário uma fase de codificação, pois o código seria gerado de forma direta a partir da compilação dos modelos UML, essa forma ficou conhecida como desenvolvimento dirigido por modelos.

e por último o **UML como esboço** nessa forma o UML é utilizado para construir diagramas leves e informais das partes de um sistema, e esses diagramas são usados para a comunicação dos desenvolvedores em duas situações principais que são:

**Engenharia Avante** é quando os desenvolvedores usam de modelos UML para discutir e analisar alternativas de design, antes mesmo de criar qualquer código.

Já a **Engenharia Reversa** é quando os desenvolvedores usam dos modelos UML para analisar e discutir uma funcionária que já está implementada no código fonte.

Nessas duas situações, o objetivo não é ter modelos completos e detalhados. Dessa forma, não é considerado o uso de ferramentas complexas e caras, como as ferramentas CASE.

Os Diagramas UML são classificados em dois grandes grupos:

**Diagramas Estáticos** que modelam a estrutura e a organização de um sistema, isso incluindo até informações sobre classes, atributos, métodos, pacotes e etc.

e os **Diagramas dinâmicos** que modelam eventos que ocorrem durante a execução do sistema.

Os Diagramas de Classes são os diagramas mais utilizados da UML. Oferecendo representações gráficas para um conjunto de classes além de fornecer informações sobre atributos, métodos e relacionamentos entre as classes que foram modeladas.

Geralmente os diagramas de classes são desenhados usando-se retângulos e setas. Cada uma sendo representada por meio de um retângulo com três compartimentos.

Os diagramas de classes também contém os seus três tipos de relacionamentos que são:

**Associações** por exemplo uma associação é uma classe que faz parte de um relacionamento de associação entre outras classes. E você pode conectar uma classe de associação a um relacionamento de associação para fornecer informações adicionais sobre o relacionamento.

A **Herança** é a capacidade que as instâncias de uma classe filha ou Subclasse de acessar dados e procedimentos ou métodos associados com uma Classe Parente ou Superclasse. Ou seja, uma Subclasse herda todos os métodos e dados da Superclasse.

Já as **Dependências** é um tipo de diagrama UML que mostra as relações entre classes e objetos em um sistema de software. Ele representa a dependência entre as classes indicando que uma classe usa uma outra classe para realizar uma tarefa ou um serviço.