Bem vindos ao nosso curso de Introdução ao Paradigma da Orientação a Objetos. Neste módulo abordaremos questões históricas e motivacionais.

O desenvolvimento de projetos de software não é uma tarefa fácil. Um projeto deve atender aos seguintes requisitos: têm que ser robustos; atuar sobre problemas complexos; estar implantados em curto prazo (“time-to-market”).

Na década de 60 com a existência de hardwares mais poderosos, o software ficou mais complexo. Como não existia uma sistematização de técnicas de desenvolvimento aconteceu a Crise do Software

Comparando a Engenharia de Software com outras engenharias, ela é uma disciplina nova que surgiu em 1968, em uma conferência organizada para discutir a crise do software. Percebia-se que o desenvolvimento informal de software não era eficiente.

Mesmo com a evolução da Engenharia de Software será que já saímos totalmente da crise do software?

A Orientação a Objetos tem uma importância relevante neste contexto pois é utilizada em boa parte das fases de um Processo de Desenvolvimento

As grandes fases de qualquer processo de desenvolvimento LEVANTAMENTO DE REQUISITOS ANÁLISE PROJETO IMPLEMENTAÇÃO TESTES

É importante ressaltar que a Orientação a Objetos beneficia principalmente projetos de grande porte com muitas linhas de código e com a necessidade de serem mantidos e evoluídos.

Coesão

O paradigma da Orientação Objetos busca uma organização semântica do projeto para obter ma boa delegação de responsabilidade, um baixo acoplamento e uma redução de complexidade, ou seja, um software coeso.

Um pouco de história

Métodos de modelagem Orientados a Objetos

começaram a aparecer entre meados da década 70 e início dos anos 80. Os primeiros registros da concepção dos conceitos de Orientação a Objetos surgiram com a linguagem Simula na década de 60, na Noruega, no centro de Norweigan Computin Center (NCC), sobre a responsabilidade de Kristen Nygaard e Ole-Johan Dahl. O cerne dessa linguagem era a criação de um modelo computacional que fosse similar às estruturas do mundo real, facilitando, assim, o projeto de sistemas. Após a linguagem Simula, um outro projeto a implementar os conceitos de Orientação a Objetos muito conhecido foi a linguagem Smalltalk, criada pelo núcleo de pesquisa da Xérox PARC e desenvolvida pelos pesquisadores Adele Goldberg, Alan Key e Dan Ingalls. Estes métodos e linguagens, inicialmente, permearam o meio acadêmico em pesquisas pouco aplicáveis. Outra linguagem importante na história da Orientação a Objetos foi a linguagem C++. Esta é uma evolução da linguagem C que está sobre o paradigma Estruturado Funcional. A primeira versão foi desenvolvida em 1980 com o nome de 'C with Classes'. Em 1983 é lançada a primeira versão do C++ com diversas características de Orientação a Objetos. O pesquisador responsável por essa evolução da linguagem C foi Bjarne Stoustrup. Apesar da linguagem C++ ser poderosa, alguns pesquisadores e professores têm restrições quanto ao primeiro contato do aluno, com os conceitos de Orientação a Objetos, ser com a linguagem C++. Um dos principais motivos dessa restrição deve-se ao fato do C++ ser uma linguagem hibrida. Como foi dito, o C++ é uma evolução da linguagem C que está sobre o paradigma Estruturado Funcional, permitindo que o aluno programe desrespeitando os conceitos da Orientação a Objetos. Já a linguagem Java surgiu no ano de 1991, projetada por James Gosling, Bill Joy e Guy Steele na Sun Microsytems. O seu projeto inicial visava ser uma linguagem para programação de sistemas embarcados. Estes programas rodariam em dispositivos eletrônicos da Sun Microsystem. Com o surgimento da Internet e da WWW (World Wide Web), a linguagem Java começou a ser amplamente utilizada. O sucesso da linguagem está associado com o recurso de independência de plataforma e por ser uma linguagem parecida e mais simples que C e C++, visto que a alocação e a liberação de memória não são feitos diretamente pelo programador. Portanto, Java possui um mecanismo de gerência de memória e coleta de lixo que evitam erros de manipulação de memória. Existem outras linguagens de programação, porém menos conhecidas.

Ferramentas RAD

Na década de 90, a aplicação comercial dos conceitos da Orientação a Objetos se for taleceu, saindo do meio acadêmico e sendo aplicados nas organizações no desenvolvimento de suas soluções de software comerciais. Uma das causas dessa popularização foi o aparecimento das ferramentas RAD (Rapid Aplication Development), que objetivavam aumentar a produtividade na construção de software. Estas ferramentas eram aplicadas, principalmente, na construção de elementos da camada de interface, tais como menus, botões, telas, tabelas, combos, etc. Os mecanismos de construção de aplicações para o desenvolvedor na ferramenta RAD se dá através

da reutilização de componentes visuais, previamente construídos, que são manipulados e customizados para atender aos requisitos específicos da aplicação que está sendo construída. Um exemplo clássico de ferramentas RAD são as IDE Visual Basic , Delphi, Centura etc.

Apesar dos benefícios alcançados com a produtividade e facilidade no desenvolvimento de software, diversos problemas relacionados com a construção de aplicações baseados em RAD são identificados. Uma primeira questão a ser analisada é que um software normalmente possui um custo muito mais alto de manutenção do que de desenvolvimento. Como as ferramentas RAD e as técnicas de Engenharia de Software difundidas na década de 90, os desenvolvedores acabavam construindo softwares com um forte acoplamento entre a camada de aplicação e a camada de interface . Este tipo de solução gerava graves problemas de acoplamento e baixa reutilização dificultando a manutenção corretiva e evolutiva das aplicações. Por exemplo, o espalhamento de regras de negócio nas telas exige uma mudança em vários pontos do software.

UML

Na Ciência da Computação, a modelagem Orientada a Objeto se popularizou na década de 90 com o sucesso da linguagem de notação UML (The Unified Modeleing Language) que unificou as melhores práticas da OMT (Object Modeling Technique) proposta por Booch, Rumbough e modelo OOSE (Object-Oriented Software Engineering) proposta por Jacobson. A UML definida como uma linguagem unificada para modelagem gráfica de objetos foi um marco na Engenharia de Software, pois antes do seu surgimento cada autor de referência na área tinha a sua forma de modelagem específica com alguns poucos elementos em comum. A UML tem uma grande aceitação na Engenharia de Software e já extrapola o

A Orientação a Objetos está ainda em plena evolução, apesar de ser um paradigma de desenvolvimento da década 1970. Exemplo são as técnicas de desenvolvimento baseadas em: Framework, Padrões de Projeto, Orientação a Aspectos, TDD (Test Driven Development), BDD (Behavior Driven Development) entre outras que agregam novas possibilidades a Orientação a Objetos.

Abstração

Um campo de pesquisa na Ciência da Computação que buscou formas de materialização de conceitos foi a Orientação a Objetos, pensada por Allan Key. Os métodos de modelagem Orientados a Objeto são orientados pela estratégia “top-down” e começaram a surgir entre os meados da década 70 e início dos anos 80. Sua base teórica foi influenciada por alguns conceitos presentes na Rede Semântica (é-um e parte-de).

Os autores Rumbaugh e Blaha discorrem sobre a relação entre a abstração e o modelo, enfatizando o alcance da redução de complexidade no processo de modelagem. Assim, nos seus textos os autores afirmam que: "Como um modelo omite os detalhes não-essenciais, sua manipulação é mais fácil do que a da entidade original.". Em seguida os autores ressaltam a importância da abstração visto que, segundo os mesmos, "A abstração é uma fundamental capacidade humana que nos permite lidar com coisas complexas.". A aplicação da abstração para compreensão e representação da realidade é tão importante que Segundo Carvalho, Monteiro e Genaro explicam: A noção de modelo abstrato precede qualquer consideração teórica. No entanto pode ser considerada como uma visão a respeito de um determinado estado de coisas. Tais visões não pretendem, em geral, fazer descrições completas da realidade, pois na própria idéia está implícita a de simplificação ou abandono de detalhes irrelevantes, mesmo que posteriormente tais visões sejam refeitas ou reavaliadas, fazendo com que certos detalhes mudem de status e passem a ser considerados O final desta citação pondera-se uma das questões da maior relevância que é a dinâmica da compreensão de uma realidade que pode ser sempre reavaliada, realimentando o modelo originalmente pensado.

Bons Estudos e Realize os exercícios para reter o que foi abordado neste módulo