Introdução a Análise Orientada a Objetos

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Outubro de 2019

Qualidade de *Software*

- Obter *software* de qualidade é um dos principais objetivos da Engenharia de *Software*;

- Desafio desde os primórdios da computação;

- Glend Myers(1975), escreveu que: obter qualidade em *software* de grande porte, era um dos maiores desafios para época.

UML

- UML (*Unified Modeling Language*) foi criado por: **Grady Booch**, **James Rumbaugh**, e **Ivar Jacobson** que são conhecidos como "os três amigos";
- os três amigos são autores independentes de modelagem orientado a objetos e junto a eles haviam muitos outros autores;
- Os três decidiram unir seus conhecimentos e desenvolveram a UML, que é a junção do que havia de melhor nas três metodologias distintas.

Objetivos da UML

- Os objetivos da UML são:
  - a modelagem de sistemas, não apenas de *software*, usando os conceitos da orientação a objetos;
  - estabelecer uma união, fazendo com que métodos conceituais sejam também executáveis e
  - criar uma linguagem de modelagem usável, tanto pelo homem quanto pela máquina.

Fases do Desenvolvimento UML

- Existem cinco fases no desenvolvimento de sistemas de *software*:
  - análise de requisitos,
  - análise,
  - *design* (projeto),
  - programação e
  - testes.

Análise de Requisitos

- captura as intenções e necessidades dos usuários do sistema a ser desenvolvido, através do uso do diagrama de *use-cases* (casos de uso),
- as entidades externas ao sistema, "atores externos", que interagem e possuem interesse no sistema também são modelados na forma de *use-cases*.

Análise de Requisitos

- O diagrama de casos-de-uso mostrará o que os atores externos, ou seja, os usuários do futuro sistema deverão esperar do aplicativo, conhecendo toda sua funcionalidade sem importar como esta será implementada.

Análise

- A fase de análise está preocupada com as primeiras abstrações, classes e objetos, e mecanismos que estarão presentes no domínio do problema.

Design (Projeto)

- Nesta fase, o resultado da análise é expandido em soluções técnicas;
- novas classes são adicionadas para prover uma infraestrutura técnica:
  - a interface do usuário e de periféricos,
  - o gerenciamento de banco de dados,
  - a comunicação com outros sistemas,
  - dentre outros.

Design (Projeto)

- As classes do domínio do problema, modeladas na fase de análise, são mescladas nessa nova infraestrutura técnica;
- A partir deste ponto é possível mensurar o real tamanho do domínio do problema em relação à infraestrutura;
- O *design* resulta no detalhamento das especificações para a fase de programação do sistema.

Programação

- Nesta fase, as classes provenientes do *design* são convertidas para o código da linguagem orientada a objetos escolhida.

Testes

- Um sistema normalmente é executado em:
  - testes de unidade,
  - integração e
  - aceitação.

Teste de Unidade

- Os testes de unidade (ou teste unitário) são para classes individuais ou grupos de classes e são geralmente testados pelo programador.

Teste de Integração

- Os testes de integração são aplicados já usando as classes e componentes integrados, isto, para se confirmar se as classes estão cooperando umas com as outras, como especificado nos modelos.

Teste de Aceitação

- Os testes de aceitação observam o sistema como uma "caixa preta" e
- verificam se o sistema está funcionando como foi especificado nos primeiros diagramas de "*use-cases*".

A notação UML

- Tendo em mente as cinco fases do desenvolvimento de *softwares*, são utilizados:
  - cinco tipos de visões,
  - nove tipos de diagramas e
  - vários modelos de elementos.
- Tudo isto é utilizado na criação dos diagramas e mecanismos gerais que, em conjunto, especificam e exemplificam a definição estática e dinâmica de um sistema.

Partes que Compõem a UML

- Visões,
- Modelos de Elementos,
- Mecanismos Gerais e
- Diagramas.

Visões

- mostram diferentes aspectos do sistema que está sendo modelado;
- a visão não é um gráfico, mas uma abstração consistindo em uma série de diagramas;
- definindo um número de visões, cada qual mostrará aspectos particulares do sistema, dando enfoque a ângulos e níveis de abstrações diferentes e uma figura completa do sistema poderá ser construída.

Modelos de Elementos

- Os conceitos usados nos diagramas são modelos de elementos que representam definições comuns da orientação a objetos, como:
  - as classes, objetos, mensagem, relacionamentos entre classes incluindo associações, dependências e heranças.

Mecanismos Gerais

- Os mecanismos gerais provêm comentários suplementares, informações, ou semântica sobre os elementos que compõem os modelos;
- Eles provêm também, mecanismos de extensão para adaptar ou estender a UML para um método e/ou processo, organização ou usuário específico.

Diagramas

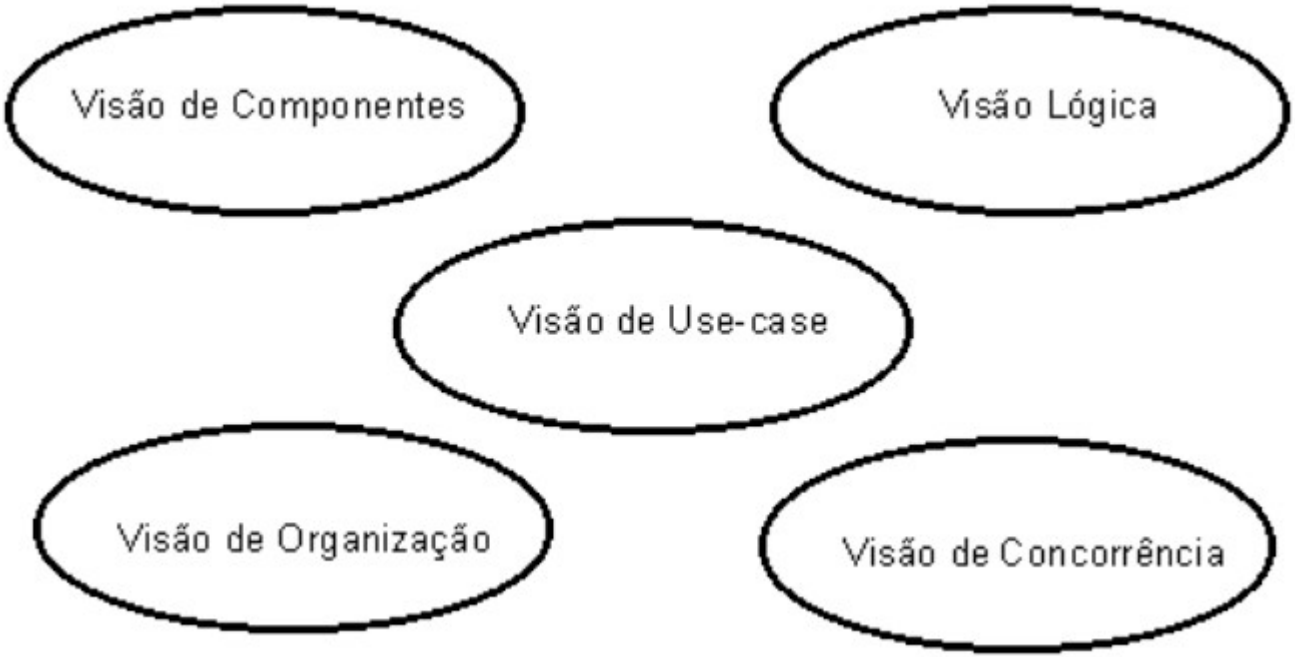
- Os diagramas são os gráficos que descrevem o conteúdo em uma visão;
- a UML possui nove tipo de diagramas que são usados em combinação para prover todas as visões do sistema.

Visões

- Um sistema é composto por diversos aspectos:
  - **funcional** (que é sua estrutura estática e suas interações dinâmicas),
  - **não funcional** (requisitos de tempo, confiabilidade, desenvolvimento, etc.) e
  - **organizacionais** (organização do trabalho, mapeamento dos módulos de código, etc.).
- Então o sistema é descrito em um certo número de visões, cada uma representando uma projeção da descrição completa e mostrando aspectos particulares do sistema.

Visões

- Cada visão é descrita por um número de diagramas que contêm informações que dão ênfase aos aspectos particulares do sistema;
- As visões que compõem um sistema são:



Visão *Use-case*

- Descreve a funcionalidade do sistema desempenhada pelos atores externos do sistema (usuários);
- A visão *use-case* é central, já que seu conteúdo é base do desenvolvimento das outras visões do sistema;
- Essa visão é montada sobre os diagramas de: ***use-case*** e de **atividade**.

Visão Lógica

- Descreve como a funcionalidade do sistema será implementada;
- feita pelos analistas e desenvolvedores;
- diferentemente da visão *use-case*, a visão lógica observa e estuda o sistema internamente;
- especifica a estrutura estática do sistema (classes, objetos e relacionamentos) e as colaborações dinâmicas (mensagens enviadas entre os objetos).

Visão de Componentes

- É uma descrição da implementação dos módulos e suas dependências;
- executada por desenvolvedores;
- consiste nos componentes dos diagramas.

Visão de Concorrência

- Trata a divisão do sistema em processos e processadores;
- propriedade não funcional do sistema,
- definine se o sistemas possui execuções paralelas, e eventos assíncronos;
- uma vez dividido o sistema em linhas de execução de processos concorrentes (*threads*), esta visão de concorrência deverá mostrar como se dá a comunicação e a concorrência destas *threads*;
- suportada pelos diagramas dinâmicos de: **estado**, **sequência**, **colaboração** e **atividade**, e
- pelos diagramas de implementação, que são os de: **componente** e **execução**.

Visão de Organização

- mostra a organização física do sistema, os computadores, os periféricos e como eles conectam entre si;
- esta visão será executada pelos desenvolvedores, integradores e testadores, e será representada pelo **diagrama de execução**.

Modelos de Elementos

- Os conceitos usados nos diagramas são chamados de modelos de elementos;
- um elemento pode existir em diversos tipos de diagramas;
- alguns exemplos de modelos de elementos são:
  - classes, objetos, estados, pacotes, componentes e os relacionamentos.

Bibliografia

- DEITEL, H. M.; DEITEL, P. J. **Java Como Programar**; tradução Edson Furmankiewicz; revisão técnica Fábio Lucchini. 6. ed., São Paulo: Pearson, 2005.
- ESCOLA TÉCNICA LAURO GOMES. UML – **Linguagem de Modelagem Unificada**. Disponível em: <http://www.etelg.com.br/paginaete/downloads/informatica/apostila_uml.pdf>. Acesso em: 09 Out. 2019.

Nivelamento

- Utilizando o conhecimento que você possui da disciplina de POO, faça um programa em "Linguagem Java" que resolva o seguinte problema:
  - crie um software que permita a um usuário *cadastrar*, *excluir*, *alterar*, *consultar* e *imprimir* quantos veículos ele quiser em m ArrayList. Cada veículo será referenciado por sua posição no ArrayList. Para cada veículo deverá ser possível guardar as seguintes informações:
    - String marca
    - String modelo
    - int anoFabricacao
    - Int anoModelo
    - float preco
  - seu software deverá, obrigatoriamente, fazer uso de uma classe "gerencia" ou "fichário".
- Envie sua solução, apenas o projeto compactado (anexado), para o e-mail:
  - eddiesaliba3@yahoo.com
  - no campo assunto coloque: ***APOO – Nivelamento***
  - e no corpo do e-mail coloque seu **nome completo**.
- ATENÇÃO!
  - antes de enviar o projeto, apague todos os arquivos ".class" e compacte-o e
  - só serão verificados os projetos que forem feitos na IDE Eclipse.