CURSO: Ciência da Computação	**DATA: ____/____/ 2013**
PERÍODO: 4º.	**PROFESSOR: Andrey**
DISCIPLINA: Técnicas Alternativas de Programação	**AULA: 11**

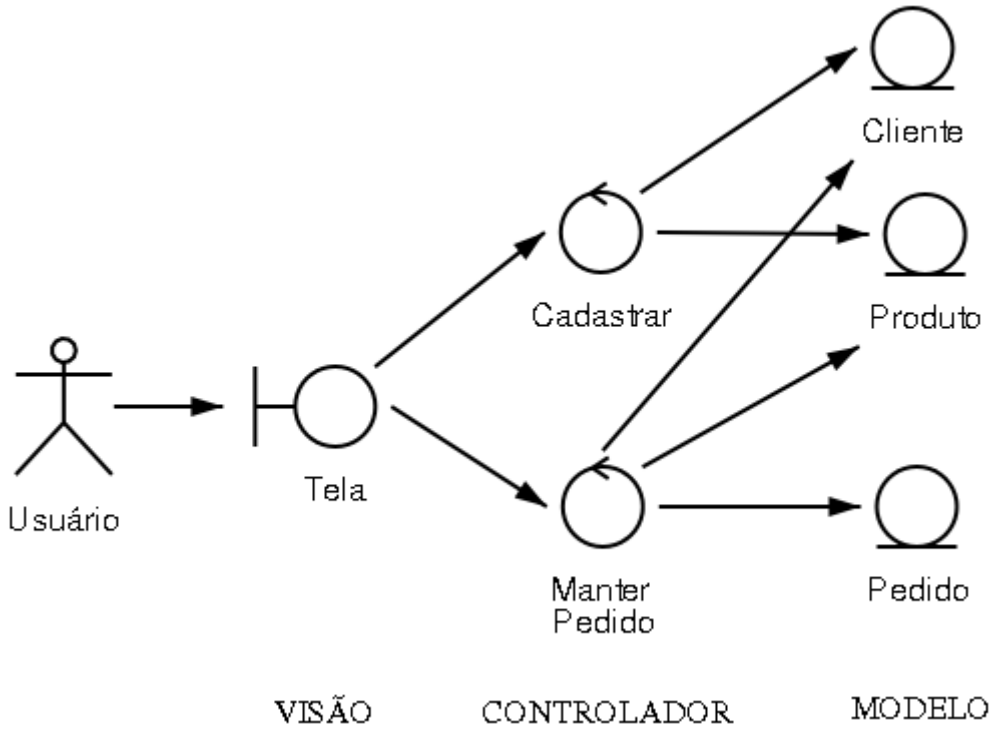
APRESENTAÇÃO

Nesta aula serão discutidos os conceitos relacionados aos modelos de Análise Orientada a Objetos.

DESENVOLVIMENTO

Modelo-Visão-Controlador

A arquitetura de aplicações Modelo-Visão-Controlador (MVC) é usada para analisar algumas características de aplicações distribuídas.
Esta abstração nos ajuda a particionar mais facilmente, uma aplicação em componentes lógicos.
A arquitetura MVC é uma forma de dividir funcionalidade entre objetos envolvidos em manter e apresentar dados de tal forma a minimizar o grau de acoplamento entre os objetos.
A arquitetura MVC foi originalmente desenvolvida para mapear as tarefas do processo tradicional entrada-processamento-saída para o modelo de interface gráfica com o usuário. Esta arquitetura (MVC) foi desenvolvida pela Xerox (PARC) e usada pela então recém criada linguagem Smalltalk 80.
Nesta aula vamos apresentar uma variação do MVC original que é usado por metodologias conhecidas na modelagem de sistemas, representada na figura abaixo:



Modelo

Na arquitetura MVC, o Modelo representa os dados da aplicação e as informações do negócio. Frequentemente o Modelo serve como uma representação na forma de classes das entidades do negócio. Normalmente, nas classes do Modelo vamos ter os acessos aos dados da base de

dados. As classes que compõem o Modelo são as chamadas classes persistentes. As classes do modelo, também chamadas classes entidade terão as responsabilidades de inserir, atualizar e remover a informação e mantê-la atualizada na base de dados.

Visão
A Visão representa a interface com o usuário, como as telas do sistema. As classes da Visão irão receber os comandos do usuário e repassá-los para o controlador e receber os resultados do controlador e mostrá-los ao usuário. As classes da Visão, também chamadas de classes limite, terão apenas as responsabilidades de entrada de dados e exibição de resultados.

Controlador
O Controlador representa os processos que existem no sistema e as regras do negócio para manipular a informação. São as classes que irão controlar os processos no sistema. O Controlador existe para separar o Modelo da Visão. Permitindo que pequenas mudanças na interface não afetem o Modelo e vice-versa. As classes do Controlador terão as responsabilidades de conduzir o processo, pedindo informações para o modelo, realizar as modificações e enviar os resultados para a Visão.

Modelo de Análise Orientados a Objetos

O modelos de Análise Orientada a Objetos são divididos em:

- Estruturais
- Comportamentais

O principal modelo estrutural é o modelo de classes. Este modelo é composto pelo diagrama de classes de análise.

Classes de Análise

As classes de análise são óbvias no contexto do domínio do problema. São mais conceituais e de granularidade maior do que as classes de projeto e implementação.

- definem responsabilidades – dificilmente incluem operações.
- definem atributos, em um nível mais alto nível. Os atributos são conceituais e reconhecíveis no domínio do problema. Podem se tornar classes no projeto e implementação.
- São inicialmente obtidas a partir do modelo de objetos de negócios.

O modelo de classes de análise é composto pelos objetos identificados na análise do domínio e da aplicação. Seu objetivo é descrever o problema representado pelo sistema a ser desenvolvido sem levar em consideração características da solução a ser desenvolvida.

As Classes de análise são modeladas em termos de estereótipos. Não existe um livro de receitas que ensine como descobrir classes. O RUP (Rational Unified Process) sugere que as classes de análise sejam descobertas no desenvolvimento do sistema, procurando as classes de limite, controle e entidade. Estes três estereótipos ajustam-se, ponto de vista *model-view-controler* e permitem ao analista particionar o sistema separando a visão do domínio do controle necessário pelo sistema.
Tendo como base, que a análise e o projeto do processo são interativos, a lista de classes irá mudar ao longo do tempo. O conjunto inicial de classes provavelmente não será o conjunto de classes que serão efetivamente implementadas. Para nós, o termo candidato para uma classe é freqüentemente usado para descrever o primeiro conjunto de classes descobertas para o sistema.

Classes Entidade

Uma classe entidade modela informação e comportamento associado que geralmente tem uma vida longa. Este tipo de classe pode refletir uma entidade do mundo real, ou ela pode ser necessária para executar uma tarefa interna do sistema. Elas são tipicamente independentes do

meio em que estão; isso é, elas não precisão saber como as classes que estão no limite se comunicam com o sistema. Muitas vezes, elas são aplicações independentes, isso significa que muitas delas poderão ser utilizadas por mais de uma sistema.
O primeiro passo é examinar as responsabilidades documentadas no fluxo de eventos para o caso de uso identificado (i.e., o que o sistema deve fazer). Classes entidade são normalmente utilizadas pelo sistema para definir alguma responsabilidade. Os nomes e as frases usadas para descrever a responsabilidade podem ser um bom ponto de partida. A lista inicial de nomes deve ser filtrada, pois ela pode conter nomes que estão fora do domínio do problema, nomes que são somente expressão de linguagem, nomes que são redundantes, e nomes que são descrições da estrutura da classe.
Classes entidade são normalmente encontradas na Fase de Elaboração. Elas são freqüentemente chamadas de classes de domínio, desde que elas usualmente tratam com abstrações de entidades do mundo real.

Classes Limite

Classes Limite cuidam da comunicação entre meio com o qual o sistema interage e o sistema propriamente dito. Elas fornecem a interface para um usuário ou para um outro sistema (i.e., a interface para um ator). Elas constituem a parte do sistema que dependem do meio em que elas estão. Classes limite são utilizadas para modelar as interfaces do sistema.
Cada par ator/cenário é examinado para descobrir as classes limite. As classes limites escolhidas na Fase de Elaboração do desenvolvimento são tipicamente de alto nível. Por exemplo, você pode modelar uma janela mas não modela cada caixa de diálogo e botões. Neste ponto, você está documentando os requerimentos da interface com o usuário, não implementando a interface.
Os requerimentos das interfaces com o usuário tendem a ser muito vagos - os termos amigáveis e flexíveis são muito utilizados. Mas amigável, tem significados diferentes para pessoas diferentes. Neste caso as técnicas de prototipagem podem ser muito úteis. O cliente pode dar uma olhada e sentir o sistema e realmente entender o que o termo amigável significa. O "*o que*" é então compreendido assim como a estrutura e o comportamento da Classe Limite. Durante o projeto, essas classes são refinadas para levar em consideração os mecanismos da interface escolhida.
Classes Limite são também adicionadas para facilitar a comunicação com outros sistemas. Durante o projeto, estas classes são refinadas para levar em consideração o protocolo de comunicação escolhido.

Classes Controle

Classes Controle modelam uma seqüência de comportamento específico a um ou mais casos de uso. Classes Controle coordenam os eventos necessários para a realização do comportamento especificado em um caso de uso. Você pode pensar que uma Classe Controle como "rodando" ou "executando" o caso de uso - ela representa a dinâmica do caso de uso. Estas classes normalmente são classes dependentes da aplicação.
Logo nos primeiros estágios da fase de Elaboração, uma Classe Controle é adicionada para cada par Ator/Caso de Uso. A Classe Controle é responsável pelo fluxo de eventos do caso de uso.
O uso de Classes Controle é muito subjetivo. Muitos autores sentem que o uso de Classes Controle resultam no comportamento sendo separado dos dados. Isso pode acontecer se a Classe Controle não foi escolhida prudentemente. Se uma Classe Controle está fazendo mais do estabelecer uma seqüência, então ela está fazendo coisas demais. Por exemplo, no sistema de matrículas, um estudante seleciona um curso ofertado, se o curso ofertado está disponível, o estudante é matriculado nele. Quem sabe como matricular um estudante - a Classe Controle ou a *OfertaCurso*? A resposta correta é, *OfertaCurso*. A Classe Controle sabe quando o estudante deveria ser matriculado, o curso ofertado sabe como matricular o estudante. Uma péssima Classe Controle não saberia só quando matricular o estudante mas como matricular o estudante.

A adição de uma Classe Controle para cada par Ator/Caso de Uso é somente um começo - enquanto a análise e o projeto continuam, as Classes Controle podem ser eliminadas, explodidas ou combinadas.

Objetos e Classes no problema do Sistema de Matrícula (MATRI)

Vamos dar uma olhada no cenário *Adicionando uma Oferta de Curso para Lecionar*, o qual é um dos subfluxos do caso de uso *Selecionar Cursos para Ministrar* (ver apostila anterior, sobre Comportamento do Sistema). A principal definição especificada neste cenário é a habilidade do professor selecionar um curso ofertado para lecionar em um dado semestre.
Embora nós estejamos olhando este processo passo a passo, muitos desses passos podem ocorrer ao mesmo tempo no mundo real.

Identificando Classes Limite

Este caso de uso interage somente com o ator Professor. A ação especificada neste cenário é somente uma das definidas no caso de uso (o caso de uso também afirma que o Professor pode modificar, excluir, rever e imprimir a seleção). Isto significa que alguma coisa no sistema deve prover ao Professor a capacidade de selecionar a sua opção. Uma classe contém todas as opções disponíveis ao Professor como está registrado no caso de uso, para satisfazer a especificação feita. Esta classe é chamada Tela*OpçõesCursoProfessor*. Adicionalmente podemos identificar uma classe que faça a adição de um novo Curso Ofertado para o Professor. Esta classe é chamada Tela*AdicionaOfertaCurso*.

Identificando Classes Entidade

Este cenário está relacionado com Cursos, com as Ofertas de Curso e com o Relacionamento do Curso com o Professor. Nós podemos identificar três classes entidade: *Curso*, *OfertaCurso* e *InformaçãoProfessor*.

Identificando Classes de Controle

Iremos adicionar uma classe de controle para manusear o fluxo de eventos para o caso de uso. Esta classe é chamada *ControleCursoProfessor*.
As classes identificadas foram adicionadas ao modelo.

MODELO CARTÕES CLASSE-RESPONSABILIDADES-COLABORADORES

Uma forma para modelar um problema em classes e objetos é a forma proposta no artigo "A Laboratory For Teaching Object-Oriented Thinking", por ser simples e fácil de ser aplicada. São os Cartões Classe-Responsabilidade-Colaboradores (CRC). Os cartões CRC foram usados por algumas metodologias de desenvolvimento como a apresentada por Wirfs-Brock, Wilkerson e Wiener no livro "Designing Object-Oriented Software".

É baseada nos três atributos principais de um objeto na fase de projeto: nome da classe, suas responsabilidades e seus colaboradores.

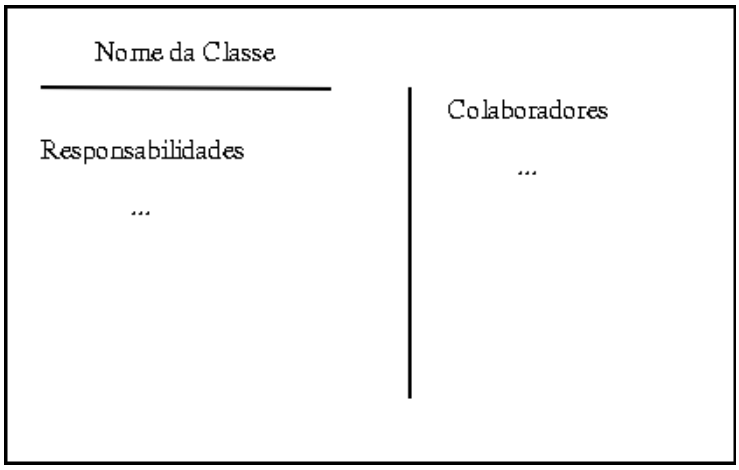
O nome da classe deve descrever o objeto no contexto geral do ambiente. Deve-se tomar um certo cuidado e gastar um pouco de tempo para escolher o conjunto certo de palavras para descrever o objeto.

As responsabilidades devem identificar os problemas a serem resolvidos e não as soluções. Identificando-se os problemas fica fácil escolher entre as soluções possíveis. Novamente deve-se tomar cuidado com o conjunto de palavras usadas. As frases devem ser curtas e concisas.

Os colaboradores de um objeto são os objetos para os quais este irá enviar mensagens a fim de satisfazer as suas responsabilidades.

Para facilitar a modelagem, Cunningham inventou os cartões Classe-Responsabilidades-Colaboradores (CRC). Esses cartões servem, também, como meio de

documentação do projeto. Os cartões CRC são pedaços de papel grosso medindo 10x15cm e contém o nome da classe, as responsabilidades e os colaboradores, como na figura.



A disposição espacial dos cartões é importante. Quando duas classes são mútuas colaboradoras, seus cartões devem ser colocados levemente sobrepostos. Quando uma classe é colaboradora de outras classes (a classe é usada pelas outras), seu cartão deve ficar abaixo dos cartões das outras classes. Se há uma relação de herança, deverá haver uma sobreposição quase completa dos cartões. A disposição dos cartões é importante, pois em projetos ainda incompletos é possível notar a localização de uma classe antes da mesma ter sido projetada.

Para achar as classes, sua responsabilidades e seus colaboradores deve-se seguir as seguintes etapas:

1. ·Definição das classes e das estruturas de dados de cada classe;
2. ·Definição das responsabilidades de cada classe;
3. ·Definição dos colaboradores de cada classe;

1 - O que são classes e como defini-las;

Num problema a ser resolvido, as diversas classes podem ser identificadas como as entidades que existem no problema, por exemplo, aluno, turma, professor, etc... São classes, também, as estruturas de dados utilizadas para resolver o problema, bem como os dispositivos (físicos e virtuais) a serem acessados. Por exemplo: interface, arquivo, controlador de tempo, etc...

2 - O que são responsabilidades e como defini-las;

As responsabilidades são definidas como sendo os problemas que as classes devem resolver. São, a grosso modo, as funções das classes. Por exemplo: colocar um registro no arquivo, dar presença ao aluno, cobrar mensalidade, etc... As responsabilidades são os problemas a serem resolvidos. A maneira de resolve-los vão ser os métodos das classes(implementação).

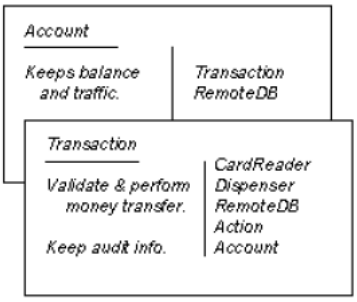
3 - O que são colaboradores e como defini-los;

Os colaboradores são as classes que ajudam uma classe a resolver as suas responsabilidades. São as classes que vão prestar serviços à classe. A classe que está sendo analisada solicita serviços a outras classes e estas, por sua vez, prestam estes serviços, ajudando a resolver os problemas.

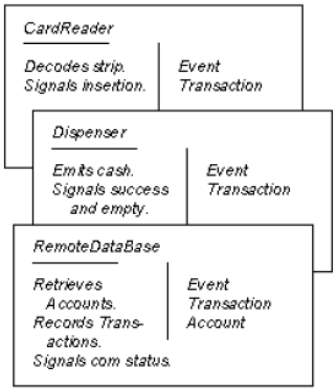
Exemplo de Modelagem usando cartões CRC

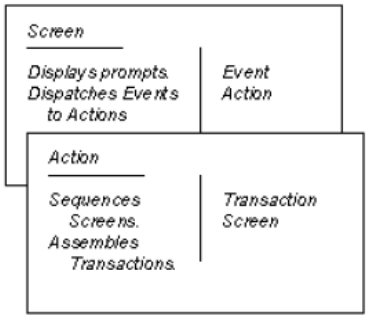
Modelagem do software de um Caixa Automático de Banco (ATM) (BECK e CUNINGHAM, 1989)

Classes do Modelo

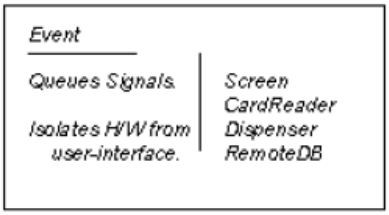


Classes Limite ou Fonteira



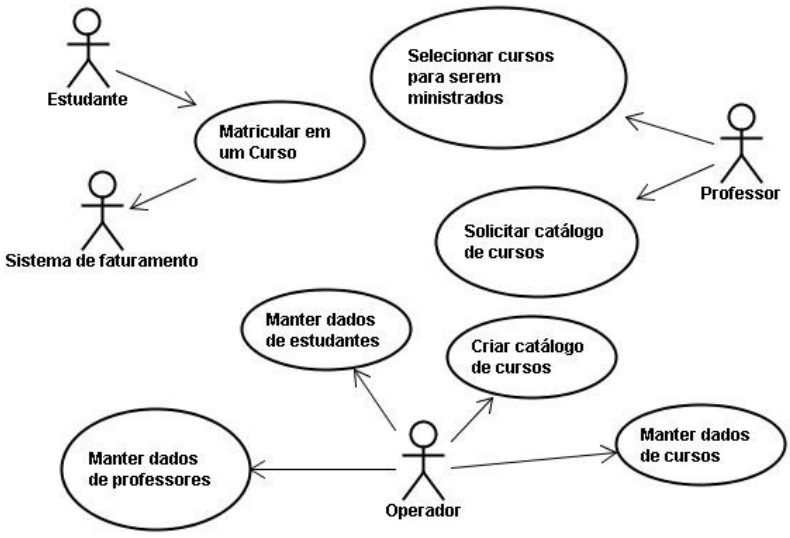


Classe Controle



ATIVIDADE

1. Baseando-se no modelo de Caso de Uso abaixo, tente levantar as classes para o sistema de Matrícula (MATRI).



2. modelar o problema de controle de uma pizzaria de entrega em domicílio. O cliente pede uma ou mais pizzas, refrigerantes ou cervejas. O atendente anota o pedido no sistema. Cada pizza pode ser (média, grande ou enorme). Cada pizza pode ter no máximo 3 sabores. O atendente anota os dados do cilente e a forma de pagamento (inclusive o troco necessário). O atendente anota o código do entregador e o tempo de entrega. O gerente cadastra os sabores de pizzas existentes, os dados dos entregadores e visualizaos relatórios de vendas(por sabor, região) e tempo de entrega.

BIBLIOGRAFIA BÁSICA

PRESSMAN, R. S.. Engenharia de Software. Makron Books. 1995

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I.. UML guia do usuário. Editora Campus. 2000.

BEZERRA, E.. Princípios de Análise e Projeto de Sistemas com UML. Editora Campus. 2003.

CARVALHO, A. M. B. R.; CHIOSSI, T. C. S.. Introdução à engenharia de software. Editora da Unicamp. 2001.

BECK, K e CUNINGHAM, W.. A Laboratory For Teaching Object-Oriented Thinking. OOPSLA'89 Conference Proceedings October 1-6, 1989, New Orleans, Louisiana and the special issue of SIGPLAN Notices Volume 24, Number 10, October 1989.