**INSTITUTO FEDERAL DO NORTE DE MINAS GERAIS  
Campus Almenara - MG**

**Análise e Desenvolvimento de Sistemas**

**Mattheus Carvalho Martins**

**RESUMO DOS CAPÍTULOS 4, 4.1, 4.2, 4.3 , 4.3.1, 4.3.2 E 4.3.3**

Almenara-MG

2023

**Mattheus Carvalho Martins**

**RESUMO DOS CAPÍTULOS 4, 4.1, 4.2, 4.3 , 4.3.1, 4.3.2 E 4.3.3**

Trabalho para apresentação da disciplina: Análise e Projeto de Sistemas.

Professor(a): Pablo Castro

Almenara-MG

2023

O capítulo 4 começa nos apresentando o que será lido ao longo dos capítulos, também nos apresenta o UML e sobre qual visão teremos ao aprender UML, além de 4 tipos de diagramas, que são os diagramas de classes, diagramas de pacotes, diagramas de sequência e diagrama de atividades.

O capítulo 4.1 começa falando sobre modelos de software. Os modelos de software atuam entre os requisitos, que são “o que” o sistema deve fazer, que foi falado nos capítulos anteriores, e o código do sistema em si, que é bem abstrato e impossível de ser lido por alguém que não entende da linguagem de programação ou das regras de negócio do sistema. Os modelos de software entram no meio desses dois, pois eles têm bem mais detalhes do que os requisitos em si, mas ainda é bem mais fácil de compreender do que as linguagens de programação.

Os modelos não são ou foram criados para a engenharia de software. Outras engenharias possuem esses modelos, e o livro vai dar o exemplo da engenharia civil, onde para provar que uma ponte ou outra construção é confiável, eles apresentam vários modelos matemáticos e físicos para medir e entender a capacidade máxima de carga suportada por exemplo, dentre outras coisas que é possível explorar com esses modelos.

Porém, na engenharia de software, esses modelos são bem menos efetivos do que os modelos das outras engenharias, pois eles descartam a complexidade que é fundamental para os sistemas modelados. O livro faz citação de Frederick Brooks sobre o famoso “Não existe bala de prata”.

Mas mesmo assim, não podemos dizer que os modelos de software são completamente inúteis, já que eles são muito usados na fase de modelagem e design do sistema. Os modelos são muito importantes principalmente para identificar qual o problema que iremos resolver e, se houverem coisas a melhorar ou não necessárias, já pode ser percebido antes de passar para a etapa de desenvolvimento, fazendo com que se economize tempo e recurso.

O UML (*Unified Modelling Language*) é uma linguagem que ajuda no design de sistemas, principalmente em sistemas orientados a objetos, e ainda define um conjunto de diagramas para documentar.

O UML surge na década de 80, quando a programação orientada a objetos estava no seu pico, e várias linguagens orientadas a objetos foram criadas, alguns projetos usavam o UML como os modelos gráficos seriam criados e repassados para os programadores do projeto. É importante lembrar que neste período, o modelo mais comum de se utilizar era o Waterfall (cascata).

A primeira proposta do UML foi feita em 1995, e foi nesse período que também foram criadas as ferramentas de desenvolvimento ou de desenho em UML. O nome dado às ferramentas era inspirado na ferramenta já existente para as outras engenharias que é o CAD (*Computer Aided Design*), e as ferramentas de desenho em UML foram chamadas de CASE (*Computer-Aided Software Engineering*).

Existem três formas de se utilizar o UML. O primeiro é o UML como blueprint que seria um modelo desenvolvido logo após o levantamento dos requisitos. O segundo seria o UML como uma linguagem de programação, onde depois de terminado o UML, a linguagem de programação seria gerada automaticamente, sem a necessidade da etapa de desenvolvimento. A terceira maneira seria o uso do UML como esboço, que será o que estudaremos. Essa terceira maneira é dividida em Engenharia avante que o UML é usado antes do desenvolvimento e a engenharia reversa, onde depois de desenvolvido o sistema, o UML é montado para ou explicar algo que falta, ou algo para um integrante novo no grupo.

Os diagramas UML são divididos em dois grupos, os diagramas estáticos, que seria a organização e estrutura de um sistema, utilizando vários elementos já da parte de programação, como classes, pacotes, atributos, métodos, etc. Nessa atividade, estudaremos apenas um exemplo de diagramas que será o de classe. O outro grupo seria os diagramas dinâmicos, que seriam modelos que ocorrem eventualmente durante a execução do sistema. Um exemplo seria uma sequência chamada de métodos para o sistema.

Os diagramas de classes são os diagramas mais usados, pois ele é representado por basicamente classes com seus atributos e métodos. Basicamente, a representação de uma classe é um retângulo com 3 divisões; a primeira é o nome, a segunda são os atributos e o terceiro os métodos. Para os atributos, é necessário colocar o seu tipo e para os métodos, o tipo de retorno.

Associação seria um atributo que se torna um objeto, onde essa associação seria representada com uma seta preta de uma para outra classe. Ainda podemos colocar valores acerca da carnalidade da associação.

Herança seria quando você possui uma classe que possui as mesmas características de outra classe e, para economizar recursos, você herda esses atributos. Em UML, a herança é representada por uma seta branca que sai da classe filha para a classe pai.

Uma dependência ocorre quando uma classe usa outra classe, porém esse uso não ocorre por meio de associação ou herança. Dependências ocorrem, por exemplo, quando um método de uma classe A declara um parâmetro ou variável local do tipo B ou quando um método de A lança uma exceção do tipo B.