Basico de Orientação a Objetos

O que é uma classe? A melhor definição, segundo Profº Rodrigo Rocha é “uma representação de características ou de formas de um grupo de entidades, ou de um conceito pela qual nós acreditamos que existe”. Alguns autores definem como uma “abstração de uma representação genérica de um grupo de objetos com características ou atributos em comum e comportamentos”.

Existem dois tipos de classes:

1 – Concretas: aquelas que podem ser instanciadas para utilizar seus métodos;

2 – Abstratas: Aquelas que nunca poderão ser instanciadas mas que podem ser utilizadas como modelo das classes que irão herdá-las;

Um conceito muito utilizado em projetos de engenharia de software com orientação a objetos é herança. Só faz sentido dizer que existe um relacionamento de herança entre dois grupos de entidades ou classes, quando uma delas **é** outra. Ex: uma classe de **carro é** um **automóvel**.

Outro conceito é associação, que são 3 tipos. Isso ocorre quando uma classe **tem** outra. Ex: Uma **concessionária tem** um ou mais **carros**.

Na orientação a objetos devemos trabalhar muito com a semântica, ou seja, qual o sentido das classes terem tais atributos e métodos. Tudo vai depender do que tal sistema ou software vai tratar. Só é possível definir as classes com a semântica correta quando conhecemos muito bem os requisitos da aplicação. Portanto é necessário entender bem o negócio do sistema e virar um profissional de uma determinada área.

Exemplo: em um projeto para a implementação de um módulo do SISCOMEX (sistema de comércio exterior da receita federal), chamado NCM (módulo responsável pelo cadastro de produtos chamado Nomenclatura Comum do Mercosul), um sistema numérico que define a classificação de qualquer produto que seja

comercializado no mercosul. Em 2006 haviam 15000 dessas NCMs definidas e em cima disso é que a RF faz a incidência de imposto de acordo com sua classificação.

A má interpretação de um negócio poderá afetar todo o sistema ou software. Portanto o ideal para que se desenvolva qualquer sistema, é que a pessoa já conviva com tal negócio e seja um especialista nisso. Ex: um sistema de gerenciamento hospitalar, que fosse desenvolvido por um médico.

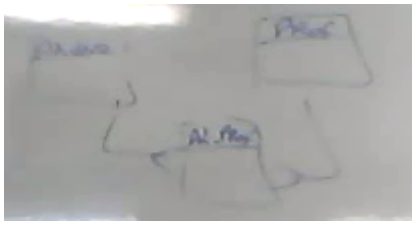
Existem duas formas mais comuns para criar documentações que facilitem a interpretação do sistema. Uma delas é por diagramas e a outra por texto, onde poderá ser descrito de forma mais detalhada. Um diagrama de classe representa um **domínio de negócio**. Ex: aluno, professor, disciplina, prova, etc representam domínio de um negócio escolar, ou seja, termos que compõe a descrição do domínio. Utilizar um diagrama de classe para representar um domínio será fundamental para fazer a construção correta de um sistema para conseguir atender os requisitos que um sistema impõe. Para isso é necessário o pleno entendimento dos requisitos e regra de negócio para conseguir criar um diagrama de classes.

Um objeto é uma instancia de uma classe, que possui uma identidade, características (que são compartilhados entre seus pares) e ações ou comportamentos. A identidade e características são os atributos e as ações e comportamentos associados aos métodos. Portanto, podemos dizer que uma classe é a representação de um conjunto de objetos.

Um exemplo de representação em um diagrama de classe uma regra de negócio é o relacionamento de composição entre venda e forma de pagamento, ou seja, uma forma de pagamento (parte) precisa de unicamente um objeto de venda para existir.

Quando tratamos de BD, a orientação a objetos demonstra menor limitação que o modelo relacional. Por exemplo: a orientação a objetos, na modelagem UML, podemos provar mais claramente que, num diagrama de classe com relacionamento entre aluno e professor, ambos podem ter várias instâncias da outra. Já no

modelo relacional, deve haver uma tabela para cada e criar mais uma tabela intermediária em que deverá ter o relacionamento entre as classes.



A representação de classes é uma implementação do paradigma de orientação a objetos. Um SGBD é uma implementação do conceito relacional. Este último não é semântico, pois não olha o problema. Pode-se entender que o modelo relacional funciona conforme os requisitos não funcionais, ou seja, sua estrutura está muito mais direcionado para os requisitos não funcionais por conta de desempenho, tempo de resposta, armazenamento, etc, enquanto que a orientação a objetos está para os requisitos funcionais e regras de negócio.