Perfil:

Emerson Antonio Klisiewicz.

Possui graduação em Ciências da Computação, pela Pontifícia Universidade Católica do Paraná (1994), Especialização em Sistemas de Informações Gerenciais , pela Pontifícia Universidade Católica do Paraná (1995), Especialização em Redes e Sistemas Distribuídos pela Pontifícia Universidade Católica do Paraná (1997), com experiência de mais de 20 anos em áreas de tecnologia de empresas estatais e financeiras. Currículo Lattes:

http://buscatextual.cnpq.br/buscatextual/visualizacv.do?id=K4282839T3

AULA 05 – UML

Introdução:

Nessa quarta aula, estaremos estudando sobre a análise orientada a objetos, UML e Ferramentas Case. Orientação a objetos apesar de antiga não era utilizada por falta de pessoas treinadas, interesse em manter a cultura adquirida, ferramentas imaturas. A UML ajudou em melhorar esse cenário completada pelas ferramentas CASE.

Contextualizando:

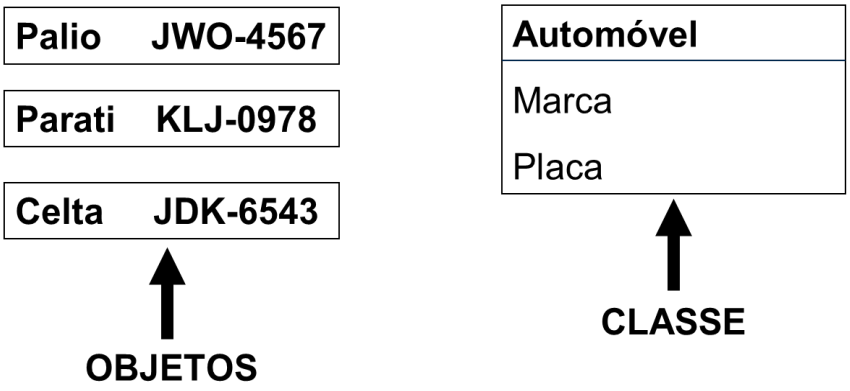
Análise Orientada a Objetos

A OO surgiu no final da década de 60, quando dois cientistas dinamarqueses criaram a linguagem Simula (Simulation Language). Em 1967 - Linguagem de Programação Simula-67- introduz os conceitos de classe e herança e o termo Programação Orientada a Objetos (POO) é introduzido com a linguagem Smalltalk (1980). No inicio dos anos 90⇨ Paradigma de Orientação a Objetos abordagem poderosa e prática para o desenvolvimento de software.

Conceitos

Criou o conceito de objeto, que é um tipo de dado com uma estrutura e operações para manipular esta estrutura. Tipos definidos pelo usuário devem se comportar da mesma maneira de tipos pré-definidos (fornecidos pelo compilador). Os objetos trocam mensagens entre si. Essas mensagens resultam na ativação de métodos, os quais realizam as ações necessárias. Os objetos que compartilham uma mesma interface, ou seja, respondem as mesmas mensagens, são agrupados em classes.

a-) Objetos e Classes



(Instâncias da classe Automóvel)

A1-) Classes:

É um tipo definido pelo usuário que contém o molde, a especificação para os objetos, assim como o tipo inteiro contém o molde para as variáveis declaradas como inteiros. A classe envolve, associa, funções e dados, controlando o acesso a estes, definí-la implica em especificar os seus atributos (dados) e suas funções membro (código). Todo objeto é uma instância de uma Classe. Possuem propriedades (ATRIBUTOS) e comportamento (MÉTODOS).

A2-) Objetos:

Tudo em Orientação a Objetos é OBJETO. Objeto, no mundo físico, é tipicamente um produtor e consumidor de itens de informação. Definição (mundo do software). “Qualquer coisa, real ou abstrata, a respeito da qual armazenamos dados e métodos que os manipulam” Martin, Odell (1995). Abstração de uma entidade do mundo real e que possuí características (VALORES). É uma instanciação de uma classe.

A3-) Atributos:

Representam um conjunto de informações, ou seja, elementos de dados que caracterizam um objeto. Descrevem as informações que ficam escondidas em um objeto para serem exclusivamente manipulado pelas operações daquele objeto. São variáveis que definem o estado de um objeto, ou seja, são entidades que caracterizam os objetos. Cada objeto possui seu próprio conjunto de atributos.

A4-) Métodos:

Quando um objeto é mapeado dentro do domínio do software, os processos que podem mudar a sua estrutura de dados são denominados

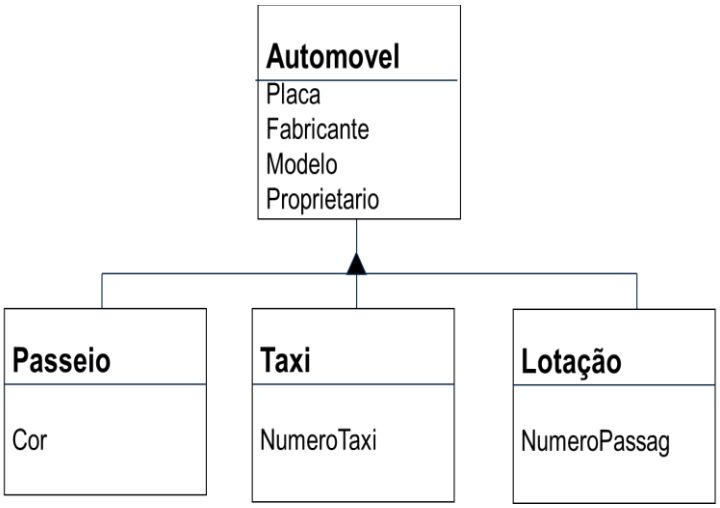
Operações ou Métodos. Métodos são invocados por Mensagens e cada objeto possui seu próprio conjunto de métodos. São procedimentos ou funções definidos e declarados que atuam sobre um objeto.

A5-) Encapsulamento:

Objetos encapsulam seus atributos. Encapsulamento é a propriedade segundo a qual os atributos de uma classe são acessíveis apenas pelos métodos da própria classe. Outras classes só podem acessar os atributos de uma classe invocando os métodos públicos. Restringe a visibilidade do objeto, mais facilita o reuso. Os DADOS e os MÉTODOS são empacotados sob um nome e podem ser reusados como uma especificação ou componente de programa.

A6-) Herança:

É o mecanismo pelo qual uma subclasse herda todas as propriedades da superclasse e acrescenta suas próprias e exclusivas características. As propriedades da superclasse não precisam ser repetidas em cada subclasse. Propicia a reutilização de código.



UML

O surgimento de sistemas de software complexos resultou na necessidade de reavaliar a forma de desenvolver sistemas. As técnicas tem evoluído de forma impressionante, notavelmente no que tange à modelagem de sistemas. Percebe-se então, a necessidade de um padrão para a modelagem de sistemas, que fosse aceito e utilizado amplamente. Surge assim a UML. A UML nasceu em 1996 como a

melhor candidata para ser a linguagem unificadora de notações. Em 1997 a UML é aprovada como padrão pela OMG e desde então tem tido grande aceitação. Atualmente na versão 2.0.

a-) Características:

É uma linguagem visual independente de linguagem de programação, independente de processo de desenvolvimento. Não é uma linguagem de programação e também não é uma metodologia.

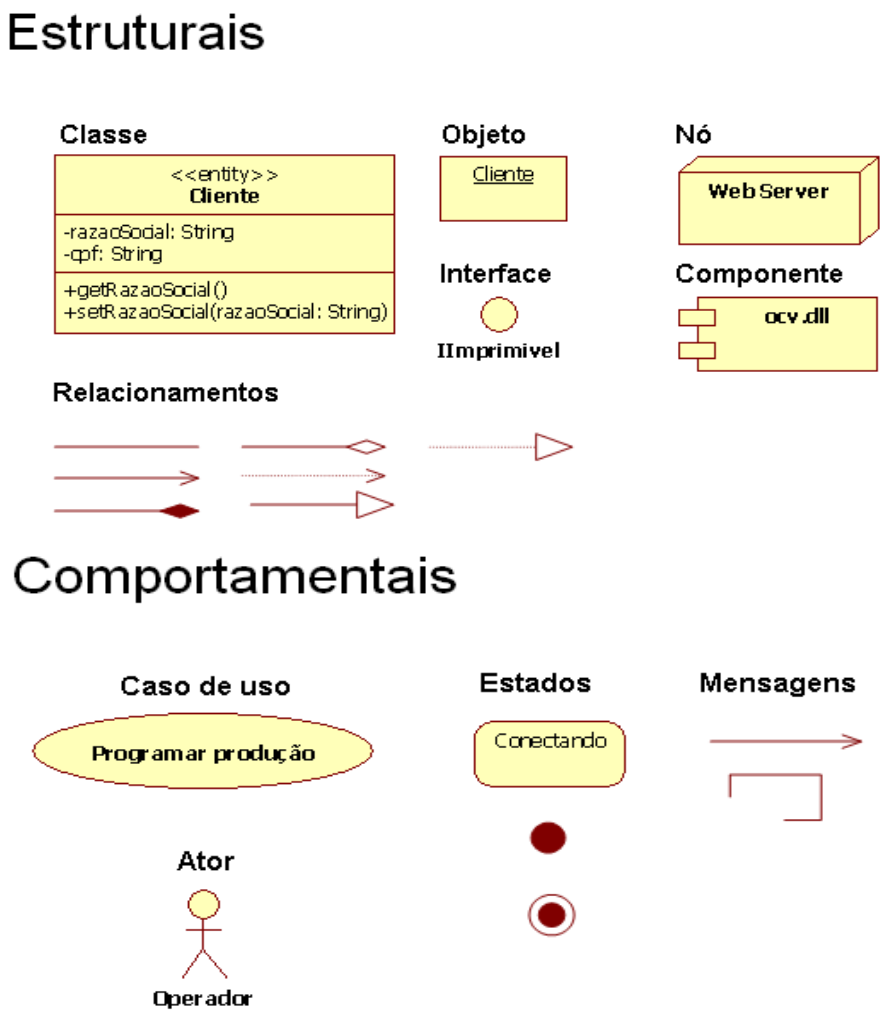
b-) Usos:

- ✓ Visualização.
- ✓ Especificação.
- ✓ Documentação.
- ✓ Comunicação.
- ✓ Construção.

c-) Diagramas:

Podem ser usados para mostrar os limites de um sistema e suas funções, representa a estrutura estática de um sistema, modelar o comportamento de objetos e apresentar a implementação física e a arquitetura de um sistema.

Podem ser estruturais e comportamentais:



Os documentos gerados em um processo de desenvolvimento são chamados de artefatos na UML.

Programas procedurais (não orientados a objeto) utilizam fluxogramas.

Programas orientados a objeto normalmente usam a linguagem gráfica UML.

Ferramentas CASE

A complexidade dos requisitos dos softwares/sistemas exige um desenvolvimento sistemático apoiado por técnicas eficazes que possibilitem mensurar os riscos de uso e provar para a comunidade que o uso do software é seguro.

Consequentemente exige-se a melhora da qualidade dos produtos e para alcançar isso:

- ✓ Necessita-se de um processo de software bem definido, assistido e monitorado.
- ✓ Necessita-se de métodos estruturados e formais para apoio ao desenvolvimento de software.

- ✓ Necessita-se de apoio às atividades do processo de software.

A ferramenta CASE oferece um conjunto de serviços, fortemente relacionados, para apoiar uma ou mais atividades do processo de desenvolvimento de software.

Estudar ferramentas CASE é estudar como construir a definição de requisitos e arquitetura e como usar processos de adoção, avaliação e seleção relacionados ao processo que se está desenvolvendo.

As ferramentas CASE podem ser:

- ✓ Horizontais: oferecem serviços utilizados durante todo o processo de software.
- ✓ Verticais: utilizadas em fases específicas do processo de software.

Também podem ser classificadas de acordo com os serviços que oferecem, dentre as quais, cita-se:

- Documentação
- Planejamento e gerenciamento de projetos
- Especificações formais
- Comunicação
- Análise e projeto de software
- Projeto e desenvolvimento de interfaces
- Programação
- Gerenciamento de Configuração
- Controle de Qualidade
-

Sua construção também demanda a utilização das técnicas de desenvolvimento de software – afinal ela é um software. Cada ferramenta tem propósitos diferentes, fornece serviços diferentes, mas possuem algumas características em comum. Do processo de desenvolvimento de software comum, destacam-se duas atividades que apresentam peculiaridades quando envolvidas no processo de desenvolvimento de uma ferramenta CASE:

- ✓ Levantamento de requisitos
- ✓ Projeto de arquitetura

a-) Requisitos de Ferramentas CASE

A captura dos requisitos do sistema junto ao usuário é um pouco diferenciada porque:

- Os usuários de ferramentas CASE não são tão bem definidos quanto os usuários de um aplicação comum.
  - ✓ São desenvolvedores
- Membros de equipes de marketing também auxiliam no processo.

- Trata-se de um produto dirigido a “mercado”.
- O processo desta fase se dá basicamente por meio de atividades macro:
- Análise do mercado.
- Análise de documentação de ferramentas similares existentes.
- Testes sobre as ferramentas similares existentes.
- Elaboração e aplicação de questionários (na forma de ciclo de questões) que deverão ser respondidos pelos desenvolvedores e pessoal de marketing.

b-) Arquitetura de Ferramentas CASE

A definição da arquitetura está intimamente relacionada ao contexto no qual a ferramenta atuará. Os principais aspectos a serem analisados são:

- As atividades do ciclo de vida que a ferramenta vai abranger.
- O repositório de dados que será utilizado.
- O estilo de interface que será adotado.
- Os serviços disponíveis em outras ferramentas que serão reutilizados.
- Quais as ferramentas existentes no mercado com as quais esta ferramenta deveria cooperar.
- Quais mecanismos de comunicação com outras ferramentas, serão utilizados.
- Quais filtros de dados serão utilizados.
- Para quais plataformas a ferramenta será desenvolvida.

Uma ferramenta CASE deve ser flexível, com arquitetura modular para facilitar sua configuração para diferentes propósitos. A arquitetura deve ser baseada em:

- Componentes: que representam os subsistemas principais e objetos da ferramenta.
- Mecanismos de interação (tecnologia de integração): que representam a forma como os componentes interagem, trocam informações e afetam uns aos outros.

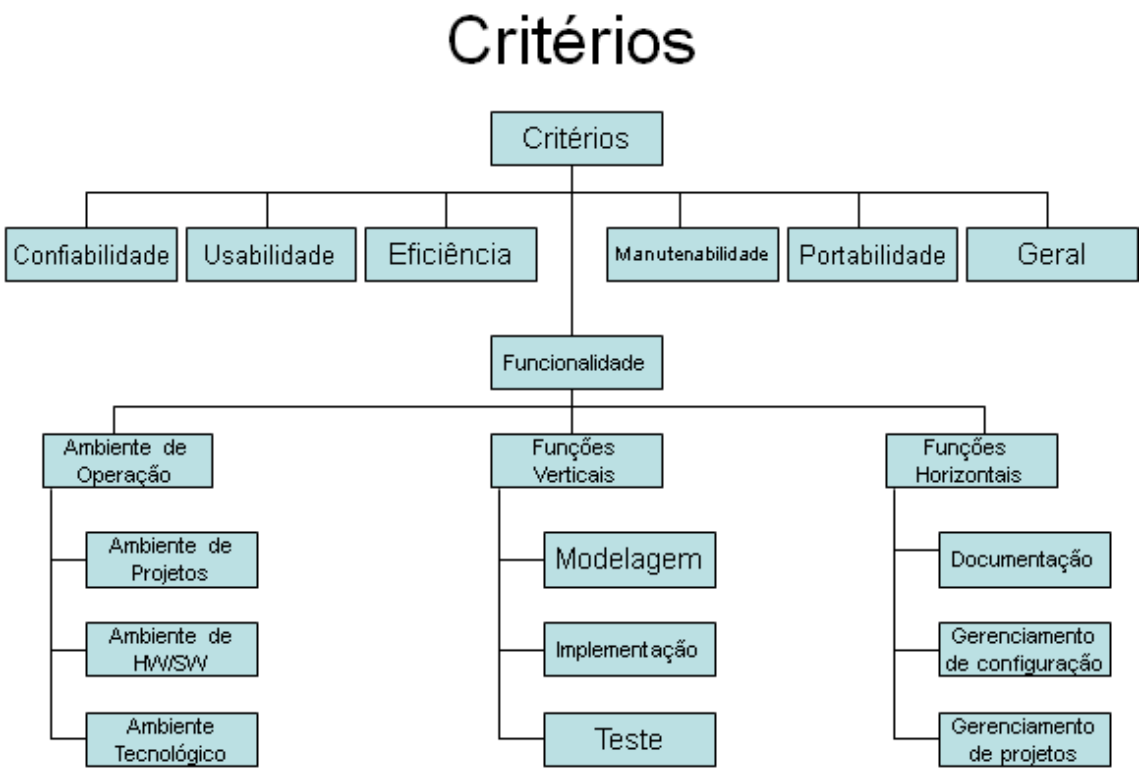
b-) Seleção de um Ferramentas CASE

Processo no qual os dados de uma ou mais avaliações de ferramentas são ponderados e comparados, considerando-se critérios definidos, para determinar se uma ou mais ferramentas podem ser recomendadas para adoção.

Passos:

- ✓ Definir o propósito da seleção.
- ✓ Definir o escopo da seleção.
- ✓ Identificar suposições e restrições.

- ✓ Definir as atividades de seleção.
- ✓ Identificar e ponderar os critérios de seleção.
- ✓ Identificar as ferramentas candidatas (quando não identificadas em um processo de avaliação prévio).
- ✓ Acessar os resultados da avaliação (quando realizada).
- ✓ Aplicar os critérios considerados aos resultados da avaliação.



Síntese

A abordagem orientada a objetos possibilita uma melhor organização, versatilidade e reutilização do código fonte, o que facilita atualizações e melhorias nos programas.

Orientação a objetos apesar de antiga não era utilizada por falta de pessoas treinadas, interesse em manter a cultura adquirida, ferramentas imaturas. Isso começa a se resolver.

Uma das formas da resolução passa pela UML, cuja forma de modelagem completa os conceitos da orientação a objetos. Com muitos diagramas a Modelagem Visual na UML permite que você construa da forma correta na primeira vez:

- ✓ Entender os requisitos do usuário
- ✓ Validar que o design atende as necessidades
- ✓ Visualizar interface, lógica de negócio e dados separadamente

- ✓ Separar domínios de negócio conforme apropriado
- ✓ Vizualizar todas as dependências
- ✓ Validar performance antes do código começar

Frequentemente a modelagem nesse formato usa algum tipo de notação gráfica e são apoiados pelo uso de Ferramentas CASE. O processo de adoção de ferramentas CASE é um processo crítico dentro de uma empresa. Existe um contraste neste processo: um aumento da oferta de ferramentas CASE no mercado contra a dificuldade das empresas em obter aumentos significativos de produtividade. O IEEE P1348 – Recommended Pratice for the Adoption of CASE Tools tenta fornecer um conjunto de questões que devem ser analisadas quando da adoção de uma ferramenta CASE, para aumentar as chances de sucesso em seu uso.

Os métodos de estruturação (independente de paradigma) utilizados no desenvolvimento de software oferecem procedimentos e notações diagramáticas que especificam a função do software em diferentes níveis de abstração, e permitem a construção do mesmo.

A sofisticação dos métodos leva a uma complexidade maior no gerenciamento do processo de desenvolvimento de software. As ferramentas entram neste processo como agentes que pretendem simplificar as ações envolvidas no processo.