PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Modelagem de Software Orientado a Objetos

Apresentação da disciplina

Módulo I – 2º semestre de 2011

Prof. Dr. Maurício Nacib Pontuschka
tuska@pucsp.br

Modelagem de Software
Orientado a Objetos

Apresentações iniciais

Prof. Maurício Nacib Pontuschka

- Bacharel em Ciência da Computação – PUC-SP
- Mestre em Engenharia Elétrica – Mackenzie
- Pós Graduado pelo MBIS –
  Master Business Information Systems – PUC-SP
- Doutor em Comunicação e Semiótica – PUC-SP
- Professor e Coordenador do Curso de
  Ciência da Computação da PUC-SP
- Consultor nas áreas de Desenvolvimento de Sistemas e Gestão de Projetos de TI, Business Games e Processamento de Imagens

Modelagem de Software Orientado a Objetos

Apresentações iniciais

- Sua empresa
- Seu cargo
- Sua experiência
  - experiência com tecnologia de objetos
  - experiência em desenvolvimento de software
- Suas espectativas com esta disciplina

Modelagem de Software Orientado a Objetos

Público alvo

- Graduados nas áreas de Computação, Sistemas de Informação, Análise de Sistemas e Tecnologia da Informação.
- Desenvolvedores de software interessados em modelagem visual de sistemas.
- Gerentes de desenvolvimento que desejam entender melhor a tecnologia de objetos.
- Desejável que conheça e tenha alguma experiência em programação de computadores.

Modelagem de Software Orientado a Objetos

PUC • SP COGEAE

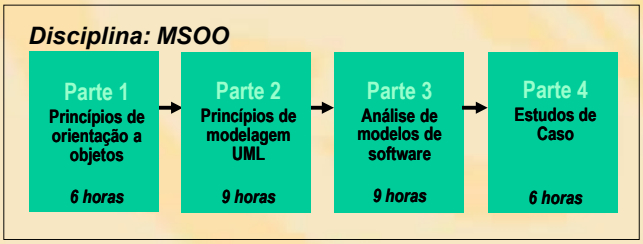
Objetivos previstos na disciplina MSOO

- Definir o histórico e a aplicação atual da tecnologia de objetos.
- Explicar o que a UML representa.
- Explicar abstração, encapsulamento, modularidade e hierarquia.
- Descrever a estrutura física de uma classe.
- Identificar o relacionamento entre objetos e classes.
- Definir polimorfismo e generalização.

Modelagem de Software Orientado a Objetos

PUC • SP COGEAE

Estrutura da disciplina



Modelagem de Software Orientado a Objetos

PUC • SP COGEAE

CRAIG LARMAN, “Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development”,Bookman, 3rd Edition, 2007.

ERIC FREEMAN, ELISABETH FREEMAN, “Use a Cabeça - Padrões de Projetos”, Alta Books, 2005.

Modelagem de Software Orientado a Objetos

PUC • SP COGEAE

ERIC GAMMA, RICHARD HELM, RALPH JOHNSON, JOHN VLISSIDES, “Design Patterns”,Addison Wesley, 1995.

GRANDY BOOCH, IVAR JACOBSON, JAMES RUMBAUGH “Uml Guia do Usuario”, Campus, 2006.

Modelagem de Software Orientado a Objetos

ALISTAIR COCKBURN “Surviving Object-Oriented Projects”, Addison Wesley, 1998.

BRUCE F. WESTER "Pitfalls of Object-Oriented Development",M&T Books, 1995.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Modelagem de Software Orientado a Objetos

Parte 1 – Princípios de Orientação a Objetos

Introdução

Prof. Dr. Maurício Nacib Pontuschka
tuska@pucsp.br

Tecnologia de Objetos

O que é a tecnologia de objetos?

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

Tecnologia de Objetos

"Um conjunto de princípios utilizado na construção de software, em conjunto com linguagens, bancos de dados e outras ferramentas que suportam estes princípios."

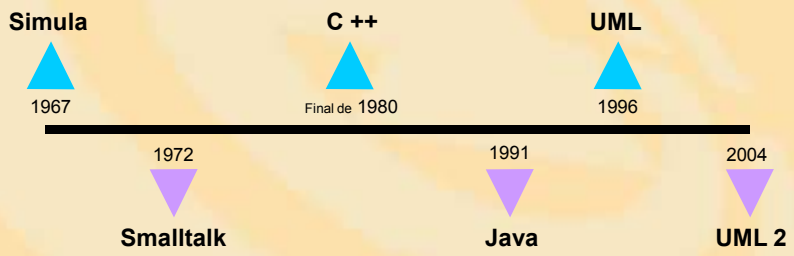
Object Technology – A Manager's Guide, Taylor, 1997.

Vantagens da Tecnologia de Objetos

- Reflete um único paradigma
- Facilita o reuso de arquitetura e de código
- Possui modelos mais próximos do mundo real
- Oferece uma maior estabilidade
- É suscetível a mudanças

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

Marcos da Tecnologia de Objetos



Tecnologia de Objetos

Onde a tecnologia de objetos é utilizada atualmente?

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

Sistemas cliente/servidor e Desenvolvimento WEB

A tecnologia de objetos permite as empresas a encapsular as informações de negócios em objetos e ajuda a distribuir o processo por meio da Internet ou uma rede convencional.

Sistemas em tempo real

A tecnologia de objetos permite o desenvolvimento de sistemas em tempo real de maior qualidade e flexibilidade.

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

Orientação a objetos

- Une dados e os processos de fluxo de dados nos primeiros momentos do ciclo de vida de desenvolvimento.
- Possui um alto nível de encapsulamento.
- Promove o reuso de código.
- Permite uma maior abrangência do software.

Referências

IBM Software Group, "***Essentials of Visual Modeling with UML 2.0***".

Rational Web site
http://www-306.ibm.com/software/rational/

Rational developerWorks
http://www-136.ibm.com/developerworks/

UML Resource Center
http://www-306.ibm.com/software/rational/uml/

Rational Edge
http://www-106.ibm.com/developerworks/rational/rationaledge/

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

Dúvidas?

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Modelagem de Software Orientado a Objetos

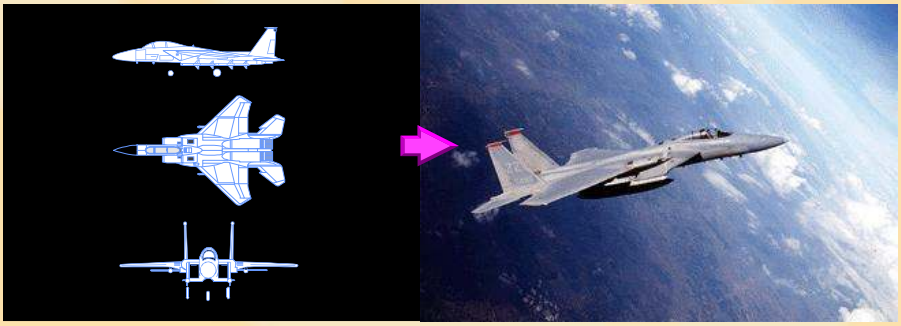
Parte 1 – Princípios de Orientação a Objetos

Continuação

Prof. Dr. Maurício Nacib Pontuschka
tuska@pucsp.br

O que é modelagem?

Um modelo é uma simplificação da realidade.



Objetivos da modelagem

- Ajudar a visualizar o sistema como gostaríamos que ele fosse
- Permitir a especificação da arquitetura e comportamento de um sistema
- Fornecer padrões de desenvolvimento que serve como guia de construção do sistema
- Documentar as decisões tomadas durante o processo de desenvolvimento

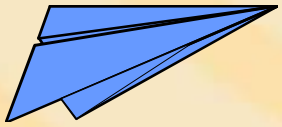
Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

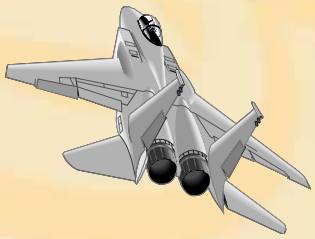
Por que modelar?

- Modelos de sistemas complexos são construídos porque nem sempre é possível compreendê-los em sua totalidade
- Modelos auxiliam o entendimento dos sistemas que construímos

Importância da modelagem

Menos importante — Mais importante





Avião de papel

Jato militar

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

Muitas equipes de desenvolvimento desenvolvem seus sistemas como se estivessem construindo aviões de papel

- Iniciam a codificação diretamente a partir das especificações do projeto.
- Trabalham durante horas e produzem mais código.
- Falta de um plano de arquitetura de software.
- Desenvolvimento fadado ao fracasso.

Por que modelar?

- A modelagem é uma tarefa comum em projetos bem sucedidos.

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

Model Driven Architecture (MDA)

- Uma abordagem de utilizar modelos no desenvolvimento de software.
  - Separar a especificação de uma operação dos detalhes da forma como o sistema utiliza os recursos da plataforma utilizada.
    - Especificar um sistema independentemente da plataforma que o suportará.
    - Especificar plataformas.
    - Escolher uma determinada plataforma para um sistema.
    - Transformar a especificação de um sistema específica para uma determinada plataforma.

Pontos de visão do MDA

CIM – Computational Independent Model

O foco está no ambiente do sistema e seus requisitos.

PIM – Platform Independent Model

O foco está na operação do sistema, independentemente da plataforma.

PSM – Platform Specific Model

O foco está na utilização detalhada do sistema em uma plataforma específica.

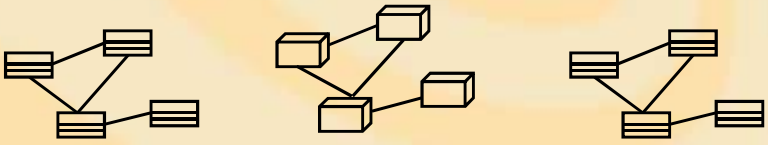
Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

Quatro princípios da modelagem visual

- O modelo que é criado influencia na forma como o problema é atacado.
- Todo modelo pode ser expresso em vários níveis de precisão.
- Os melhores modelos são os mais próximos à realidade.
- Nenhum modelo único é suficiente.

Principio 1: A escolha do modelo é importante

- Os modelos criados influenciam profundamente em como o problema é atacado e como a sua solução é elaborada.

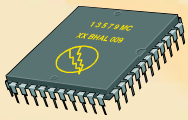


Modelo de Processo | Modelo de Implantação | Modelo de Projeto

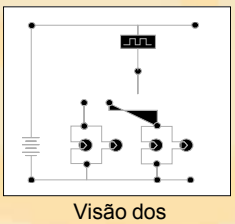
Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

Principio 2: Níveis de precisão podem ser diferenciados

- Todo modelo pode ser expresso em diferentes níveis de precisão.
  - Quem visualiza o modelo e por que precisam visualizá-lo?



Visão dos consumidores



Visão dos desenvolvedores

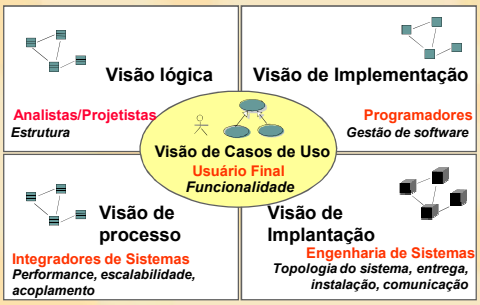
Principio 3: Os melhores modelos estão ligados à realidade

- Todos os modelos são simplificações da realidade.
- Um bom modelo reflete características reais.

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

Principio 4: Nenhum modelo único é suficiente

- Todo sistema (não trivial) é melhor abordado por um conjunto de modelos.



Próximos objetivos

- Descrever uma abstração, encapsulamento, modularidade e herança.
- Descrever a estrutura física de uma classe.
- Descrever o relacionamento entre uma classe e um objeto.
- Definir polimorfismo e generalização.

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

- O que é um objeto?
- Quatro princípios da OO
- O que é uma classe?
- Polimorfismo e Generalização
- Organizando elementos de modelo

Informalmente, um objeto representa uma entidade que pode ser física, conceitual ou de software.

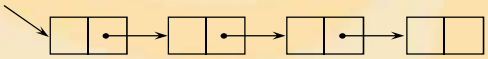
Entidade física

Caminhão

Entidade conceitual

Processo químico

Entidade de software

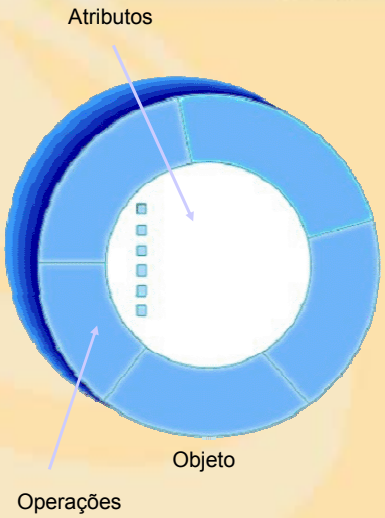


Lista ligada

Em uma definição mais formal:

Um objeto é uma entidade com uma fronteira bem definida e uma *identidade* que encapsula *estados* e *comportamento*.

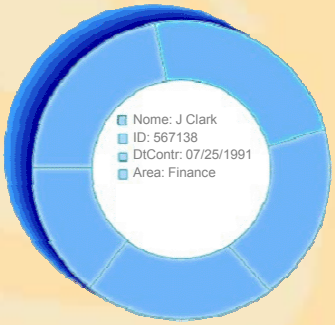
- Estado é representado por atributos e relacionamentos.
- Comportamento é representado por operações, métodos e máquinas de estados.



Objetos possuem estados

O estado de um objeto é a condição ou situação durante o ciclo de vida de um objeto o qual satisfaz algumas condições, executa alguma atividade ou aguarda algum evento. O estado de um objeto normalmente mudo ao longo do tempo.

Nome: J Clark
ID Funcionário: 567138
Contratação: 25 de Julho de 1991
Área: Finanças

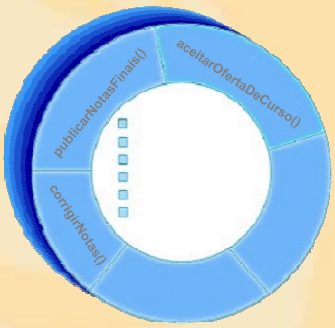


Professor Clark

Objetos possuem comportamento

O comportamento determina como um objeto age e reage. O comportamento observável de um objeto é modelado por um conjunto de mensagens que ele pode responder. (operações que o objeto executa).

Comportamento do Professor Clark
- Publicar notas finais
- Aceitar oferta de curso
- Corrigir notas



Professor Clark

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

PUC • SP COGEAE

Um objeto possui identidade

Cada objeto possui uma identidade única, mesmo que o estado do objeto seja idêncico ao de outro objeto.

Nome: "J Clark"
Ensina: Biologia

Nome: "J Clark"
Ensina: Biologia

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

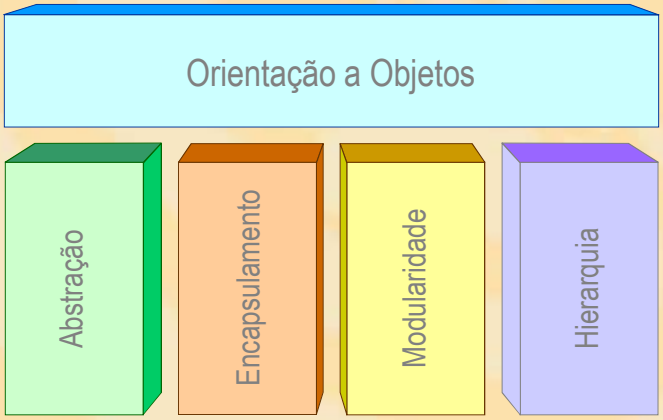
PUC • SP COGEAE

- O que é um objeto?
- Quatro princípios da OO
- O que é uma classe?
- Polimorfismo e Generalização
- Organizando elementos de modelo

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

PUC • SP COGEAE

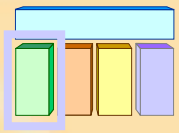
Princípios Básicos de Orientação a Objetos



Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

PUC • SP COGEAE

O que é abstração?



- São características essenciais de uma entidade que a distingue de todos os outros tipos de entidade.
- Define uma fronteira relativa à perspectiva do observador.
- Não é uma manifestação concreta, denota a essência ideal de alguma coisa.

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

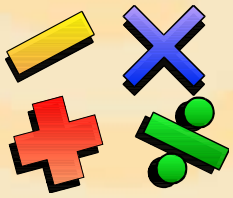
PUC • SP COGEAE

Exemplos de Abstação

Estudante

Professor

Oferta de curso (9:00h, segunda-quarta-sexta)



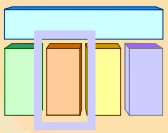
Curso (ex: Álgebra)

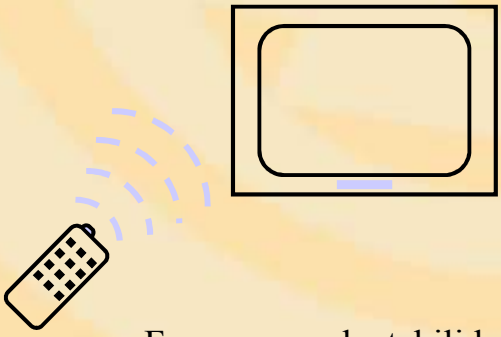
Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

PUC • SP COGEAE

O que é encapsulamento?

- Esconde a implementação dos clientes
- Clientes dependem da interface.





Favorece a adaptabilidade

PUC • SP COGEAE

Exemplo de Encapsulamento

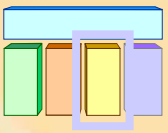
Professor Clark precisa ser capaz de ministrar quatro turmas no próximo semestre.

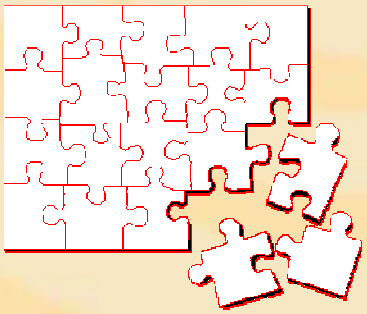
setCargaMáxima(4)

PUC • SP COGEAE

O que é Modularidade?

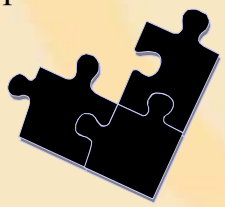
- Quebra algo complexo em partes gerenciáveis.
- Auxilia as pessoas a entender sistemas complexos.





Exemplo de Modularidade

Por exemplo, quebrar um sistema complexo em módulos menores.



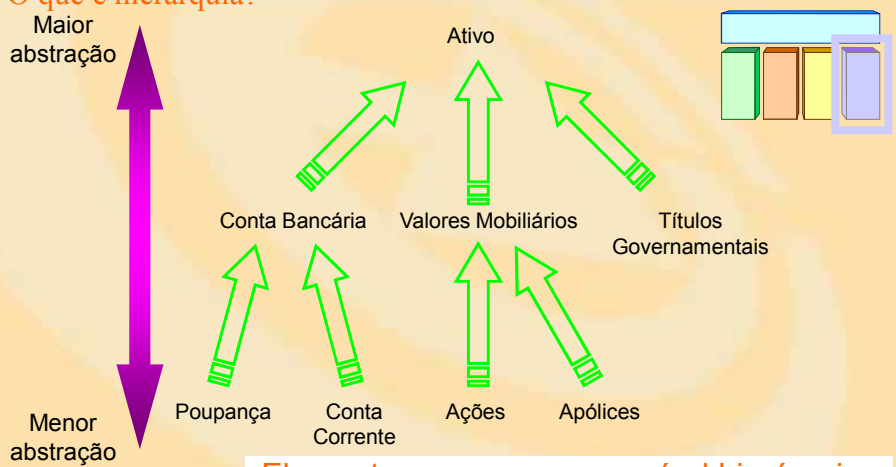
Sistema de registro de cursos

Sistema de cobrança

Sistema de catálogo de cursos

Sistema de gerenciamento de estudantes

O que é hierarquia?



Elementos em um mesmo nível hierárquico devem estar no mesmo nível de abstração.

Representação de Objetos em UML

Um objeto é representado por um retângulo com um nome sublinhado.

Professor J Clark

J Clark : Professor

Objeto "nomeado"

: Professor

Objeto "anônimo"

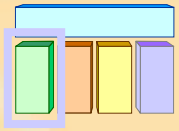
Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

- O que é um objeto?
- Quatro princípios da OO
- O que é uma classe?
- Polimorfismo e Generalização
- Organizando elementos de modelo

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

PUC • SP COGEAE

O que é uma classe?



Uma classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica.

- Um objeto é uma instância de uma classe.
- Uma classe é uma abstração que:
  - *evidencia características relevantes.*
  - *suprime outras características.*

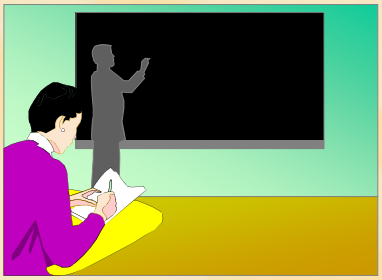
Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

PUC • SP COGEAE

Exemplo de Classes

Classe
Curso

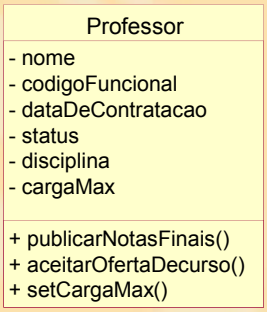
Propriedades
Nome
Localização
Carga horária
créditos
Hora de início
Hora de término



Comportamento
Adicionar estudante
Remover estudante
Obter lista de nomes
Indicar turma lotada

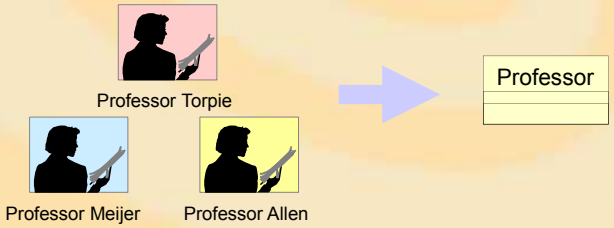
Representando classes em UML

- Uma classe é representada utilizando um retângulo com três compartimentos:
  - O nome da classe
  - A estrutura (atributos)
  - O comportamento (operações)



Relacionamento entre classes e objetos

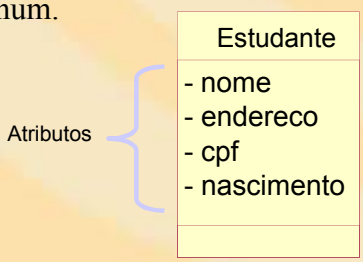
- Uma classe é uma definição abstrata de um objeto.
  - Ela define a estrutura e o comportamento de cada objeto da classe.
  - Se comporta como um padrão (template) para criação de objetos.
- Classes não são coleções de objetos.



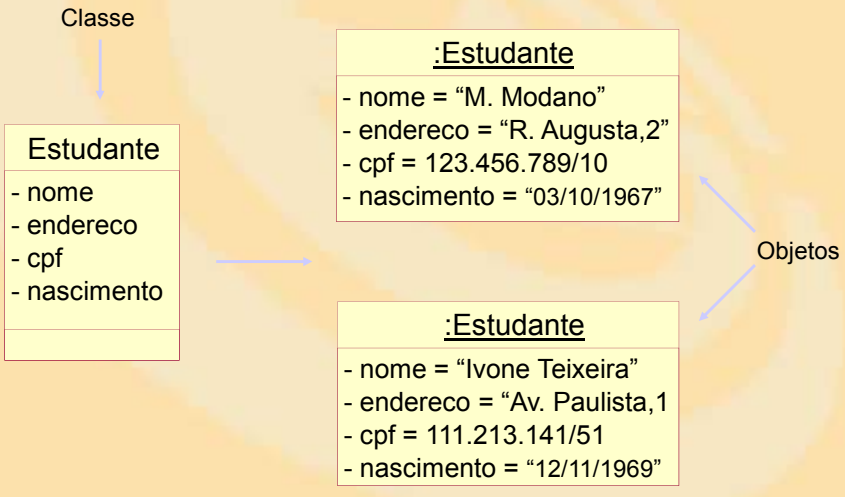
O que é um atributo?

- Um atributo é uma propriedade nomeada de uma classe e descreve um conjunto de valores que as instâncias desta propriedade podem assumir.

  Uma classe pode possuir um número qualquer de atributos inclusive nenhum.



Atributos em classes e objetos

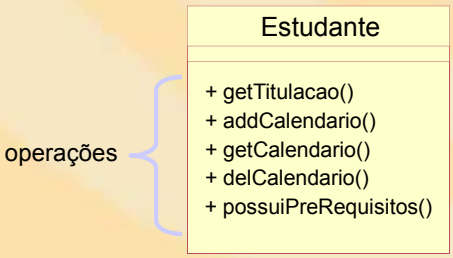


Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

PUC • SP COGEAE

O que é uma operação?

- Um serviço que pode ser requisitado de um objeto para afetar seu comportamento. Uma operação possui uma assinatura a qual define a forma e os parâmetros da operação.
- A classe pode ter nenhuma ou muitas operações.



Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

PUC • SP COGEAE

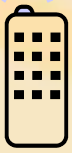
- O que é um objeto?
- Quatro princípios da OO
- O que é uma classe?
- Polimorfismo e Generalização
- Organizando elementos de modelo

PUC • SP COGEAE

O que é polimorfismo?

- Habilidade de esconder muitas implementações por trás de uma única interface.

Princípio OO:
Encapsulamento

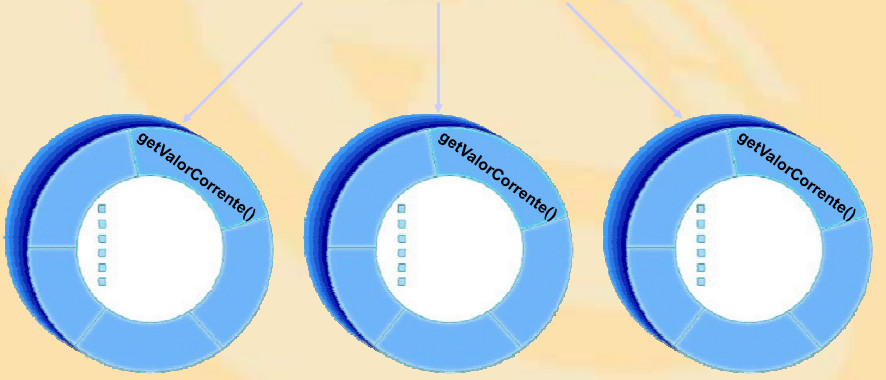


Controle Remoto

PUC • SP COGEAE

Exemplo: Polimorfismo

instrumentoFinanceiro.getValorCorrente()



Ação Título Fundo de Investimento

O que é sobrecarga?

Em uma mesma classe pode possuir operações com o mesmo nome e parâmetros diferentes.

Quando isso ocorre dizemos que este método foi sobrecarregado.

O que é sobreposição?

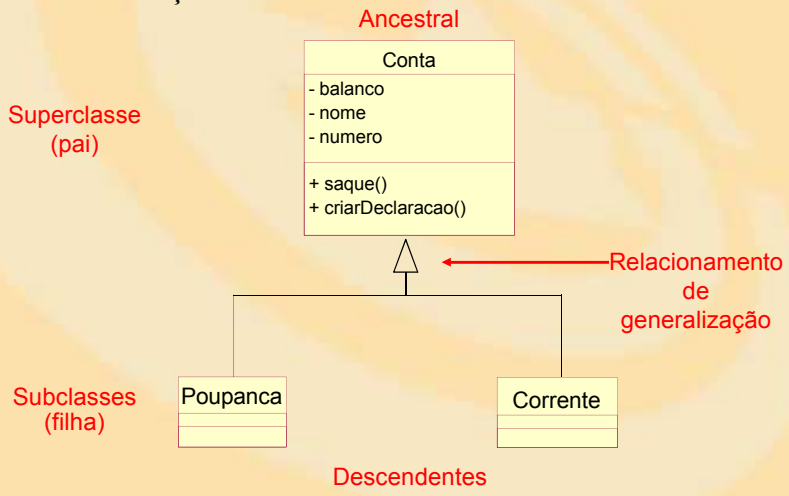
- Em uma hierarquia de classes caso exista um método na sub-classe com a mesma assinatura de um método da classe pai, este se sobreporá o método da super-classe.

O que é Generalização?

- Um relacionamento entre classes onde uma classe compartilha sua estrutura e/ou seu comportamento de uma ou mais classes.
- Define uma hierarquia de abstrações na qual uma subclasse herda elementos de uma hierarquia de superclasses, recebendo como herança suas características e comportamento.
  - Herança simples.
  - Herança múltipla.
- É um relacionamento "é um".

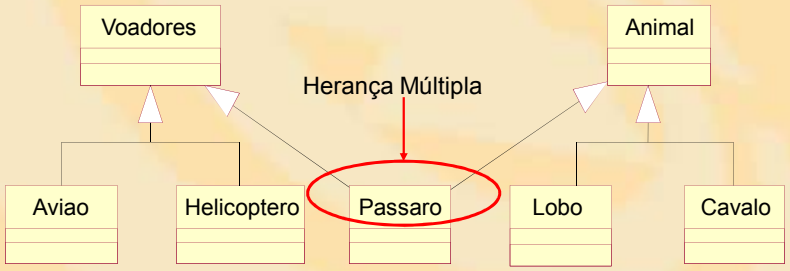
Exemplo: Herança Simples

Uma classe é herança de outra.



Exemplo: Herança Múltipla

Uma classe pode ser herdeira de várias classes.



Use a herança múltipla somente quando realmente necessário e mesmo assim com muita atenção!

O que é Herança?

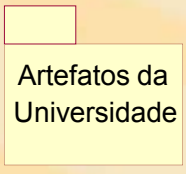
- Um subclasse herda os atributos, operações e relacionamentos da superclasse.
- Uma subclasse pode:
  - Adicionar atributos, operações e relacionamentos novos.
  - Redefinir operações herdadas. (Use com cuidado!)
- Atributos, operações e relacionamentos, são mostrados no nível mais alto aplicável da hierarquia.

A herança alavanca as similaridades através das classes.

- O que é um objeto?
- Quatro princípios da OO
- O que é uma classe?
- Polimorfismo e Generalização
- Organizando elementos de modelo

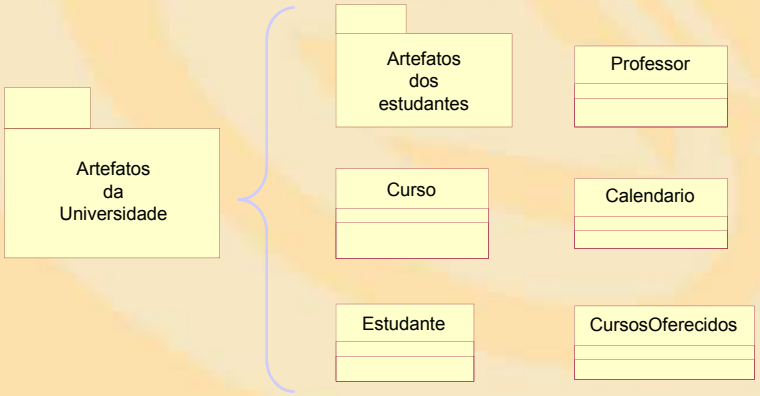
O que é Pacote?

- Um mecanismo de proposta geral para organizar elementos em grupos.
- Um elemento de modelagem que pode conter outros elementos de modelagem.
- Um pacote pode ser usado:
  - Para organizar o modelo em desenvolvimento.
  - Como uma unidade de gestão de configuração.



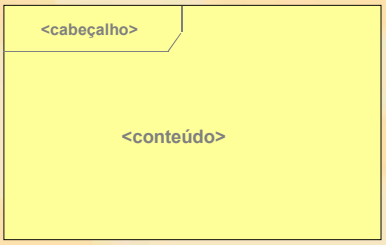
Um pacote pode conter classes

O pacote Artefatos da Universidade contém um pacote e cinco classes.



Representação gráfica de um diagrama

- Cada diagrama possui um quadro, um compartimento de cabeçalho no canto superior esquerdo e uma área de conteúdo.
  - Se o quadro não fornecer nenhuma informação adicional, este pode ser omitido e a borda da área do diagrama fornecida pela ferramenta será o quadro indicado se necessário.



Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

PUC • SP

Revisão

- O que é um objeto?
- Quais são os quatro princípios de orientação a objetos? Descrever cada um.
- O que é uma classe? Como relacionar classes e objetos?
- O que é um atributo? Uma operação?
- Defina polimorfismo. Forneça um exemplo de polimorfismo.
- O que é generalização?
- Por que usar pacotes?

Modelagem de Software Orientado a Objetos
Princípios de Orientação a Objetos

PUC • SP

Dúvidas?

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Modelagem de Software Orientado a Objetos

Parte 2- Princípios de Modelagem UML

Prof. Dr. Maurício Nacib Pontuschka
tuska@pucsp.br

O que é a UML?

Uma linguagem para:

- visualização,
- especificação,
- construção e
- documentação

de artefatos de um sistema de software

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

A UML é uma linguagem de visualização

- Comunicar modelos conceituais para outras pessoas tende a erros a não ser que todos falem a mesma língua.
- Existem coisas a respeito de sistemas de software que não podem ser entendidas sem a construção de modelos.
- Um modelo explícito facilita a comunicação.

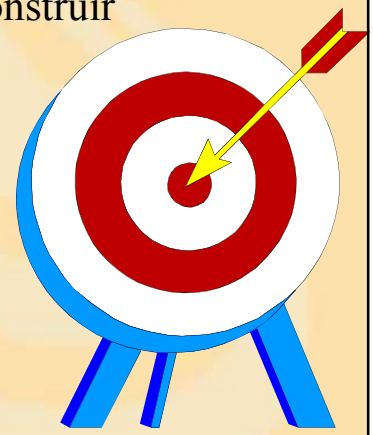
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

A UML é uma linguagem de especificação

É possível, por meio da UML construir modelos de forma

precisa, com um mínimo de

ambigüidades.



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

A UML é uma linguagem para construção

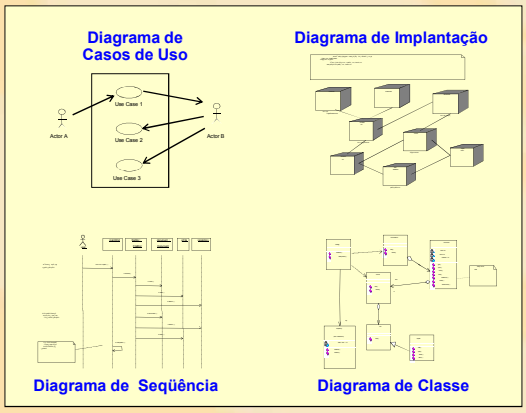
Modelos UML podem ser conectados diretamente a uma ampla variedade de linguagens de programação (Java, C++, C#, Ruby entre outras).

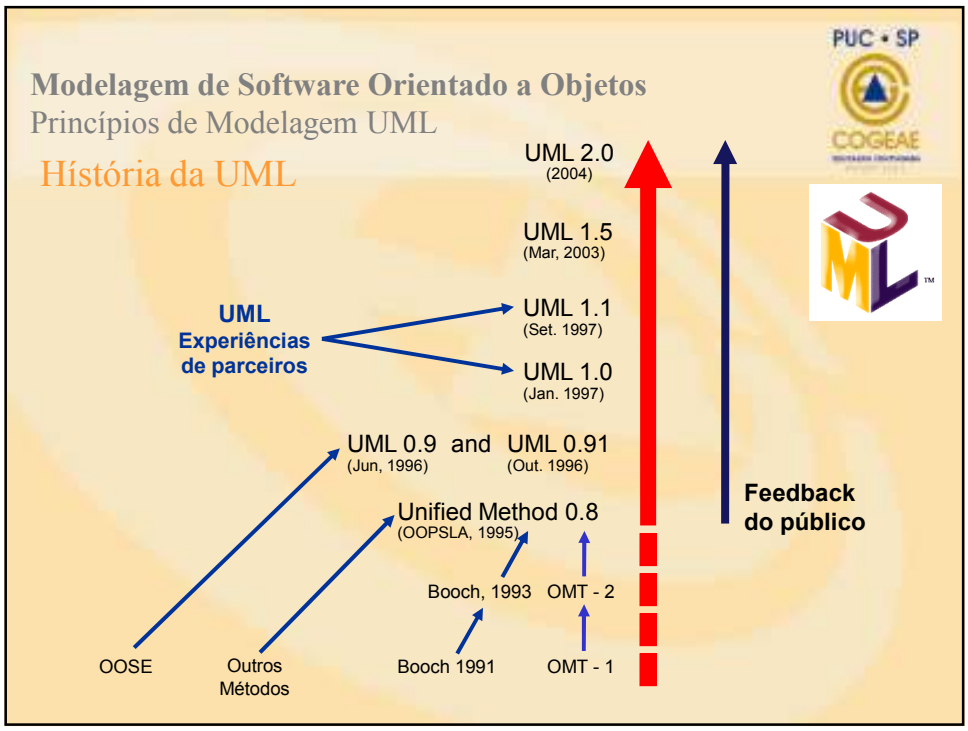
- Mapeamento para tabelas em um SGBDR ou armazenamento persistente em um SGBDOO.
- Permite a engenharia e engenharia reversa.

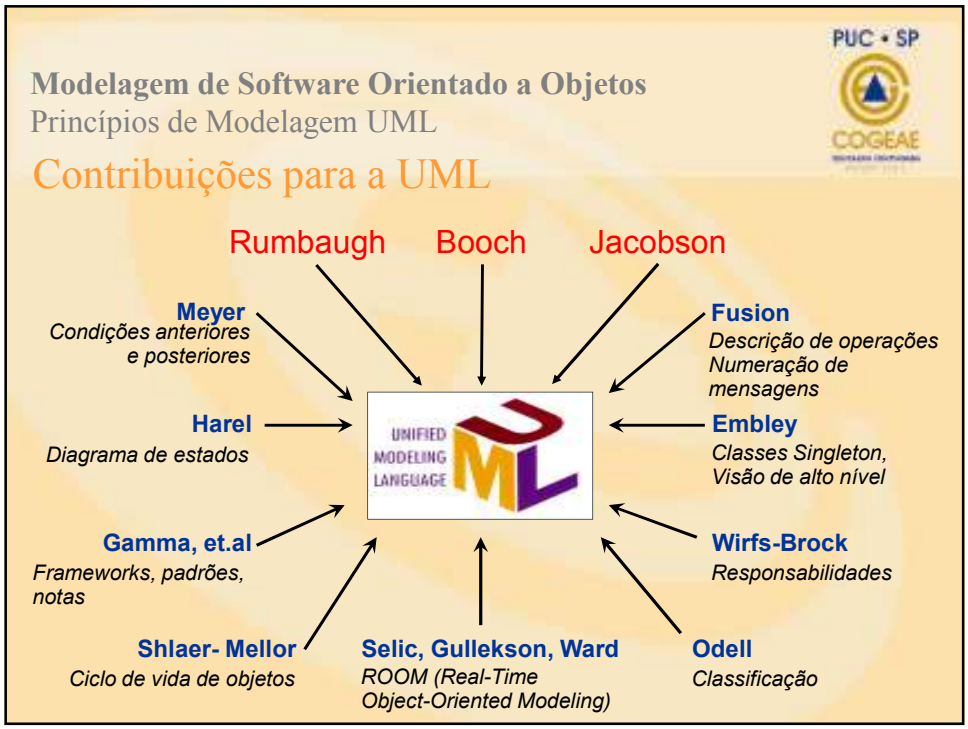
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

A UML é uma linguagem para Documentação

Documenta a arquitetura do sistema, requisitos, testes, planejamento do projeto e gestão de versionamento.

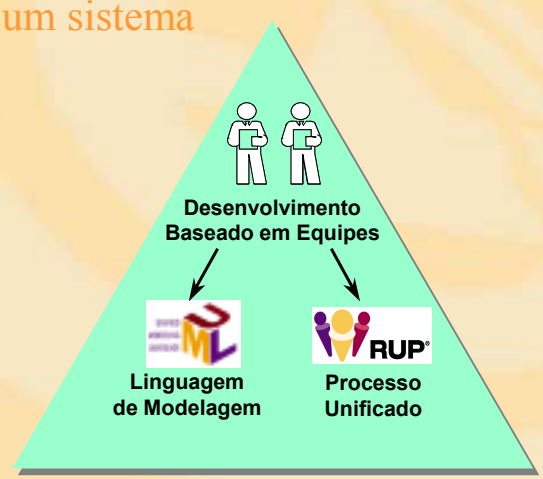



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML
História da UML
PUC • SP
COGEAE
UML 2.0
(2004)
UML 1.5
(Mar, 2003)
UML
Experiências
de parceiros
UML 1.1
(Set. 1997)
UML 1.0
(Jan. 1997)
UML 0.9 and UML 0.91
(Jun, 1996) (Out. 1996)
Feedback
do público
Unified Method 0.8
(OOPSLA, 1995)
Booch, 1993
OMT - 2
OOSE
Outros
Métodos
Booch 1991
OMT - 1


Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML
Contribuições para a UML
PUC • SP
COGEAE
Rumbaugh
Booch
Jacobson
Meyer
Condições anteriores
e posteriores
Fusion
Descrição de operações
Numeração de
mensagens
Harel
Diagrama de estados
UNIFIED
MODELING
LANGUAGE
Embley
Classes Singleton,
Visão de alto nível
Gamma, et.al
Frameworks, padrões,
notas
Wirfs-Brock
Responsabilidades
Shlaer- Mellor
Ciclo de vida de objetos
Selic, Gullekson, Ward
ROOM (Real-Time
Object-Oriented Modeling)
Odell
Classificação

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

A linguagem não é suficiente para construir um sistema



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

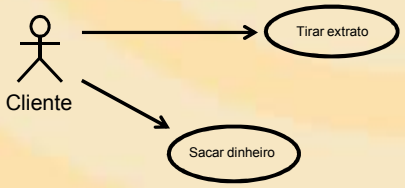
Que tipo de processo benificia mais a UML?

A UML é independente de processo. Um processo beneficia completamente a UML quando ele é:

- guiado por casos de uso;
- centrado em arquitetura;
- iterativo e incremental.

Processo guiado por casos de uso

- Casos de uso definidos para um sistema são base para o processo de desenvolvimento inteiro.
- Benefícios dos casos de uso:
  - Concisos, simples e de fácil entendimento por um grande número de stakeholders.
  - Ajudam a sincronizar o conteúdo de diferentes modelos.



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Um processo centrado na arquitetura

- A arquitetura do sistema é usada como artefato primário na concepção, construção, gerenciamento e para envolver o sistema sob desenvolvimento..
- Benefícios:
  - Controle intelectual sobre o projeto para gerenciar a complexidade do projeto a fim de manter a integridade co sistema.
  - Base efetiva para reúso em larga escala.
  - Uma base para o gerenciamento do projeto.
  - Assistência no desenvolvimento baseado em componentes.

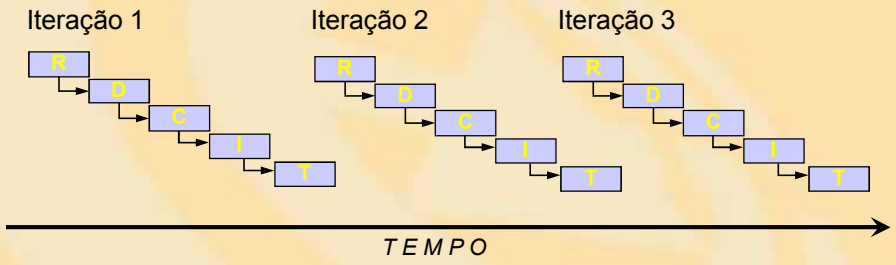
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Um processo iterativo e incremental

- Riscos críticos são resolvidos antes de realizar grandes investimentos.
- Iterações iniciais habilitam um feedback de usuário antecipado.
- Testes e integração são contínuos.
- Foco em etapas objetivas a curto prazo.
- O progresso é mensurado através do fornecimento de implementações.
- Implementações parciais podem ser disponibilizadas.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Desenvolvimento Iterativo



- Iterações mais recentes endereçam os maiores riscos.
- Cada iteração produz uma versão executável, um incremento adicional do sistema.
- Cada iteração inclui integração e teste.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Revisão

- O que é a UML? Descreva cada um de seus benefícios.
- Quais características de processo melhor se adapta à UML? Descreva cada característica.
- O que é uma iteração?

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Dúvidas?

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Modelagem de Software Orientado a Objetos

Parte 2- Princípios de Modelagem UML
Modelagem por Casos de Uso

Prof. Dr. Maurício Nacib Pontuschka
tuska@pucsp.br

Objetivos

- Descrever o comportamento de um sistema e mostrar como capturá-lo em um modelo.
- Demostrar como ler e interpretar:
  - um diagrama de casos de uso;
  - um diagrama de atividades.

PUC • SP COGEAE

Onde estamos?

- **Conceitos na modelagem de casos de uso**
- Diagramas de caso de uso
- Diagramas de atividade

PUC • SP COGEAE

O que comportamento do sistema?

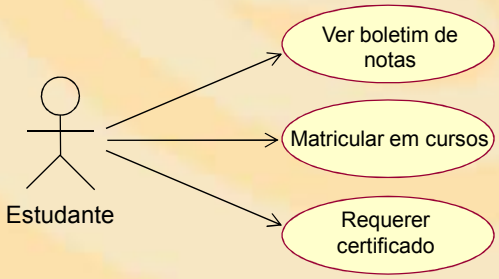
- Comportamento do sistema é como o sistema age e reage.
  - É representado pelas ações e atividades de um sistema.
- O comportamento do sistema é capturado em casos de uso.
  - Casos de uso descrevem as interações entre sistema e (partes do) ambiente.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

O que é um modelo de casos de uso?

- Um modelo descreve um requisito funcional do sistema em termos de casos de uso.
- Um modelo das necessidades pretendidas do sistema (casos de uso) e seu ambiente (atores).

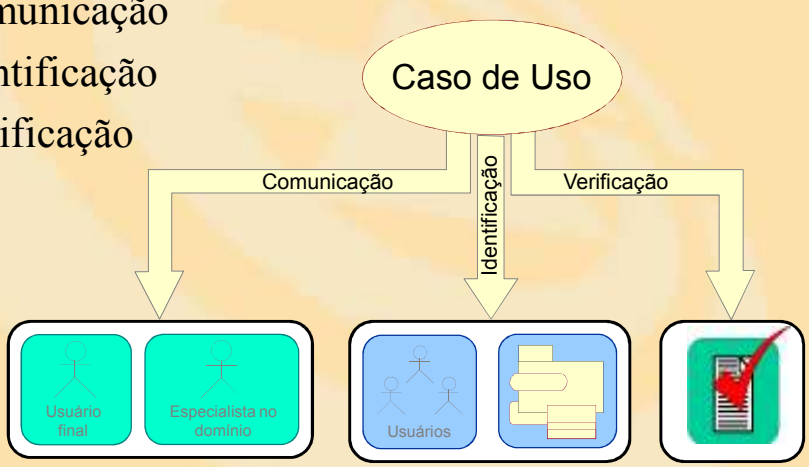


Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Quais são os benefícios de um modelo de casos de uso?

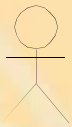
- Comunicação
- Identificação
- Verificação



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

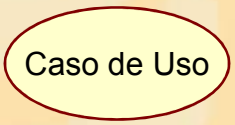
Conceitos principais na modelagem de casos de uso

- Um ator representa qualquer coisa que interage com o sistema.



Ator

- Um caso de uso descreve uma seqüência de eventos, realizados pelo sistema, que produzam resultados de valor observáveis a um ator em particular.



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

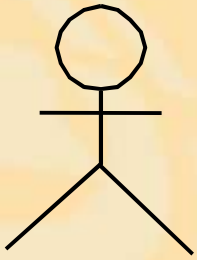
Onde estamos?

- Conceitos na modelagem de casos de uso
- **Diagramas de caso de uso**
- Diagramas de atividade

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

O que é um ator?

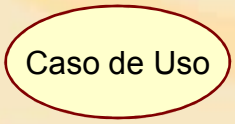
- Atores representam papéis que os usuários do sistema podem assumir.
- Podem representar uma pessoa, uma máquina ou outros sistemas.
- Podem trocar informações com o sistema ativamente.
- Podem ser fornecedores de informação.
- Podem ser receptores passivos de informação.
- Atores não fazem parte do sistema
  - **Atores são EXTERNOS**.



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

O que é um caso de uso?

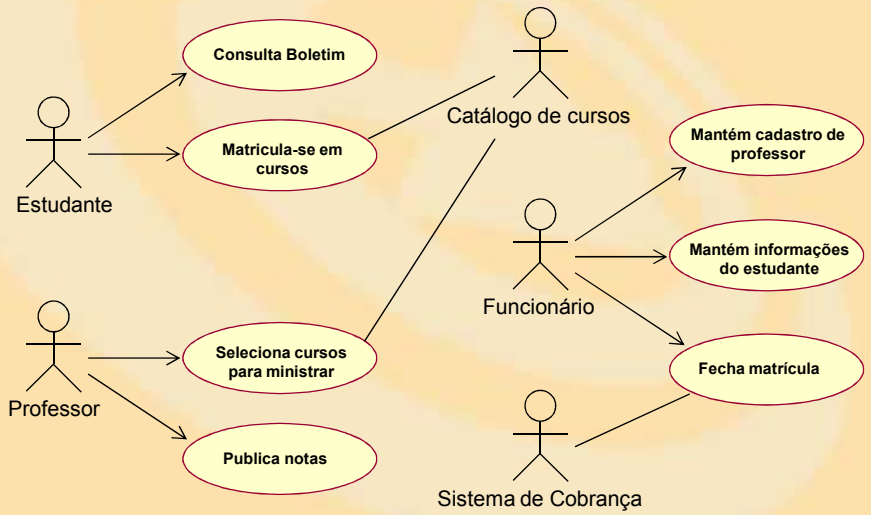
- Define um conjunto de instâncias de casos de uso onde cada instância é uma seqüência de ações que o sistema executa e que produz resultados de valor a um ator em particular.
  - Um caso de uso modela um dialogo entre um ou mais atores com o sistema.
  - Um caso de uso descreve as ações que o sistema realiza para produzir algo de valor ao ator.



Casos de Uso e Atores

- Um caso de uso modela um diálogo entre atores e o sistema.
- Um caso de uso é iniciado por um ator para invocar uma determinada funcionalidade no sistema.

Como você lê este diagrama?



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Onde estamos?

- Conceitos na modelagem de casos de uso
- Diagramas de caso de uso
- **Diagramas de atividade**

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

O que é um diagrama de atividades?

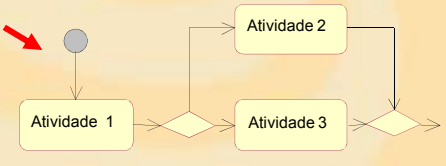
- Um diagrama de atividades em um modelo de casos de uso podem ser usados para capturar as atividades e ações executadas em um caso de uso.
- É em sua essência um diagrama de fluxo evidenciando o controle do fluxo entre as atividades.

Fluxo de Eventos

Este caso de uso inicia quando o Funcionário requisitar que o sistema feche a matrícula.

1. O sistema verifica se existe uma matrícula em andamento. Se estiver, uma mensagem é exibida ao Funcionário e o caso de uso termina. O fechamento do processo de matrícula não pode ser fechado se houve matrícula em andamento.

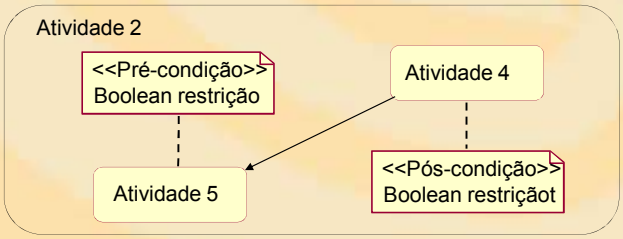
2. Para cada oferta de curso o sistema verifica se um professor está designado a ministrar o curso oferecido e pelo menos três estudantes tenham sido matriculados. No caso positivo, o sistema realiza a oferta de curso para cada horário dos cursos.



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

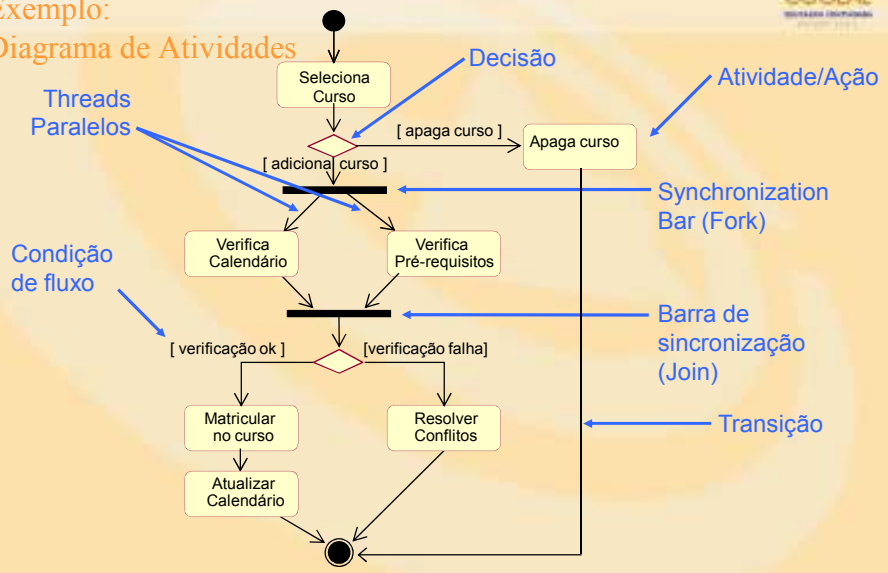
O que é uma atividade?

- Uma especificação do comportamento expresso como um fluxo de execução através de uma seqüência de unidades subordinadas.
  - Unidades subordinadas incluem atividades internas e resultantes de ações individuais.
- Pode conter expressões booleanas quando a atividade é iniciada ou finalizada.



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Exemplo: Diagrama de Atividades



Descrição textual de casos de uso

É possível também descrever um caso de uso utilizando outros recursos que não diagramas de atividades como por exemplo, por meio de textos ao invés de diagramas.

O texto a seguir descreve um caso de uso sem a utilização de diagramas.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

MN# 01 – *Publicar Notas*	
Ator Principal: ***Professor***	
Pré-Condições: ***O Professor deve estar alocado em algum curso em andamento.***	
Pós-Condições: ***As notas de uma determinada turma do professor estará com as notas de seus alunos lançadas.***	
FLUXO PRINCIPAL	
ATOR	**SISTEMA**
1. Acessa a opção de publicar notas.	
	2. Solicita a turma.
3. Informa a turma.	
	4. Mostra tela com os nomes dos alunos com os respectivos campos para digitação das notas.
5. Digita as notas dos alunos.	
	6. Registra e disponibiliza as notas dos alunos.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

FA1	FLUXO ALTERNATIVO
FLUXO DE ORIGEM: FP	**FLUXO DE RETORNO: FP**
ATOR	**SISTEMA**
	4. Informa que a turma não existe
5. Solicita abertura de nova turma.	
	6. Solicita o nome da nova turma
7. Informa o nome.	
	8. Registra o nome da turma e continua no passo 4 do Fluxo Principal.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

FE1	FLUXO EXCEPCIONAL
FLUXO DE ORIGEM: FP	**FLUXO DE RETORNO: -**
ATOR	**SISTEMA**
	4. Informa que a turma não existe
5. Cancela a operação.	

OBSERVAÇÃO
Embora nestes slides os fluxos tenham sido colocados de forma separada para uma melhor visualização no formato de apresentação, estes podem ser contínuos em uma mesma tabela.

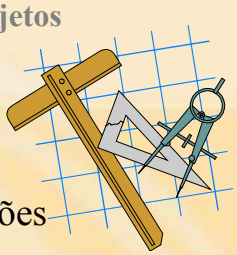
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Revisão

- O que é comportamento do sistema?
- O que é modelo de casos de uso? Quais são os seus benefícios?
- O que é um ator? E um caso de uso?
- O que é um diagrama de atividades?

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Exercício



- Dados:
  - Casos de uso, atores e associações
- Desenhe:
  - Um diagrama de casos de uso
- Dados:
  - Ações, estados e atividades
- Desenhe:
  - Um diagrama de atividades

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Fábio é um construtor de casas para condomínios de alto padrão em uma cidade do interior de São Paulo. Em suas atividades profissionais ele necessita realizar alguns controles de forma a garantir que seus projetos sejam executados com qualidade.

Uma das principais necessidades de Fábio é a de controlar a compra, utilização e estoque de materiais de construção. Muitas vezes materiais acabam sendo comprados em duplicidade, uma vez que o chefe de obras possui uma pequena verba para compras de materiais em situações de emergência.

Outro controle que representa um fator crítico em suas atividades é controle de mão de obra. Os operários devem ser cadastrados e acompanhados em termos de horas trabalhadas a fim de realizar os pagamentos no final da semana.

Fábio gostaria de criar um mapa de atividades das obras e de vincular os operários a cada uma das atividades a fim de permitir otimizações no processo.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Dúvidas?

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Modelagem de Software Orientado a Objetos

Parte 2- Princípios de Modelagem UML
Diagramas de Interação

Prof. Dr. Maurício Nacib Pontuschka
tuska@pucsp.br

Objetivos

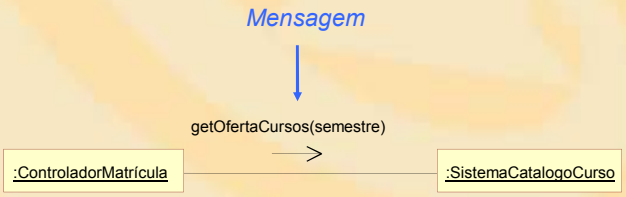
- Descrever a dinâmica comportamental e mostar como capturá-la em um modelo.
- Demonstrar como ler e interpretar:
  - Um diagrama de seqüência
  - Um diagrama de comunicação
- Explicar as similaridades e diferenças entre os diagramas de comunicação e de seqüência.

Objetos precisam colaborar

- Objetos são inúteis a não ser que possam colaborar na solução de problemas.
  - Cada objeto é responsável por seu comportamentoe estado.
  - Nenhum objeto pode assumir toda a responsabilidade sozinho.
- Como os objetos interagem entre si?
  - Eles interagem por meio de mensagens.

Objetos interagem por meio de mensagens

- Uma mensagem mostra como um objeto pede a outro para que alguma atividade seja executada.

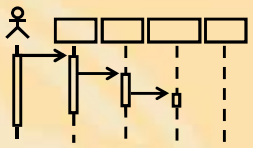


O que é um diagrama de interações?

- É um termo genárico que se aplica a vários diagramas que enfatizam as interações entre objetos
  - Diagrama de Seqüência
  - Diagrama de Comunicação
- Variantes Especializadas
  - Diagrama de Tempo
  - Diagrama Geral de Interação

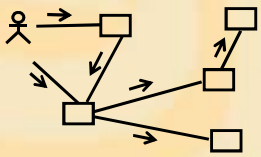
Diagramas de Interação

- Diagrama de Seqüência
  - Visão cronológica das interações dos objetos.



Diagramas de Seqüência

- Diagrama de Comunicação
  - Visão estrutural das mensagens dos objetos.

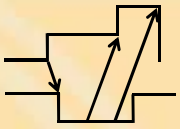


Diagramas de Comunicação

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

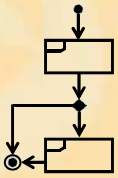
Diagramas de Interação

- Diagrama de Tempo
  - Visão de restrições de tempo das mensagens envolvidas em uma interação.



Diagramas de Tempo

- Diagrama Geral de Interação
  - Visão de alto nível dos conjuntos de interação combinados em seqüências lógicas.



Diagramas Gerais de Interações

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Onde estamos?

- **Diagramas de Seqüência**
- Diagramas de Comunicação
- Comparação dos diagramas de interação

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

O que é um diagrama de seqüência?

- Um diagrama de seqüência é um diagrama de interação que enfatiza a ordem cronológica de mensagens.
- O diagrama exibe:
  - Os objetos que participam da interação.
  - A seqüência de mensagens trocadas.

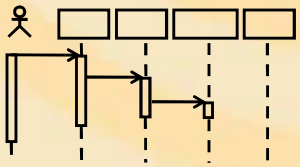
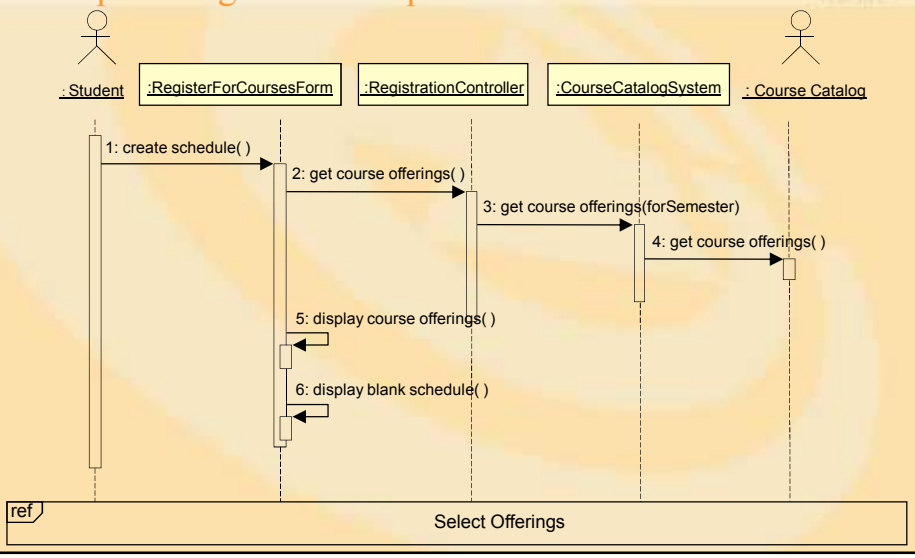


Diagrama de seqüência

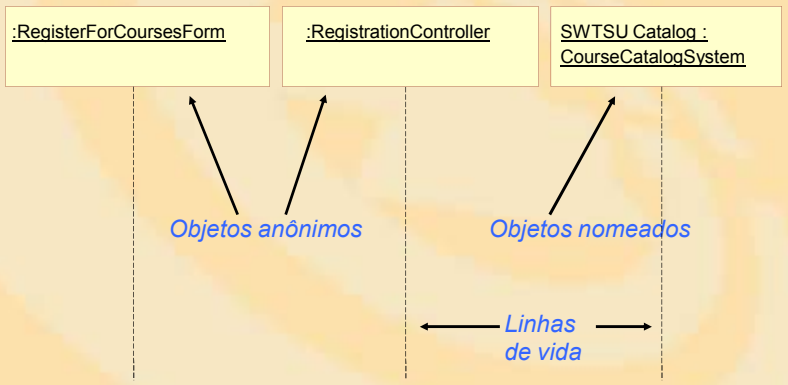
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Exemplo: Diagrama de seqüência



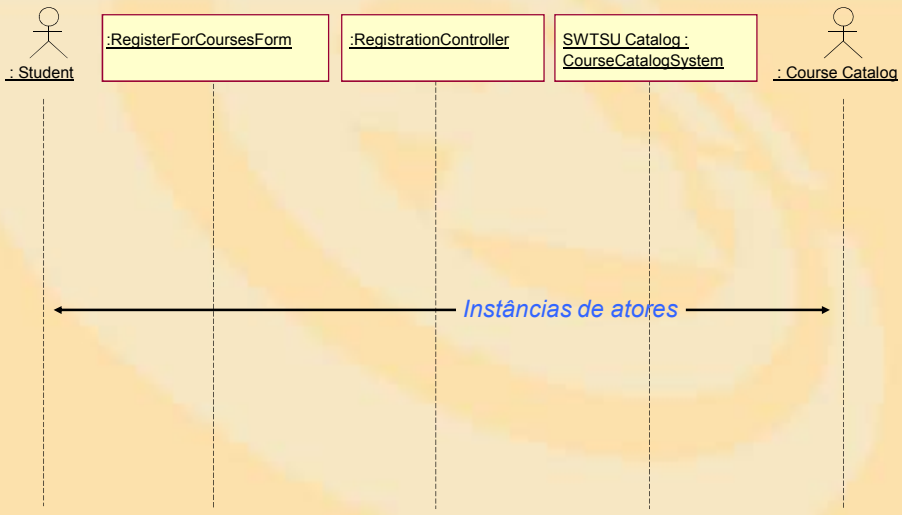
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Elementos do Diagrama de Seqüência: Objetos

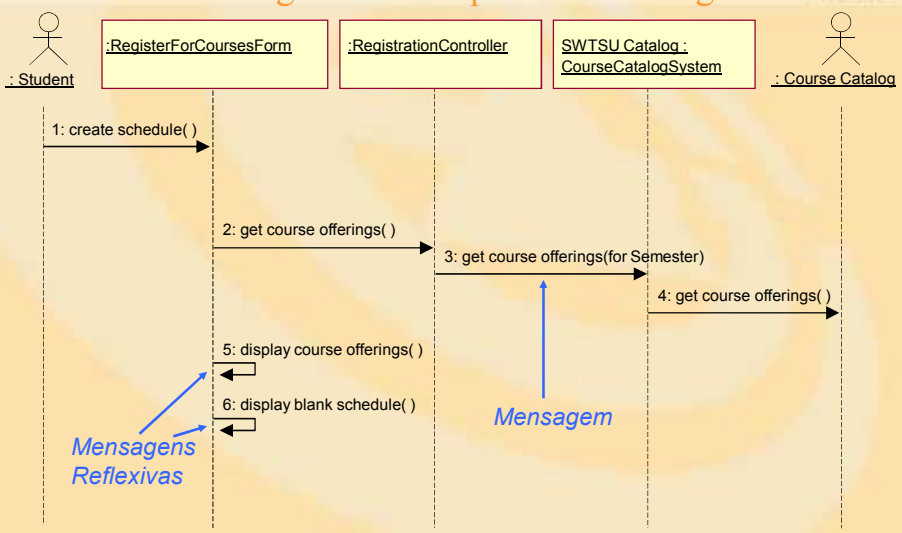


Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

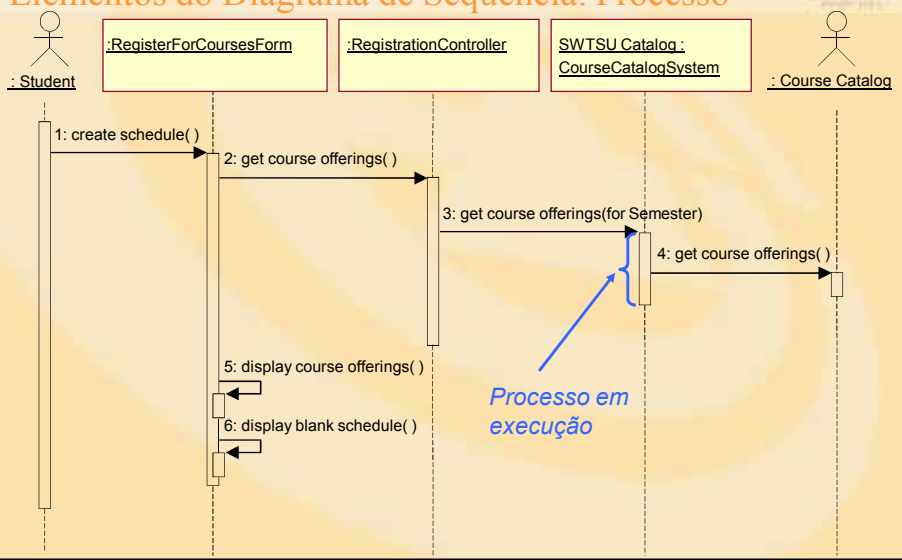
Elementos do Diagrama de Seqüência: Atores

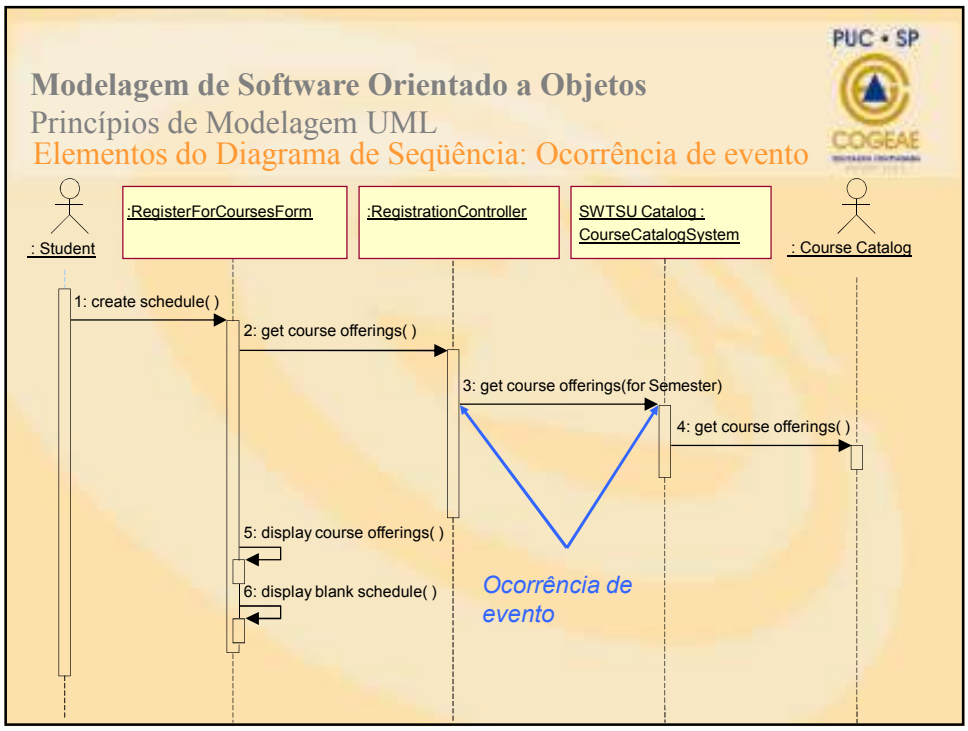


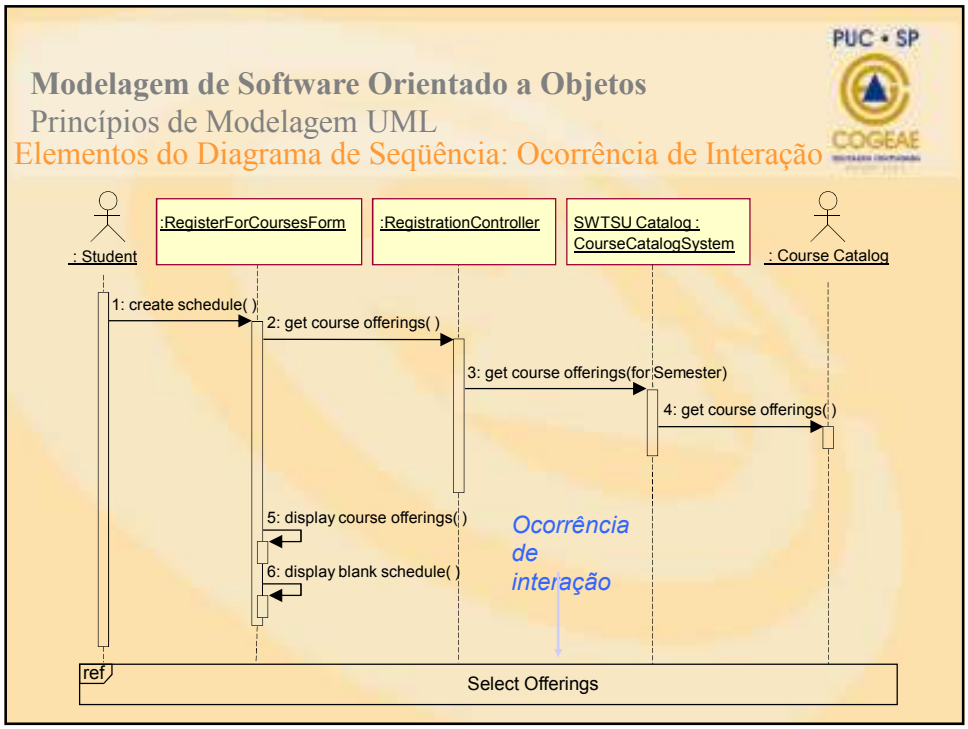
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML
Elementos do Diagrama de Seqüência: Mensagens



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML
Elementos do Diagrama de Seqüência: Processo




PUC • SP
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML
Elementos do Diagrama de Seqüência: Ocorrência de evento
: Student
:RegisterForCoursesForm
:RegistrationController
SWTSU Catalog : CourseCatalogSystem
: Course Catalog
1: create schedule( )
2: get course offerings( )
3: get course offerings(for Semester)
4: get course offerings( )
5: display course offerings( )
6: display blank schedule( )
Ocorrência de evento


PUC • SP
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML
Elementos do Diagrama de Seqüência: Ocorrência de Interação
: Student
:RegisterForCoursesForm
:RegistrationController
SWTSU Catalog : CourseCatalogSystem
: Course Catalog
1: create schedule( )
2: get course offerings( )
3: get course offerings(for Semester)
4: get course offerings( )
5: display course offerings( )
6: display blank schedule( )
Ocorrência de interação
ref
Select Offerings

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

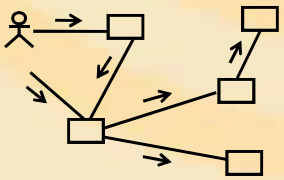
Onde estamos?

- Diagramas de Seqüência
- **Diagramas de Comunicação**
- Comparação dos diagramas de interação

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

O que é um diagrama de comunicação?

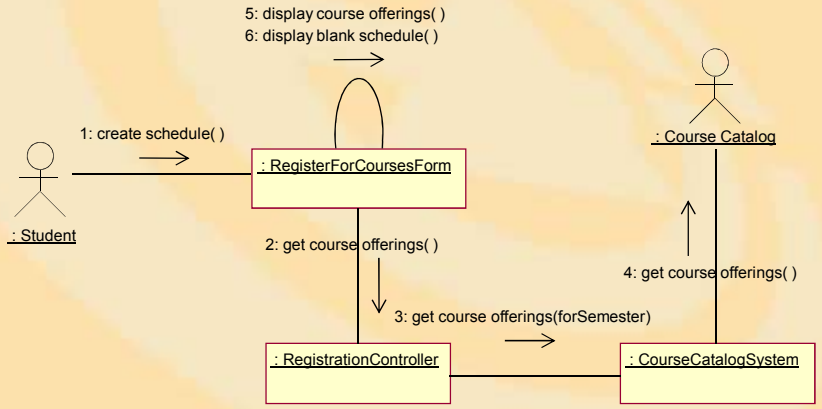
- Um diagrama de comunicação enfatiza a organização de objetos em uma interação.
- Um diagrama de comunicação mostra:
  - Os objetos que participam da interação.
  - Ligações entre os objetos.
  - Mensagens passadas entre os objetos.



Diagramas de Interação

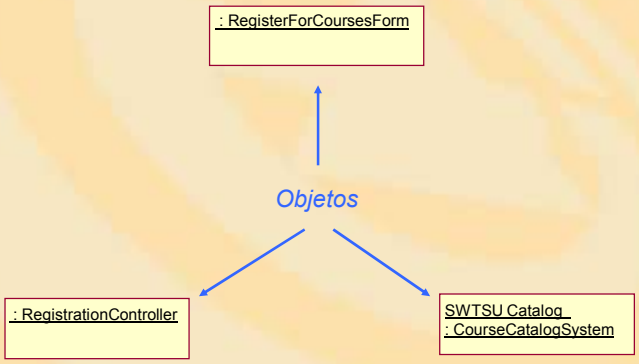
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Exemplo: Diagrama de Comunicação



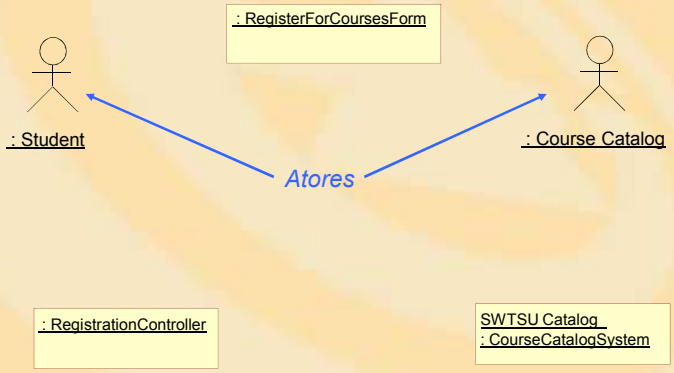
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Elementos do Diagrama de Comunicação: Objetos



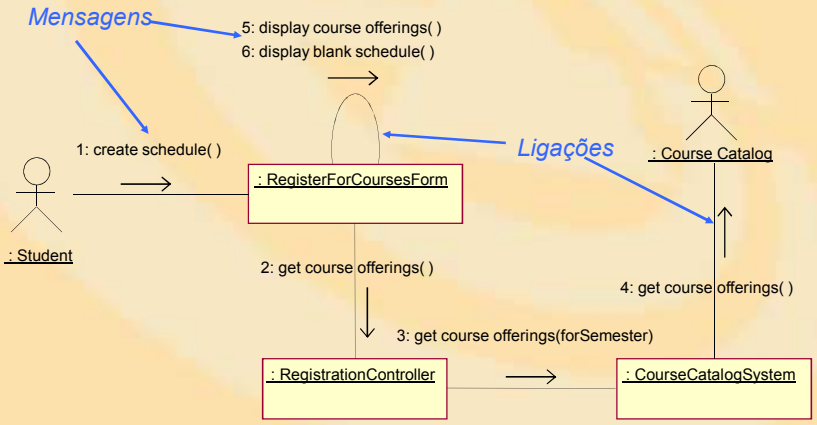
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Elementos do Diagrama de Comunicação: Atores



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Elementos do Diagrama de Comunicação: Ligações e mensagens



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Onde estamos?

- Diagramas de Seqüência
- Diagramas de Comunicação
- **Comparação dos diagramas de interação**

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Similaridades entre os diagramas de Seqüência e de Comunicação

- São semanticamente equivalentes
  - Podem ser convertidos de um para o outro sem perda de informação
- Modelam aspectos dinâmicos de um sistema
- Modelam o cenário de um caso de uso

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Diferenças entre diagramas de Seqüência e de Comunicação

Diagramas de Seqüência	Diagramas de Comunicação
– Exibem explicitamente a seqüência de mensagens. – Exibem as ocorrências de execução. – Melhores para a visualização geral do fluxo. – Melhores para especificações de tempo-real e cenários complexos.	– Exibem relacionamentos, além das interações. – Melhores para visualização de padrões de comunicação. – Melhores para visualização de todos os efeitos de um dado objeto. – Mais fáceis de serem utilizados em sessões de "brainstorming".

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

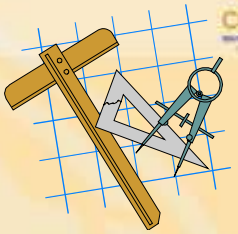
Resumo

- Qual a proposta de um diagrama de interação?
- O que é um diagrama de seqüência? E um diagrama de comunicação?
- O que é um diagrama de tempo? E um diagrama geral de interação?
- Quais são as similaridades entre os diagramas de seqüência e de comunicação?
- Quais são as diferenças entre os diagramas de seqüência e de comunicação?

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Exercício



- Dado:
  - Um conjunto de objetos, suas ligações e mensagens.
- Produzir:
  - Um diagrama de seqüência.
  - Um diagrama de mensagens.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Dúvidas?

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Modelagem de Software Orientado a Objetos

Parte 2- Princípios de Modelagem UML
Diagrama de Classes

Prof. Dr. Maurício Nacib Pontuschka
tuska@pucsp.br

Objetivos

- Descrever uma visão estática do sistema e mostrar como capturá-la em um modelo.
- Demonstrar como ler e interpretar um diagrama de classes.
- Modelar associações e agregações e mostrar como incluir o modelo em um diagrama de classes.
- Modelar generalizações em um diagrama de classes.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Onde Estamos?

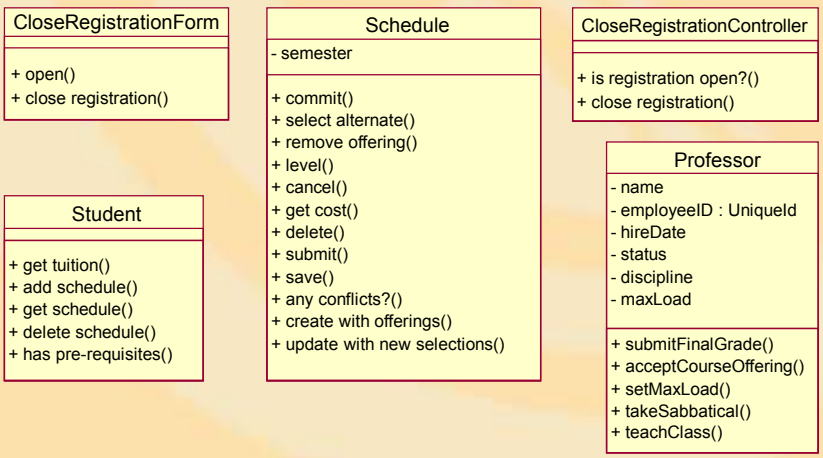
- **Diagrama de Classes**
- Relacionamento de Classes
  - Associação
  - Agregação
  - Generalização

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

O que é um diagrama de classes?

Visão estática de um sistema

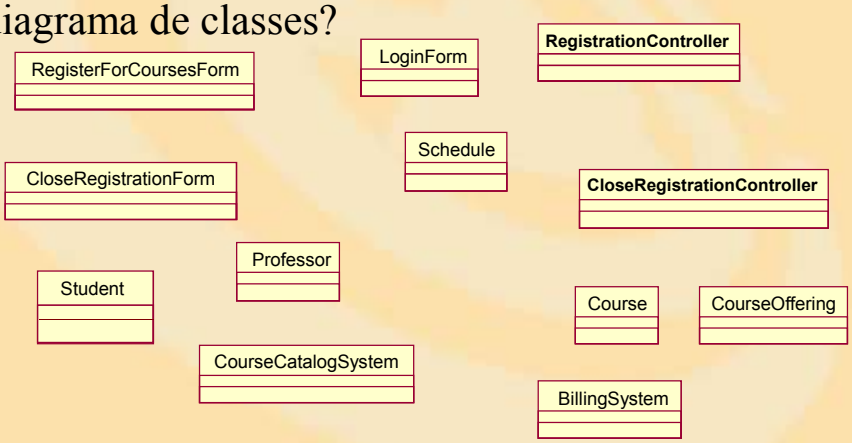


Utilidade de um diagrama de classes

- Ao modelar a visão estática de um sistema, o diagrama de classes é tipicamente utilizado de três formas, para modelar:
  - o vocabulário de um sistema,
  - colaborações,
  - Esquema lógico de um banco de dados.

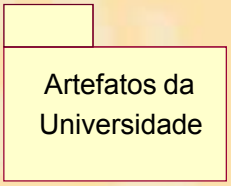
Exempo: Diagrama de Classes

Existe uma melhor forma de se organizar um diagrama de classes?

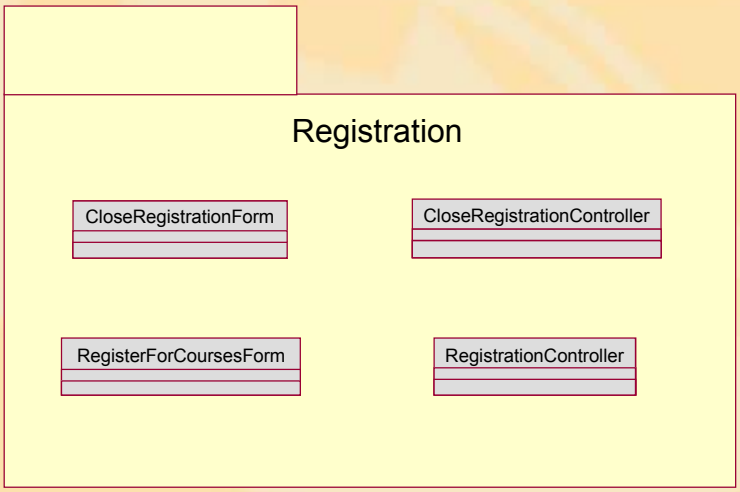


Revisão: O que é um pacote?

- Um mecanismo genárico para organização de elementos em grupos.
- Um elemento de modelo que pode conter outros elementos de modelo..
- Um pacote pode ser utilizado para:
  - Organizar um modelo sob desenvolvimento;
  - Como unidade de gestão de configuração.



Exemplo: Pacote Registration



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Onde Estamos?

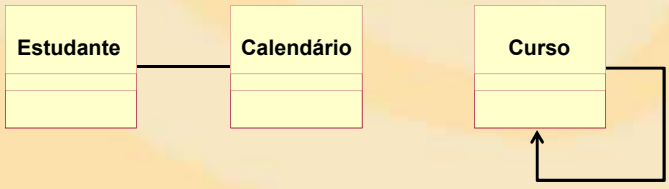
- Diagrama de Classes
- **Relacionamento de Classes**
  - **Associação**
  - Agregação
  - Generalização

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

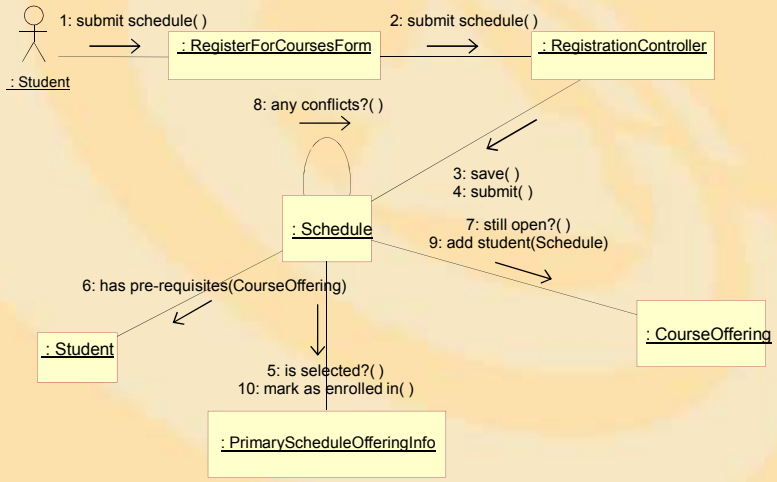
O que é uma associação?

- Relacionamento semântico entre dois ou mais classificadores que especifica conexões entre as suas instâncias.
- Um relacionamento estrutural que especifica que objetos de uma classe estão conectados aos da outra classe.



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Exemplo: Que associações você pode encontrar?



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

O que é multiplicidade?

- Multiplicidade é o número de instâncias de uma classe qual se relaciona com UMA instância da outra classe.
- Para cada associação, existem duas decisões de multiplicidade a serem tomadas, uma para cada extremidade da associação.
  - Para cada instância de Professor, muitas ofertas de cursos podem existir.
  - Para cada instância de oferta de curso, podem haver um ou nenhum professor como instrutor.

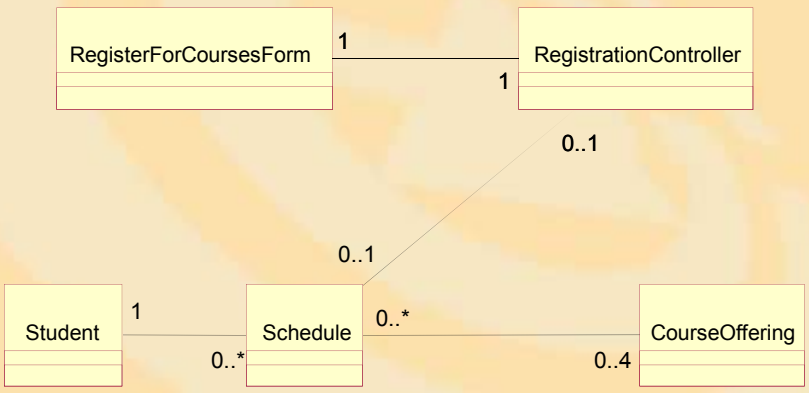
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Indicadores de Multiplicidade

Não especificado	
Exatamente um	1
Zero ou mais	0..*
Zero ou mais	*
Um ou mais	1..*
Zero ou um (valor opcional)	0..1
Faixa específica	2..4
Múltiplas Faixas descontínuas	2, 4..6

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Exemplo: Multiplicidade



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Onde Estamos?

- Diagrama de Classes
- **Relacionamento de Classes**
  - Associação
  - **Agregação**
  - Generalização

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

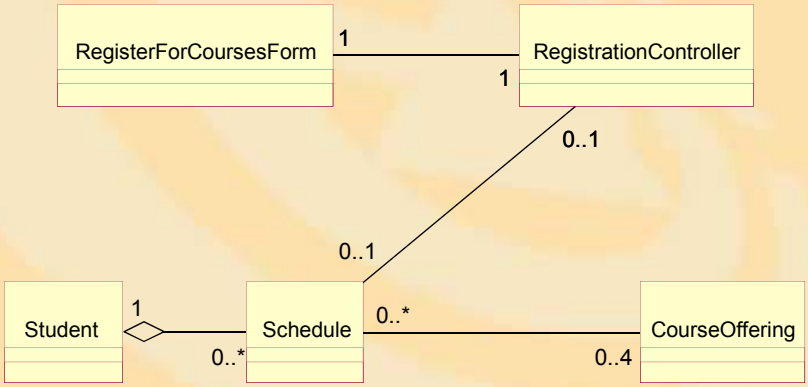
PUC • SP COGEAE

O que é uma agregação?

- Uma forma especial de associação a qual modela um relacionamento parte-de entre a agregação (o todo) e suas partes.
  - Uma agregação é um relacionamento "parte-de". A multiplicidade é representada da mesma forma que em outras associações.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Exemplo: Agregação



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Onde Estamos?

- Diagrama de Classes
- **Relacionamento de Classes**
  - Associação
  - Agregação
  - **Generalização**

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

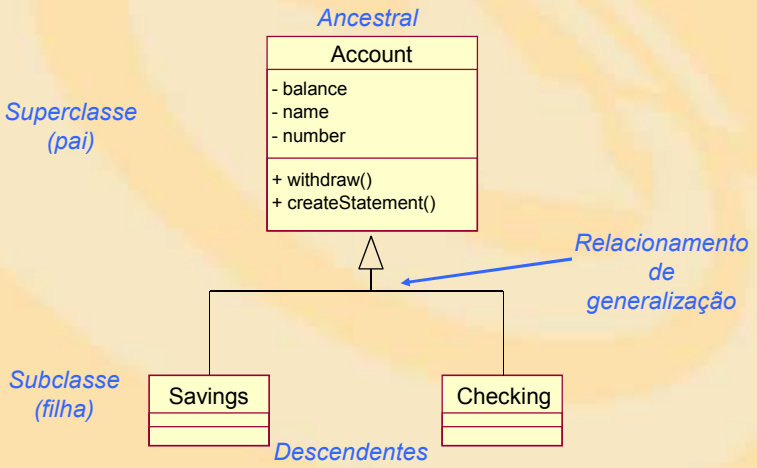
Revisão: O que é Generalização?

- Um relacionamento entre classes onde uma classe compartilha sua estrutura e/ou seu comportamento de uma ou mais classes.
- Define uma hierarquia de abstrações onde a subclasse herda elementos de uma ou mais superclasses.
  - Herança simples
  - Herança múltipla
- É um relacionamento "é-um".

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

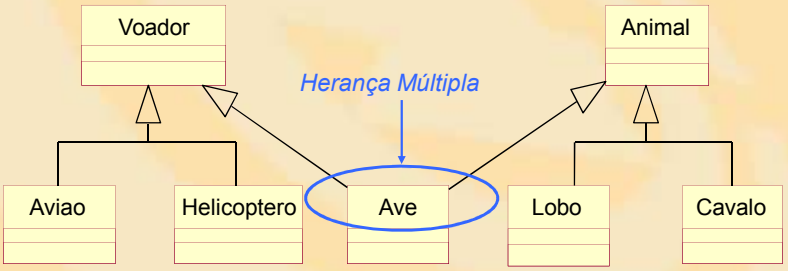
Exemplo: Herança simples

Uma classe herda elementos de outra



Exemplo: Herança Múltipla

Uma classe pode ser herdeira de várias outras



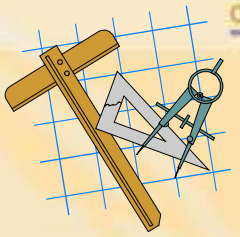
Utilize herança múltipla somente se for necessário e sempre com muita atenção!

Revisão

- O que um diagrama de classes representa?
- Qual benefício os pacotes propiciam ao modelo?
- Defina associação, agregação e generalização.
- Como encontramos associações?
- O que é multilicidade? Quais informações a multiplicidade fornece ao modelador?

PUC • SP COGEAE

Exercício



Dado:

- Um conjunto de classes e seus relacionamentos

Desenhe:

- Um diagrama de classes

PUC • SP COGEAE

Dúvidas?

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Modelagem de Software Orientado a Objetos

Parte 2- Princípios de Modelagem UML
Outros Diagramas da UML

Prof. Dr. Maurício Nacib Pontuschka
tuska@pucsp.br

Objetivos

- Demonstrar como ler e interpretar um:
  - Diagrama de máquina de estados
  - Diagrama de componentes
  - Diagrama de Implantação

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Onde Estamos?

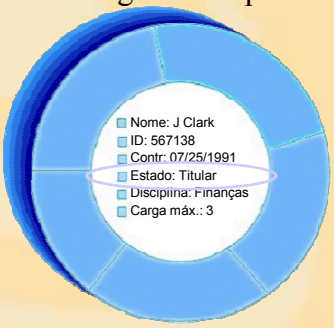
- **Diagrama de máquina de estados**
- Diagrama de componentes
- Diagrama de implantação

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Revisão: Um objeto possui estados

- Estado é a condição ou situação durante a vida de um objeto, a qual satisfaz alguma condição, executa alguma atividade ou aguarda por algum evento.
- O estado de um objeto normalmente muda ao longo do tempo.

Nome: J Clark
ID: 567138
Data Contratação: July 25, 1991
Estado: Titular
Disciplina: Finanças
Carga máxima: 3 classes



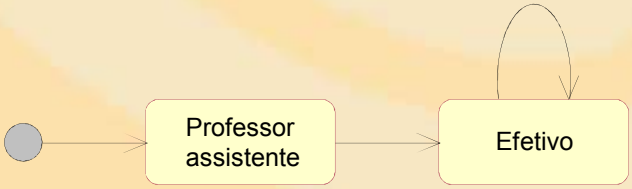
Professor Clark

Exemplo: Professor

- Existe uma seqüência de eventos entre um instrutor se tornar um professor universitário.
  - Professor assistente (será efetivo através de um número de publicações de qualidade)
  - Professor Efetivo/Associado
  - Titular
    (baseado na sua contribuição acadêmica)

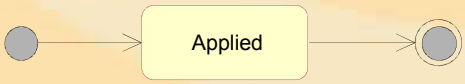
O que são diagramas de máquina de estados?

- Um diagrama de máquina de modela o comportamento dinâmico.
- Ele especifica a seqüência de estados nos quais um objeto pode existir:
  - Os eventos e condições que levam o objeto a alcançar tais estados
  - As ações acionadas quando tais estados são alcançados



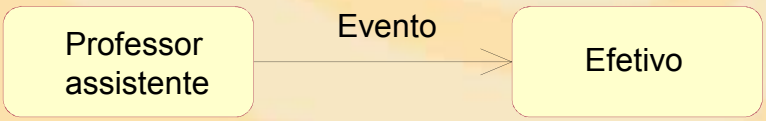
Estados especiais

- O estado inicial é o estado que o objeto se encontra quando ele é criado.
  - Um estado inicial é obrigatório
  - Só pode haver um estado inicial.
  - O estado inicial é representado por um círculo sólido.
- Um estado final indica o final do ciclo de vida de um objeto.
  - Um estado final é opcional.
  - Um estado final é indicado por um círculo sólido circuscrito por um outro círculo vazado.
  - Pode haver mais de um estado final.



O que são eventos?

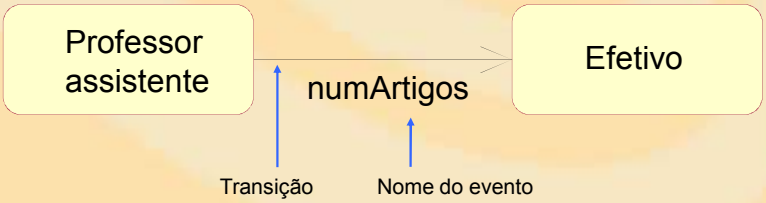
- Um evento é uma especificação de uma ocorrência significante que possui uma localização no tempo e no espaço.
  - Um evento é uma ocorrência de um estímulo que pode disparar uma transição de estado.
  - Exemplo:
    - Publicação com sucesso de um certo número de artigos



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

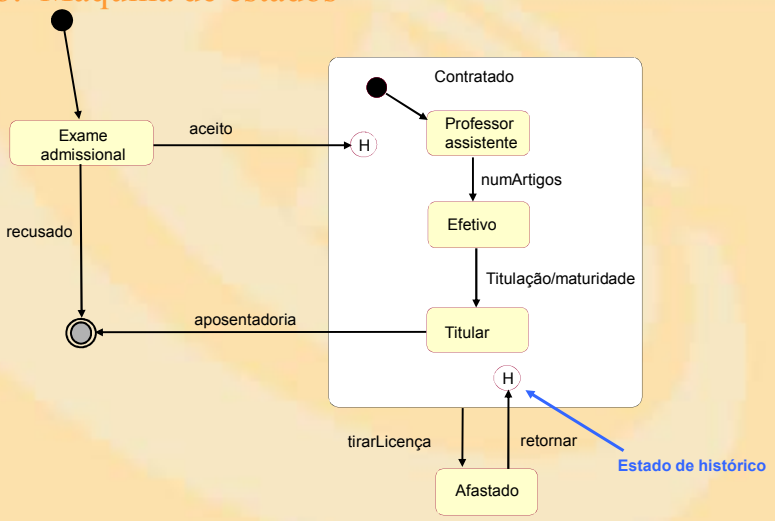
O que são transições?

- Uma transição é a mudança de um estado original para um estado sucessor como resultado de algum estímulo.
  - O estado sucessor pode ser inclusive o próprio estado original.
- A transição pode ser a resposta a um evento.
- Transições podem ser rotuladas com nomes de eventos.



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Exemplo: Máquina de estados



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Onde Estamos?

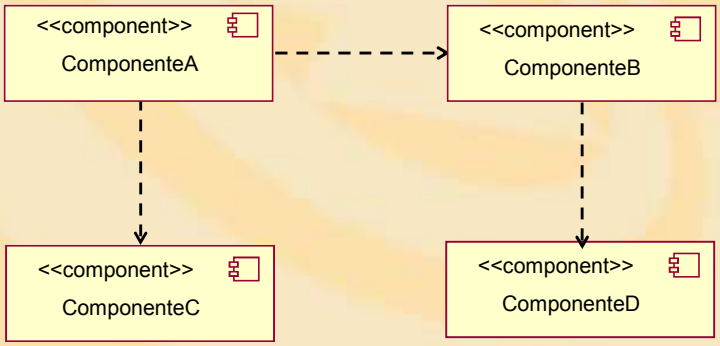
- Diagrama de máquina de estados
- **Diagrama de componentes**
- Diagrama de implantação

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

O que é um diagrama de componentes?

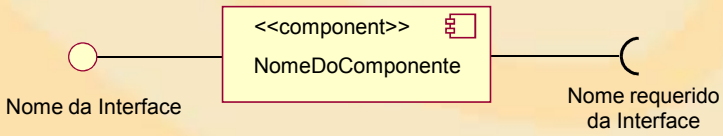
- Um diagrama que mostra a organização e dependências entre componentes



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

O que é um componente?

- Parte modular de um sistema a qual esconde sua implementação por trás de um conjunto de interfaces externas.
  - Parte de um sistema lógico ou físico
- Se adequa e fornece a realização física de um conjunto de interfaces.
- Especifica a dependência física para as interfaces que a requerem.



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Onde Estamos?

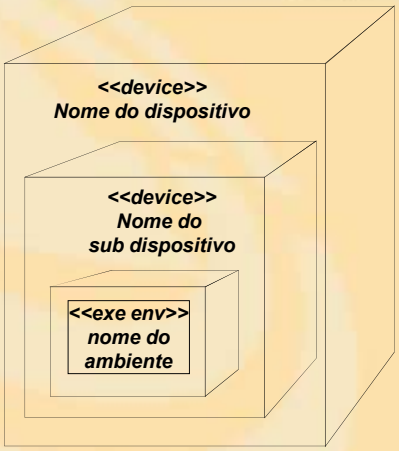
- Diagrama de máquina de estados
- Diagrama de componentes
- **Diagrama de implantação**

O que é um diagrama de implantação?

- Um diagrama de implantação mostra:
  - Configuração dos nós de processo em tempo de execução
  - Ligações de comunicação entre os nós
  - Artefatos de implantação

O que é um nó?

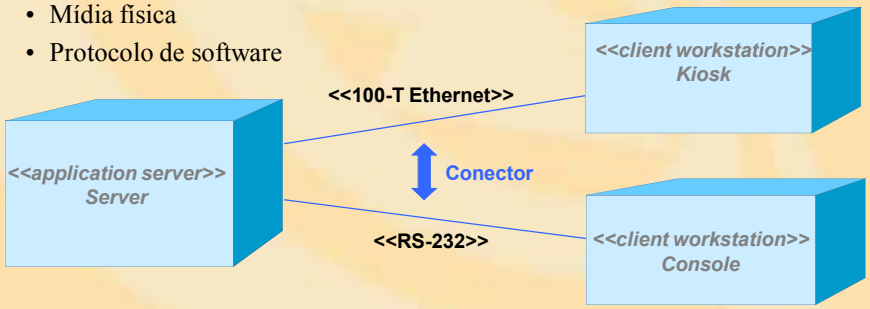
- Representa um recurso computacional em tempo de execução
  - Normalmente possui a memória mínima e capacidade de processamento.
- Tipos:
  - Dispositivo
    - Recurso computacional físico com sua capacidade de processamento.
    - Pode ser atachado.
  - Ambiente de execução
    - Representa plataformas particulares de execução.



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

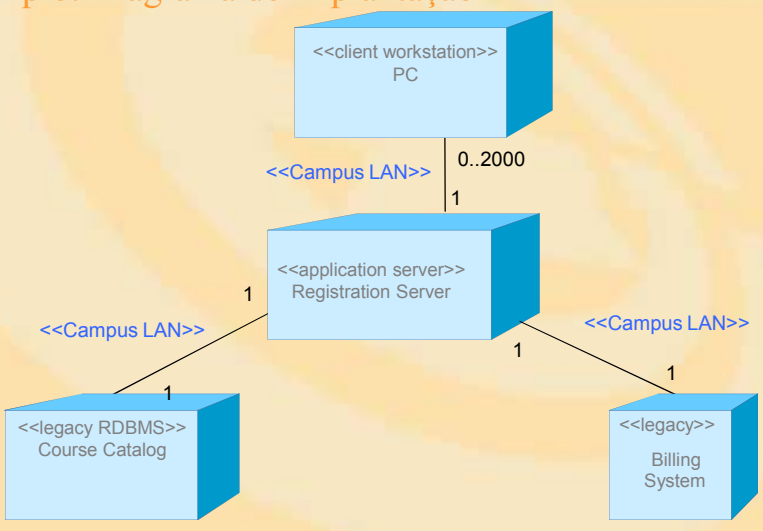
O que é um conector?

- Um conector representa um:
  - Mecanismo de comunicação
    - Mídia física
    - Protocolo de software



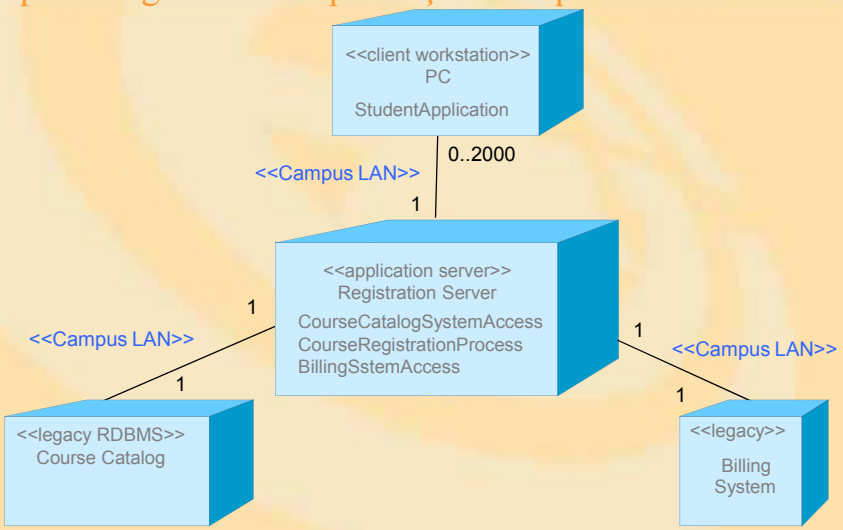
Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Exemplo: Diagrama de Implantação



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Exemplo: Diagrama de Implantação com processos



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Revisão

- Defina estado. Como você determina classes com estados significantes?
- O que é um diagrama de máquina de estados? Descreva as diferentes partes do diagrama.
- O que é um diagrama de componentes?
- Qual é a proposta do diagrama de implantação?

Dúvidas?

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Modelagem de Software Orientado a Objetos

Parte 2- Princípios de Modelagem UML

Padrões de Projeto

(*Design Patterns*)

Prof. Dr. Maurício Nacib Pontuschka
tuska@pucsp.br

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Objetivos

- Entender o que são Padrões de Projeto
- Conhecer alguns dos padrões definidos pelo GoF (*Gang of Four – por conta de serem quatro autores*)
  - Categorias de Padrões de Projeto
  - Apresentação de alguns padrões de projeto (*Composite, Observer, Strategy, Factory, Mediator e Façade*)

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Onde Estamos?

- **Categorias de Padrões de Projeto**
- Padrão: Composite
- Padrão: Observer
- Padrão: Strategy
- Padrão: Factory Method
- Padrão: Mediator
- Padrão: Façade

Categorias de Padrões de Projeto?

- É da natureza do desenvolvimento de software o fato de que os mesmos problemas tendem a acontecer diversas vezes.
- Padrões de projeto são formas padronizadas para atacar problemas conhecidos.

ver mais detalhers em
Design Patterns, Eric Gamma

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Categorias de Padrões de Projeto?

No livro Design Patterns os autores catalogaram 23 padrões de projeto. Estes padrões foram divididos em 3 categorias: Criacionais, Estruturais e Comportamentais.

CRIACIONAIS	Procuram separar a operação de uma aplicação de como os seus objetos são criados.
ESTRUTURAIS	Provêem generalidade para que a estrutura da solução possa ser estendida no futuro.
COMPORTAMENTAIS	Utilizam herança para distribuir o comportamento entre subclasses, ou agregação e composição para construir comportamento complexo a partir de componentes simples.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Categorias de Padrões de Projeto?

Os padrões definidos pelos autores do livro Design Patterns são:

CRIACIONAIS	Abstract Factory; Builder; Factory Method; Propotype; Singleton
ESTRUTURAIS	Adapter; Bridge; Composite; Decorator; Façade; Flyweight; Proxy
COMPORTAMENTAIS	Chain of Responsibility; Command; Interpreter; Iterator; Madiator; Memento; Observer; State; Strategy; Template Method; Visitor

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Onde Estamos?

- Categorias de Padrões de Projeto
- **Padrão: Composite**
- Padrão: Observer
- Padrão: Strategy
- Padrão: Factory Method
- Padrão: Mediator
- Padrão: Façade

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

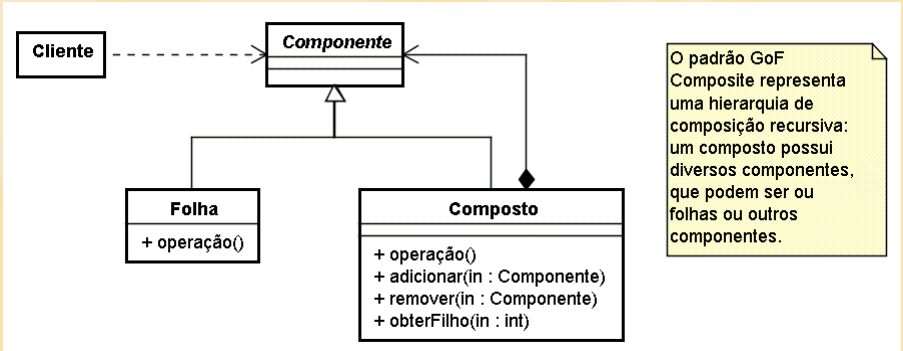
Padrão: Composite (Estrutural)

- Se propõe a definir uma relação hierárquica entre objetos de tal forma que tanto o objeto todo quanto os objetos parte sejam equivalentes em certos aspectos.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Padrão: Composite (Estrutural)



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Onde Estamos?

- Categorias de Padrões de Projeto
- Padrão: Composite
- **Padrão: Observer**
- Padrão: Strategy
- Padrão: Factory Method
- Padrão: Mediator
- Padrão: Façade

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

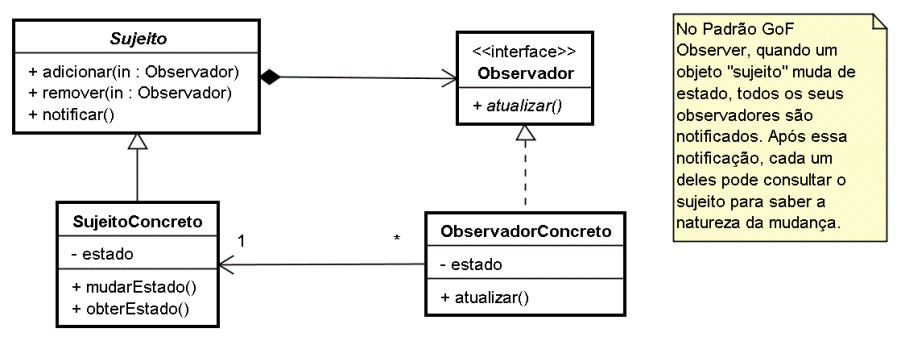
PUC • SP COGEAE

Padrão: Observer (Comportamental)

Possui o objetivo de definir de forma flexível uma dependência um para muitos entre objetos. Esta dependência é no sentido de que, se houver alguma modificação no estado do objeto central, os objetos dependentes devem ser notificados. A preocupação aqui é com o acoplamento: necessitamos que o objeto central seja capaz de enviar mensagens de notificação sem, no entanto, conhecê-los diretamente.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Padrão: Observer (Comportamental)



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Onde Estamos?

- Categorias de Padrões de Projeto
- Padrão: Composite
- Padrão: Observer
- **Padrão: Strategy**
- Padrão: Factory Method
- Padrão: Mediator
- Padrão: Façade

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

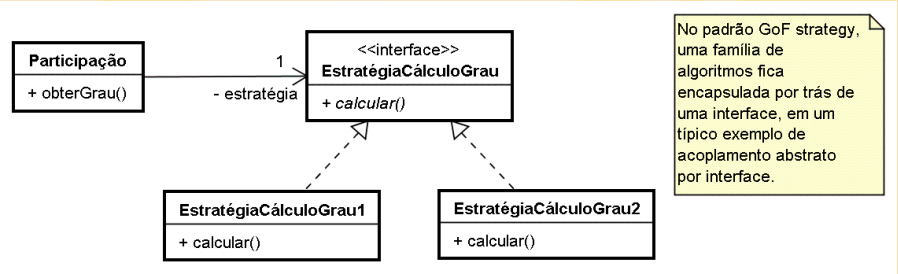
Padrão: Strategy (Comportamental)

Tem o objetivo de encapsular diferentes algoritmos para realização de alguma tarefa computacional por trás de uma interface e permitir que a região de código cliente dessa tarefa possa utilizar qualquer desses algoritmos sem precisar ser modificada.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Padrão: Strategy (Comportamental)



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Onde Estamos?

- Categorias de Padrões de Projeto
- Padrão: Composite
- Padrão: Observer
- Padrão: Strategy
- **Padrão: Factory Method**
- Padrão: Mediator
- Padrão: Façade

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

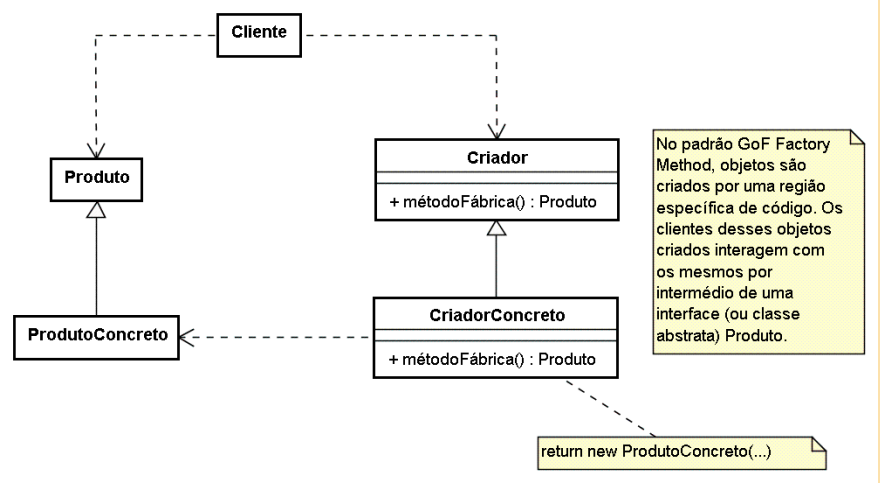
PUC • SP COGEAE

Padrão: Factory Method

Possui o objetivo de criar uma forma de instanciar objetos sem que a região do código que realiza a instanciação do objeto fique acoplada à classe e sua forma específica de prover o serviço requerido. Este serviço de instanciação, por vezes, pode ser muito complexo.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Padrão: Factory Method



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Onde Estamos?

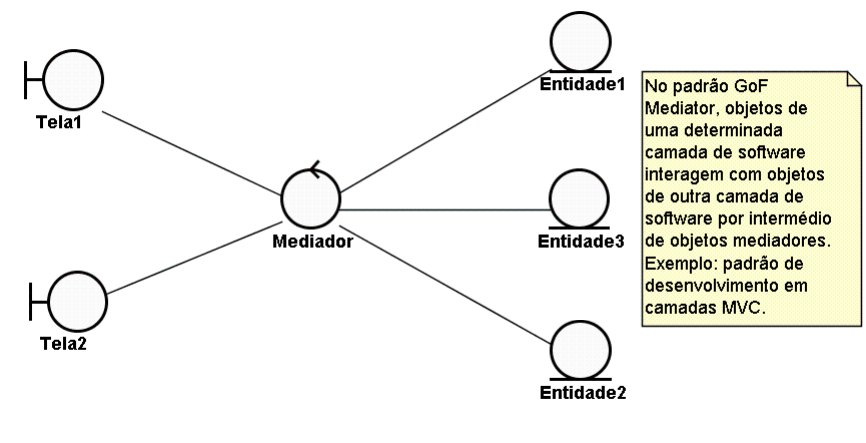
- Categorias de Padrões de Projeto
- Padrão: Composite
- Padrão: Observer
- Padrão: Strategy
- Padrão: Factory Method
- **Padrão: Mediator**
- Padrão: Façade

Padrão: Mediator

Permite um grupo de objetos interagir com outro grupo de objetos ao mesmo tempo que mantém um acoplamento fraco. A solução é a criação de um objeto mediador para encapsular interações transferindo as requisições de um grupo para o outro. (*exemplo: objetos Controller do MVC*)

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

Padrão: Mediator



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Onde Estamos?

- Categorias de Padrões de Projeto
- Padrão: Composite
- Padrão: Observer
- Padrão: Strategy
- Padrão: Factory Method
- Padrão: Mediator
- **Padrão: Façade**

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

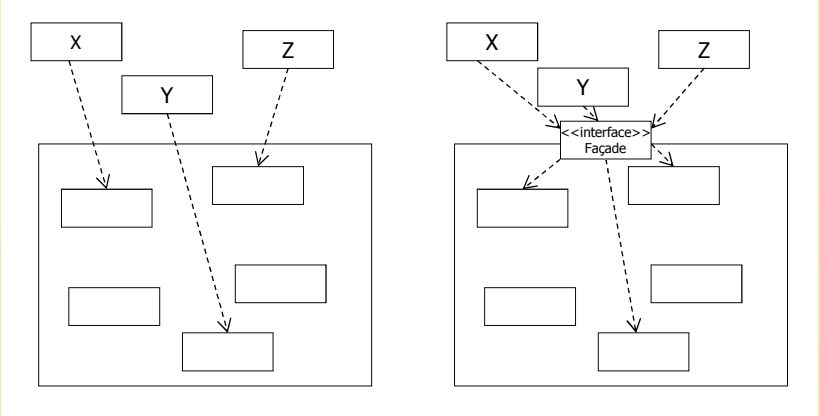
Padrão: Façade

Permite a definição de interfaces de comunicação entre subsistemas deixando-os fracamente acoplados. Define-se um uma interface de alto nível que torna o subsistema mais fácil de ser utilizado de tal forma que um cliente só se comunique com este subsistema através desta interface.

Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Padrão: Façade



Modelagem de Software Orientado a Objetos
Princípios de Modelagem UML

PUC • SP COGEAE

Dúvidas?

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Modelagem de Software Orientado a Objetos

Parte 3 – Análise de Modelos de Software

Prof. Dr. Maurício Nacib Pontuschka
tuska@pucsp.br

Objetivos

- Entender os vínculo entre os principais modelos da UML em um contexto de um processo de desenvolvimento de software.
- Conhecer alguns dos principais erros de modelagem em algumas das principais etapas no processo de desenvolvimento de software.
- Praticar a modelagem de software sob uma perspectiva de qualidade e coerência dos modelos na linguagem UML.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Onde estamos?

- **Vínculo entre Modelos**
- Problemas de Modelagem
  - Modelagem de Domínio
  - Modelagem de Casos de Uso
  - Revisão de Requisitos
  - Análise de Robustez
  - Revisão Preliminar de Design
  - Elaboração dos Diagramas de Seqüência
  - Revisão Crítica de Design
- Análise de Modelos UML

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Vínculo entre modelos UML

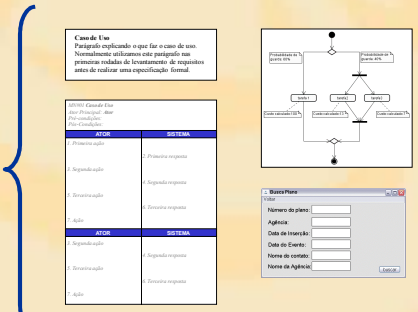
Para esta primeira análise levaremos em conta apenas alguns dos diagramas UML

- Diagramas de casos de uso representam o escopo de sistemas ou partes deste escopo identificando metas ou necessidades dos seus atores.
- Diagramas de classes representam visões estruturais do sistema e especificam as classes de objetos com os quais o sistema irá operar.
- Diagramas de interação representam funcionalidades internas de possíveis interações no sistema em termos de trocas de mensagens entre os objetos da aplicação de forma a dar conta de uma interação do sistema.
- Diagramas de atividades podem ser utilizados para representar a lógica da seqüência de passos para se resolver uma determinada interação existente no sistema.

Casos de Uso

Para cada caso de uso pode haver um conjunto de alguns artefatos relevantes:

- descrição do caso de uso;
- especificação do caso de uso;
- diagrama de atividades;
- esboços de tela.



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

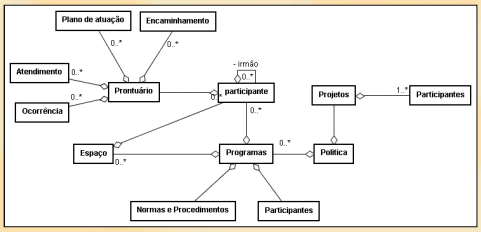
Casos de Uso

Estes artefatos devem manter um alto grau de coerência entre eles. Cada um possui um objetivo específico mas estão contribuindo para o modelo da mesma interação do usuário com o sistema.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Modelo de Domínio

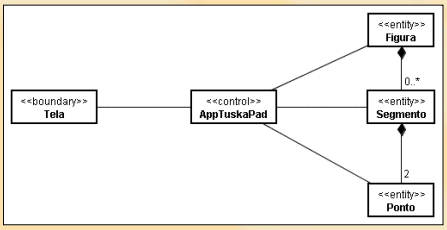
O modelo de domínio representa os conceitos envolvidos no projeto de desenvolvimento do software no que diz respeito ao domínio da aplicação. Estes conceitos são identificados a partir dos artefatos que detalham os casos de uso. Os termos importantes são identificados e representados em um diagrama de classes, que neste momento ainda não apresentará detalhes como os métodos e atributos das classes.

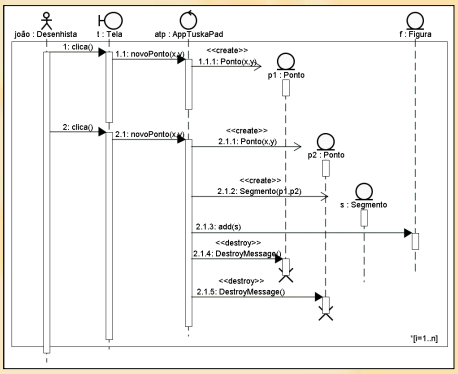


Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Modelo de Domínio X Diagramas de Interação

As classes do modelo de domínio representam ótimas referências de como distribuir os objetos no momento de se elaborar os diagramas de Interações (Diagrama de Seqüência de Mensagens e o Diagrama de Comunicação).

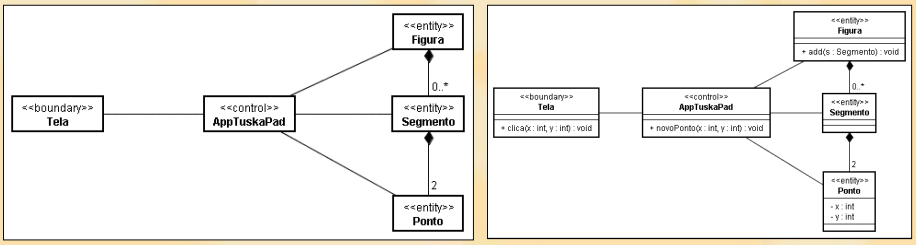




Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Modelo de Domínio X Diagramas de Interação

Após a elaboração dos modelos de Interação, será possível refinar o modelo de classes registrando atributos e métodos identificados nesta etapa.



Refinamento ➡

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Onde estamos?

- Vínculo entre Modelos
- **Problemas de Modelagem**
  - Modelagem de Domínio
  - Modelagem de Casos de Uso
  - Revisão de Requisitos
  - Análise de Robustez
  - Revisão Preliminar de Design
  - Elaboração dos Diagramas de Seqüência
  - Revisão Crítica de Design
- Análise de Modelos UML

Problemas de Modelagem

Uma série de problemas de modelagem foram levantadas e podem ser estudadas com maior profundidade no livro:

"Applying Use Case Driven Object Modeling with UML"

dos autores Doug Rosemberg e Kendall Scott.

Onde estamos?

- Vínculo entre Modelos
- **Problemas de Modelagem**
  - **Modelagem de Domínio**
  - Modelagem de Casos de so
  - Revisão de Requisitos
  - Análise de Robustez
  - Revisão Preliminar de Design
  - Elaboração dos Diagramas de Seqüência
  - Revisão Crítica de Design
- Análise de Modelos UML

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

10 principais erros na Modelagem do Domínio

1. Iniciar descrevendo as multiplicidades para associações logo no começo. Ter certeza de que toda associação possui uma multiplicidade.
2. Realizar análise de substantivos e verbos exaustivamente por todos os documentos de levantamento.
3. Associar operações à classes sem explorar os diagramas de seqüência e de casos de uso.
4. Otimizar o seu código para reuso antes de ter certeza de estar satisfeito com os requisitos do sistema.
5. Debater a respeito de agregação, associação ou composição para cada associação do modelo.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

10 principais erros na Modelagem do Domínio

6. Presumir uma implementação específica sem modelar o contexto do problema.
7. Utilizar nomes de difícil entendimento como cGerPortIntf ao invés de nomes intuitivos e óbvios como Gerenciador de Portifólio.
8. Pular diretamente para construções de implementação como acessibilidade dos relacionamentos e classes parametrizadas.
9. Criar um relacionamento de um para um entre classes e as tabelas de um banco de dados relacional.
10. Realizar uma padronização precoce a qual envolve soluções “legais” as quais possuem pouca ou nenhuma conexão com os problemas do usuário.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Onde estamos?

- Vínculo entre Modelos
- **Problemas de Modelagem**
  - Modelagem de Domínio
  - **Modelagem de Casos de Uso**
  - Revisão de Requisitos
  - Análise de Robustez
  - Revisão Preliminar de Design
  - Elaboração dos Diagramas de Seqüência
  - Revisão Crítica de Design
- Análise de Modelos UML

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

10 principais erros na Modelagem de Casos de Uso

1. Escrever requisitos funcionais ao invés de um cenário textual.
2. Descrever atributos e métodos ao invés de suas utilidades.
3. Escrever casos de uso muito sucintos.
4. Desvincular totalmente a interface com o usuário.
5. Descartar nomes explícitos para objetos de fronteira.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

10 principais erros na Modelagem de Casos de Uso

6. Escrever utilizando a perspectiva diferente da do usuário na voz passiva.
7. Descrever somente interações de usuário; ignorando as respostas do sistema.
8. Omitir texto para ações de fluxos alternativos.
9. Foco em algo diferente do que está dentro do caso de uso, assim como como você chega àquela situação ou o que ocorre depois dela.
10. Perder um mês decidindo quando utilizar "includes" ou "extends".

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Onde estamos?

- Vínculo entre Modelos
- **Problemas de Modelagem**
  - Modelagem de Domínio
  - Modelagem de Casos de Uso
  - **Revisão de Requisitos**
  - Análise de Robustez
  - Revisão Preliminar de Design
  - Elaboração dos Diagramas de Seqüência
  - Revisão Crítica de Design
- Análise de Modelos UML

10 principais erros na Revisão de Requisitos

1. Não revisar requisitos. Ao invés disto deixar que os programadores construam o que quiserem.
2. Não ter certeza de que o texto do caso de uso se encaixa com o comportamento desejado do sistema.
3. Não utilizar nenhum tipo de protótipo de tela a fim de validar o comportamento do sistema.
4. Manter seus casos de uso em um grau tão alto de abstração que seus clientes (não tecnicos) não possuem nem uma pista do que o caso de uso trata.
5. Não garantir que o modelo de domínio realmente reflete objetos do mundo real.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

10 principais erros na Revisão de Requisitos

6. Não ter certeza de que o texto do caso de uso referencia a objetos do domínio.
7. Não questionar casos de uso sem fluxo alternativo algum.
8. Não questionar cada ação do fluxo principal dos casos de uso a respeito de possíveis fluxos alternativos.
9. Não se preocupar se os casos de uso estão escritos na voz passiva.
10. Não se preocupar se um caso de uso possui 4 páginas de comprimento.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Onde estamos?

- Vínculo entre Modelos
- **Problemas de Modelagem**
  - Modelagem de Domínio
  - Modelagem de Casos de Uso
  - Revisão de Requisitos
  - **Análise de Robustez**
  - Revisão Preliminar de Design
  - Elaboração dos Diagramas de Seqüência
  - Revisão Crítica de Design
- Análise de Modelos UML

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

4 regras dos Diagramas de Robustez

1. Atores só podem falar com objetos de fronteira
2. Objetos de fronteira só falam com controles e atores.
3. Objetos de entidade só falam com controles.
4. Controles podem falar com objetos de fronteira, objetos de entidade e outros controles mas nunca com atores

D. Rosenberg, "Use Case DrivenObjectModelingwithUML: A PracticalApproach"

10 principais erros na Análise de Robustez

1. Violar uma ou mais regras do diagrama de robustez.
2. Não utilizar análise de robustez para auxiliar o uso de um formato consistente dos textos do caso de uso.
3. Não incluir fluxos alternativos nos diagramas de robustez.
4. Não utilizar análise de robustez para garantir consistência entre os nomes de classe do diagrama de classes e no texto dos casos de uso.
5. Alocar comportamentos a classes nos diagramas de robustez.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

10 principais erros na Análise de Robustez

6. Incluir muito poucos ou controladores demais.
7. Gastar muito tempo tentando deixar o diagrama de robustez perfeito.
8. Tentar realizar um desenvolvimento detalhado nos diagramas de robustez.
9. Não realizar simulações visuais entre o texto caso de uso e o diagrama de robustez.
10. Não atualizar o seu modelo estático.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Onde estamos?

- Vínculo entre Modelos
- **Problemas de Modelagem**
  - Modelagem de Domínio
  - Modelagem de Casos de Uso
  - Revisão de Requisitos
  - Análise de Robustez
  - **Revisão Preliminar de Design**
  - Elaboração dos Diagramas de Seqüência
  - Revisão Crítica de Design
- Análise de Modelos UML

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

10 principais erros na Revisão Preliminar de Design

1. Não garantir que os clientes sabem que esta é a última chance de alterar o comportamento antes que uma versão do sistema seja construída.
2. Não garantir que os textos dos casos de uso e os diagramas de robustez estejam em conformidade.
3. Não garantir que um novo objeto de entidade foram adicionados ao modelo de domínio.
4. Não procurar por atributos nas classes de domínio.
5. Esperar que operações sejam alocadas às classes durante a revisão preliminar do design.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

10 principais erros na Revisão Preliminar de Design

6. Não advertir seus clientes (novamente) que os textos dos casos de uso representam um contrato entre os desenvolvedores e os clientes.
7. Desejar que a desenho estático preliminar realize um uso extensivo de padrões de projeto.
8. Não revisar as regras dos diagramas de robustez.
9. Esperar que os diagramas de robustez mostrem um projeto completo e detalhado e não um projeto conceitual.
10. Revisar a direção de cada seta em um diagrama de robustez cuidadosamente ao invés de realizar uma rápida ligação para verificar se foi levado em conta todo o comportamento necessário.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Onde estamos?

- Vínculo entre Modelos
- **Problemas de Modelagem**
  - Modelagem de Domínio
  - Modelagem de Casos de Uso
  - Revisão de Requisitos
  - Análise de Robustez
  - Revisão Preliminar de Design
  - **Elaboração dos Diagramas de Seqüência**
  - Revisão Crítica de Design
- Análise de Modelos UML

10 principais erros na elaboração dos Diagramas de Seqüência

1. Não realizar um diagrama de seqüência para cada caso de uso.
2. Não colocar o texto do caso de uso no diagrama de seqüência.
3. Não identificar todos os objetos necessários primeiro, em um diagrama de robustez.
4. Não realizar uma conferência visual entre o texto do caso de uso e as setas de mensagem.
5. Não mostrar os meandros; ao invés disso, manter os diagramas de seqüência em um alto grau de abstração.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

10 principais erros na elaboração dos Diagramas de Seqüência

6. Transformar o seu diagrama de seqüência em um fluxograma ao invés de utilizá-lo para alocar comportamento aos objetos.
7. Não focar em métodos interessantes (comportamento real do software), como os métodos gets e sets.
8. Não pensar cuidadosamente a respeito das origens da mensagem (qual o objeto que está no controle).
9. Não acompanhar os princípios básicos de orientação a responsabilidade quando estiver alocando comportamentos ao definir setas de mensagens.
10. Não atualizar o modelo estático durante a construção do diagrama de classes para cada pacote de casos de uso.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Onde estamos?

- Vínculo entre Modelos
- **Problemas de Modelagem**
  - Modelagem de Domínio
  - Modelagem de Casos de Uso
  - Revisão de Requisitos
  - Análise de Robustez
  - Revisão Preliminar de Design
  - Elaboração dos Diagramas de Seqüência
  - **Revisão Crítica de Design**
- Análise de Modelos UML

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Quatro critérios básicos de uma boa classe (Halbert e O'Brien)

1. **Reusabilidade**: Quanto mais genéricos forem os objetos e classes, maior a probabilidade de reuso.
2. **Aplicabilidade**: Coerência das classes com seus respectivos métodos.
3. **Complexidade**: Diz respeito a facilidade de construção das classes de acordo com a alocação dos métodos.
4. **Conhecimento para implementação**: Quanto da implementação do comportamento da classe depende de detalhes internos aos métodos associados.

Critérios gerais dos relacionamentos de classes

1. **Acoplamento**: medida da força das conexões entre classes
2. **Coesão**: medida a força da conexão entre os atributos e os métodos de uma classe.
3. **Suficiencia**: condição em que uma classe encapsula abstrações suficientes para oferecer algo eficiente e significativo.
4. **Completude**: condição em que uma dada interface de uma classe captura todas as abstrações relevantes.
5. **Primitividade**: condição em que a operação pode ser eficientemente construída através da colaboração entre classes de forma construtiva.

10 principais erros na Revisão Crítica de Design

1. Convidar clientes sem o perfil técnico para a revisão de projeto.
2. Não verificar os textos dos casos de uso cuidadosamente relacionando-os com o conjunto de diagramas de seqüência.
3. Não verificar a origem e o destino de cada mensagem em cada diagrama de seqüência cuidadosamente.
4. Não pensar através dos critérios de Halbert/O'Brien ao revisar os diagramas de seqüência.
5. Não revisar os modelos estáticos por critérios de qualidade de classes.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

10 principais erros na Revisão Crítica de Design

6. Não se preocupar a respeito de descrever os meandros do software; deixar que isto ocorra naturalmente ao longo do projeto.
7. Não considerar utilidade da utilização de padrões de projeto no projeto.
8. Mostrar diagramas de seqüência genéricos os quais desconsideram a implementação de tecnologias como DCOM ou EJBs.
9. Não revisar os diagramas de seqüência para cada cenário presente na atual versão do software.
10. Não se preocupar a respeito de detalhes do projeto antes de iniciar a codificação. Assumindo então que a reengenharia a partir do código irá consertar tudo posteriormente.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Onde estamos?

- Vínculo entre Modelos
- Problemas de Modelagem
  - Modelagem de Domínio
  - Modelagem de Casos de Uso
  - Revisão de Requisitos
  - Análise de Robustez
  - Revisão Preliminar de Design
  - Elaboração dos Diagramas de Seqüência
  - Revisão Crítica de Design
- **Análise de Modelos UML**

Exercício de análise de modelagem

Modele um sistema simples de desenho de retas na tela. Um desenhista clica sobre uma tela identificando pontos e a cada 2º ponto identificado o software deve criar uma reta interligando as duas posições definidas pelo usuário. O software não prevê uma condição de término.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Exercício de análise de modelagem

Troque o seu modelo com um colega da turma e avalie os relacionamento entre os modelos apresentados.

Devolva o modelo ao seu colega juntamente com os seus comentários.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Resumo

- Qual a relação entre um caso de uso e um diagrama de atividades?
- Por que não é desejado incluir métodos e atributos antes do desenho dos diagramas de interação?
- Qual o objetivo geral deste conjunto de diagramas?
- O que representa o diagrama de classes neste contexto?
- Estas etapas de desenvolvimento são determinadas pela UML?

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Dúvidas?

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Modelagem de Software Orientado a Objetos

Parte 3 – Análise de Modelos de Software

Prof. Dr. Maurício Nacib Pontuschka
tuska@pucsp.br

Objetivos

- Exercitar a elaboração de modelos de software explorando características sintáticas e semânticas nos diagramas da UML.
- Interpretar modelos diagramados com a UML.
- Destacar erros de sintaxe e de semântica em diagramas UML mais especificamente em diagramas de casos de uso e diagramas de seqüência de mensagens.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

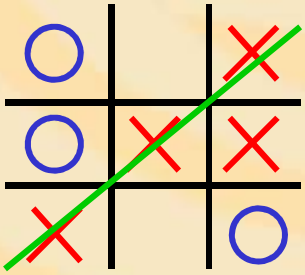
Onde estamos?

- **Exercício 1 – Jogo da Velha**
- Alguns Erros de Diagramação
- Restrições de Relacionamentos
- Exercício 2 – Forum de Notícias
- Exercício 3 – Jogo Sudoku

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Jogo da Velha

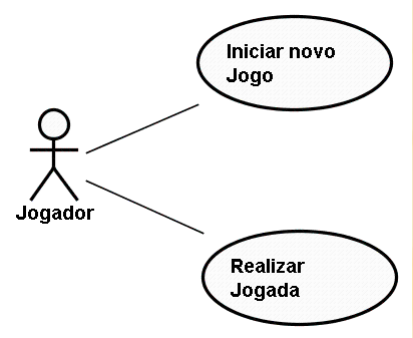
O objetivo deste exercício é realizar a modelagem de um software por meio da UML de forma a implementar o mundialmente conhecido Jogo da Velha.

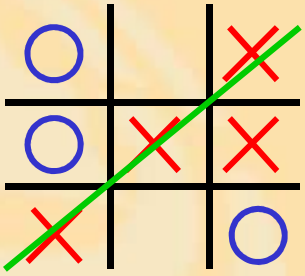


PUC • SP COGEAE

Jogo da Velha

Diagrama de casos de uso





PUC • SP COGEAE

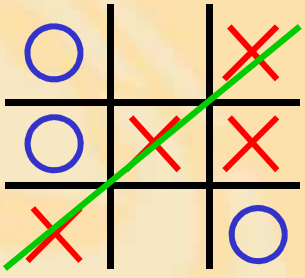
Jogo da Velha

Descrição de casos de uso

MN#: Iniciar novo jogo
Ator: Jogador
Pré-Condições: nenhuma
Pós-condições: jogo iniciado

FLUXO PRINCIPAL

1 – O jogador solicita iniciar o jogo
2 – O sistema apresenta um tablado sem nenhuma marca para o início do jogo



PUC • SP COGEAE

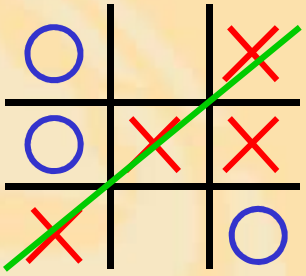
Jogo da Velha

Descrição de casos de uso

MN#: Realizar jogada
Ator: Jogador
Pré-Condições: jogo iniciado
Pós-condições: jogada realizada

FLUXO PRINCIPAL

1 – O jogador seleciona uma posição do tablado
2 – O sistema verifica se a posição está livre
3 – Caso a posição esteja livre a marca do jogador (O ou X) será colocada na posição solicitada.
4 – O sistema verifica se o jogo terminou.
5 – Se não terminou, o sistema alterna a vez do jogador.



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Jogo da Velha

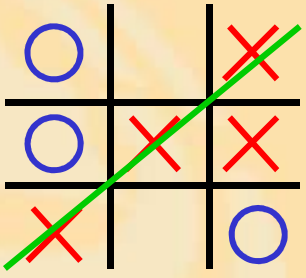
Descrição de casos de uso

FLUXO ALTERNATIVO 1

3 – O sistema solicita outra posição ao jogador e o processo volta ao passo 1 do fluxo principal.

FLUXO ALTERNATIVO 2

4 – Se algum jogador alinhou 3 marcas ou não existirem mais posições livres o sistema informa o final do jogo.

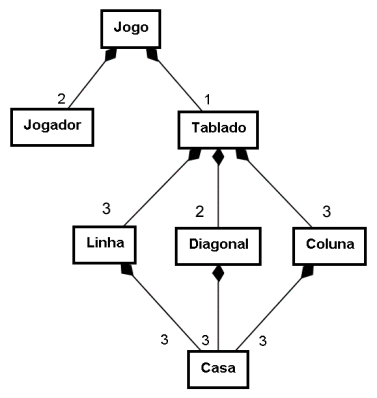


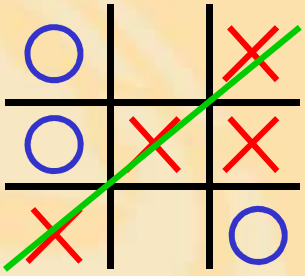
Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Jogo da Velha

Modelo Conceitual



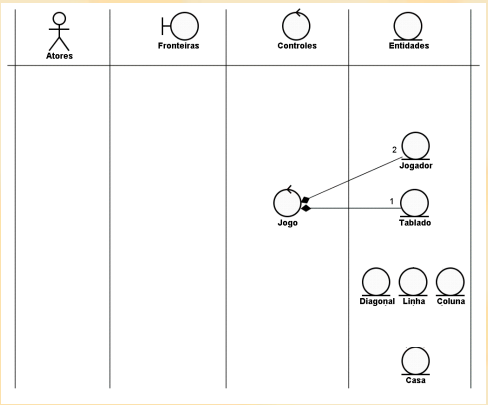


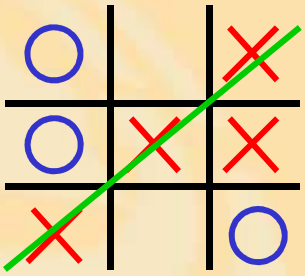
Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Jogo da Velha

Diagrama de Robustez



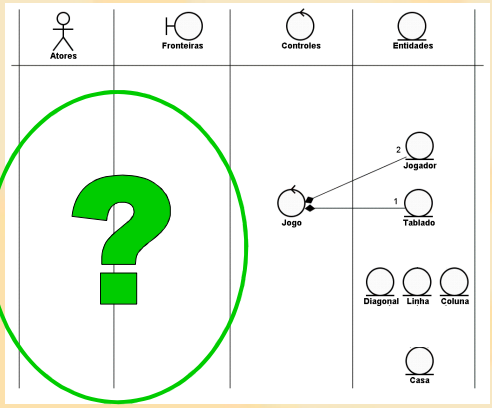


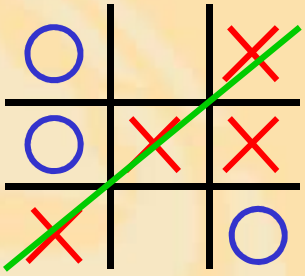
Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Jogo da Velha

Diagrama de Robustez



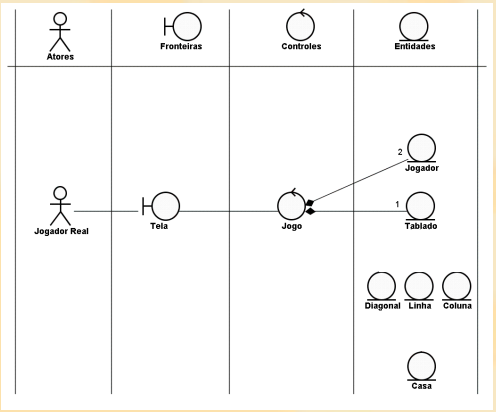


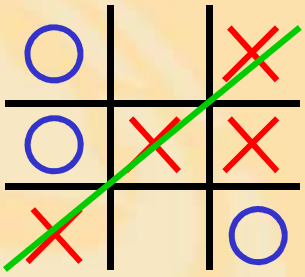
Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Jogo da Velha

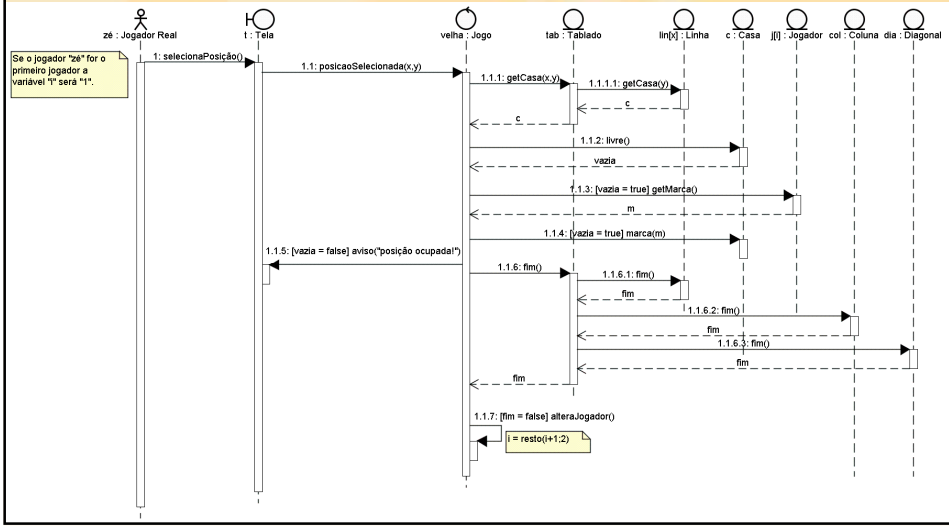
Diagrama de Robustez





Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Jogo da Velha Diagrama de Seqüência de Mensagens



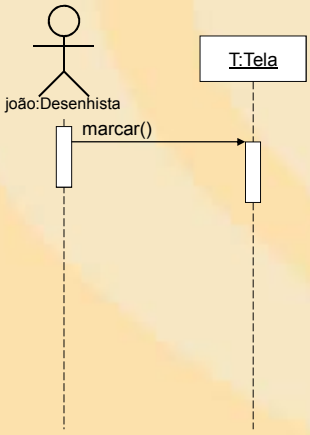
Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Onde estamos?

- Exercício 1 – Jogo da Velha
- **Alguns erros de Diagramação**
- Restrições de Relacionamentos
- Exercício 2 – Forum de Notícias
- Exercício 3 – Jogo Sudoku

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

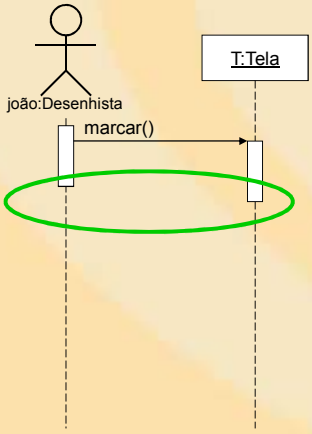
Erro de sincronismo de processos em diagramas de seqüência de mensagens



Qual o erro presente no diagrama ao lado?

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Erro de sincronismo de processos em diagramas de seqüência de mensagens

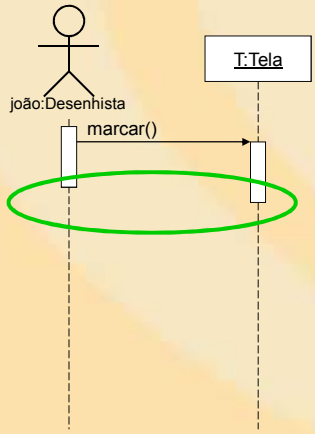


Qual o erro presente no diagrama ao lado?

O processo marcar() é iniciado por outro processo e deve estar aninhado em relação a ele. Note que neste diagrama o processo marcar() termina depois do processo que o chamou.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

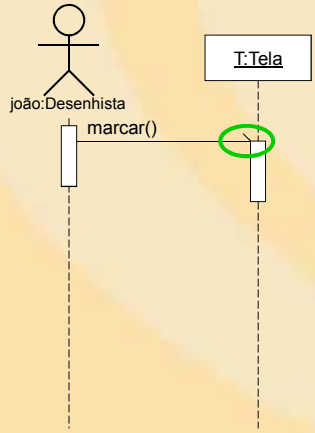
Erro de sincronismo de processos em diagramas de seqüência de mensagens



Como resolver o problema?

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Erro de sincronismo de processos em diagramas de seqüência de mensagens

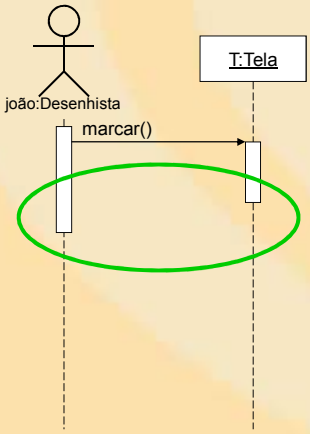


Como resolver o problema?

1) Se o processo marcar() necessariamente deve finalizar após a finalização do processo que o chamou, isto significa que a chamada deveria ser assíncrona!

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Erro de sincronismo de processos em diagramas de seqüência de mensagens

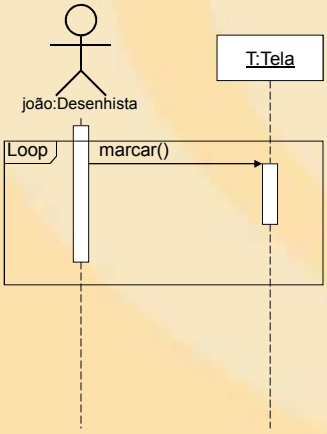


Como resolver o problema?

2) Normalmente o problema está em definir realmente que o processo marcar() deve terminar antes do processo que o chamou. Portanto devemos prolongar o tempo de execução do processo que emitiu a mensagem até que todas as mensagens que ele chamou tenham finalizado.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

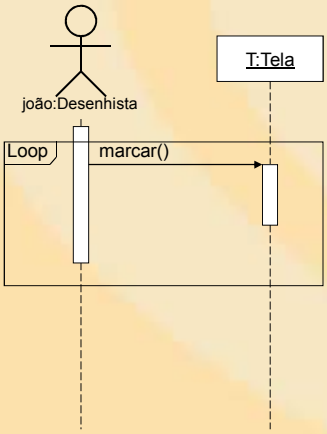
Erro na especificação de loops ou blocos restritivos



Qual o erro presente no diagrama ao lado?

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Erro na especificação de loops ou blocos restritivos

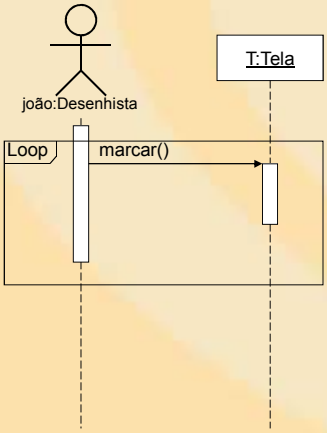


Qual o erro presente no diagrama ao lado?

Note que o bloco restritivo inicia no meio de um processo que é encerrado dentro do próprio bloco. Um processo não pode finalizar várias vezes!

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

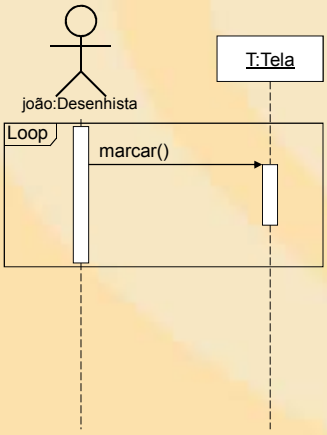
Erro na especificação de loops ou blocos restritivos



Como resolver o problema?

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Erro na especificação de loops ou blocos restritivos

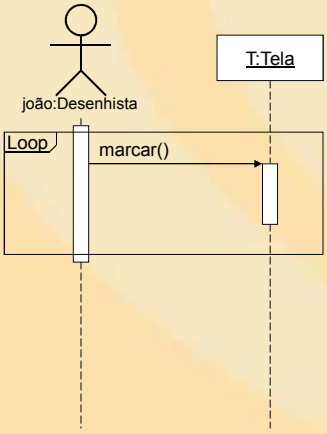


Como resolver o problema?

Devemos garantir que o início e o término do bloco de restrição ou estejam inteiramente dentro de um processo ou os processos estejam inteiramente dentro do bloco.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Erro na especificação de loops ou blocos restritivos

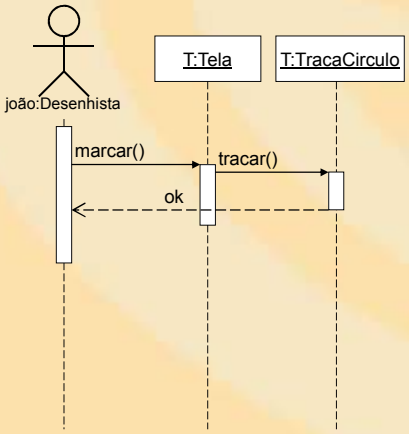


Como resolver o problema?

Devemos garantir que o início e o término do bloco de restrição ou estejam inteiramente dentro de um processo ou os processos estejam inteiramente dentro do bloco.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

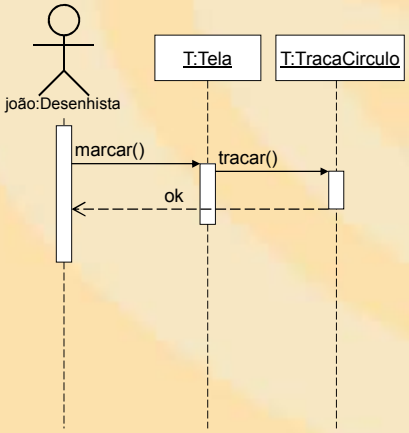
Erro no retorno de mensagens



Qual o erro presente no diagrama ao lado?

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Erro no retorno de mensagens

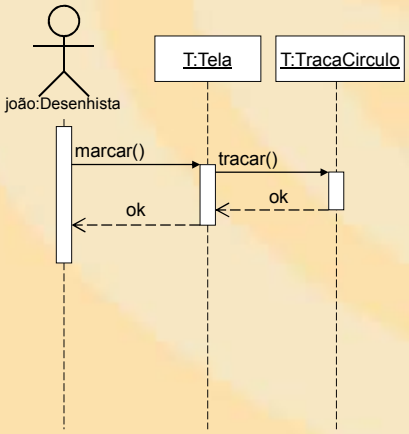


Qual o erro presente no diagrama ao lado?

O retorno do processo é enviado a outro objeto que não o que enviou a mensagem.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Erro no retorno de mensagens

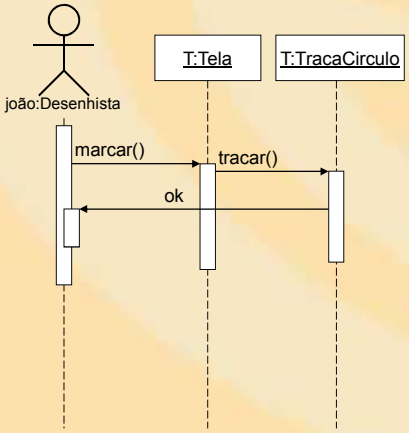


Como resolver o problema?

O retorno remete a resposta para o objeto que o chamou e este objeto retransmite a mensagem.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Erro no retorno de mensagens



Como resolver o problema?

Outra forma é ao invés de enviar a resposta através do retorno, envie uma nova mensagem.

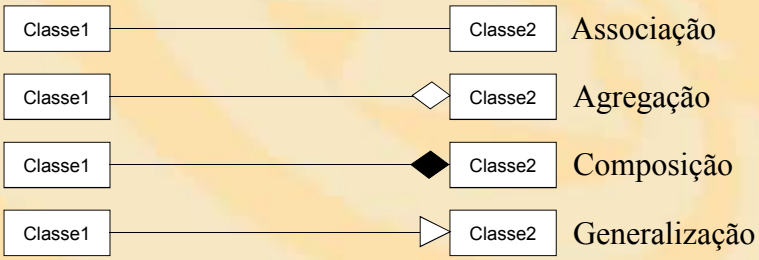
Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Onde estamos?

- Exercício 1 – Jogo da Velha
- Alguns Erros de Diagramação
- **Restrições de Relacionamentos**
- Exercício 2 – Forum de Notícias
- Exercício 3 – Jogo Sudoku

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Semântica em relacionamento de classes



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Semântica em relacionamento de classes

Associação

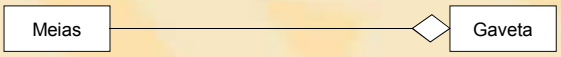
As classes vinculadas por uma associação podem trocar mensagens.

Exemplo:

Estudante fala com a Secretária

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Semântica em relacionamento de classes



Agregação

Um relacionamento de agregação indica que objetos de uma determinada classe possui objetos de outra classe.

Exemplo:

Gaveta possui Meias

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Semântica em relacionamento de classes

Composição

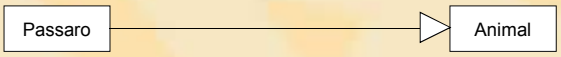
Um relacionamento de composição indica que o objetos de uma determinada classe são partes de objetos de uma outra classe.

Exemplo:

Assento é parte de uma Cadeira

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Semântica em relacionamento de classes



Generalização

Um relacionamento de generalização indica uma relação de herança entre classes e especifica que o objetos de uma determinada classe são objetos de uma outra classe.

Exemplo:

Pássaro é um Animal

Restrições em relacionamentos

Além de especificar os relacionamentos, nós podemos especificar restrições em relacionamentos.

Repare as afirmações:

"Uma gaveta pode conter meias"

"Um menino pode ser mordido por cachorros"

Como podemos modelar estas afirmações? Imaginaremos que o relacionamento entre gaveta e meia seria "guardar" e de menino com cachorro seria "morder".

Restrições em relacionamentos

Se as multiplicidades dos relacionamentos em ambos os casos forem 5, quantas meias eu estaria referenciando? E quantos cachorros?

Os modelos se diferenciam pois no caso das meias estamos querendo modelar "cardinalidade"que referencia diretamente ao número de meias guardadas e no caso dos cachorros "multiplicidade" o que referencia o número de mordidas e não de cachorros.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Restrições em relacionamentos

No caso das meias, o número de meias é o próprio número de relacionamentos, no caso do cachorro, o número de cachorros pode ser menor ou igual ao número de relacionamentos (mordidas).
As restrições de relacionamento que podemos utilizar para expressar estes tipos de relacionamento são: SET e BAG.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Restrições em relacionamentos

SET: Indica conjunto e portanto não admite mais de um relacionamento com um mesmo objeto.

BAG: Indica irrestrição, isto é, nada se diz a respeito do número de objetos. É possível que todas as mordidas tenham sido provenientes de um mesmo cachorro.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Onde estamos?

- Exercício 1 – Jogo da Velha
- Alguns Erros de Diagramação
- Restrições em Relacionamentos
- **Exercício 2 – Forum de Notícias**
- Exercício 3 – Jogo Sudoku

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Forum de Notícias

Elabore um fórum de notícias de forma que seja possível um usuário cadastrar questões, consultar as questões, responder questões e consultar respostas.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Onde estamos?

- Exercício 1 – Jogo da Velha
- Exercício 2 – Forum de Notícias
- **Exercício 3 – Jogo Sudoku**

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Jogo Sudoku

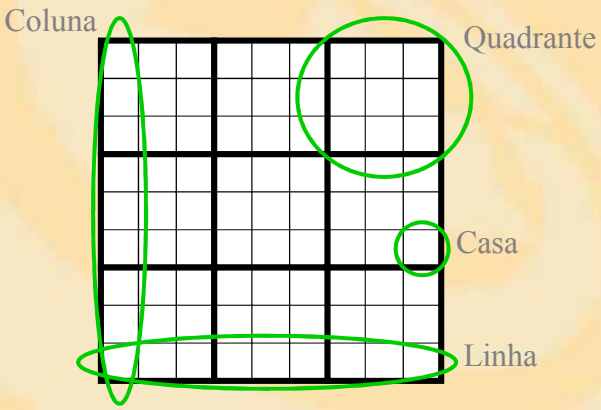
O jogo Sudoku trata-se de uma matriz 9X9 em que alguns números são dispostos como um enunciado.

- O jogador deve posicionar números de 1 a 9 nesta matriz de forma que em nenhuma linha, coluna ou quadrantes da matriz sejam alocados numeros repetidos.
- Um quadrante é uma parte da matriz geral e possui as dimensões 3X3. Existem 9 quadrantes não sobrepostos nesta matriz.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Jogo Sudoku



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Dúvidas?

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Modelagem de Software Orientado a Objetos

Parte 3 – Análise de Modelos de Software

Prof. Dr. Maurício Nacib Pontuschka
tuska@pucsp.br

Objetivos

- Exercitar a elaboração de modelos de software explorando características sintáticas e semânticas nos diagramas da UML.
- Interpretar modelos diagramados com a UML.
- Destacar erros de sintaxe e de semântica em diagramas UML mais especificamente em diagramas de casos de uso e diagramas de seqüência de mensagens.

Fórum de Discussões

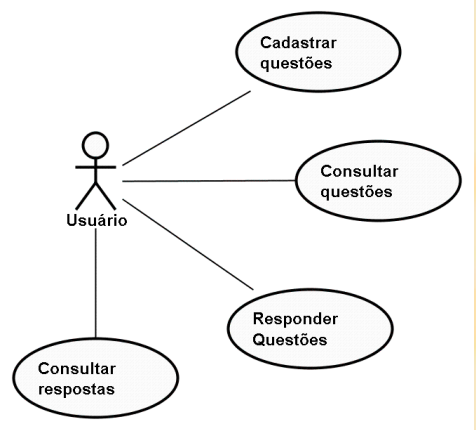
O sistema Fórum de Notícias permite que questões sejam postadas e permite que várias respostas sejam anexadas a cada questão. O diagrama acima identifica os casos de uso previstos para o software.

Qualquer usuário assume o mesmo papel ao utilizar o sistema por tanto todas as funcionalidades estarão disponíveis para qualquer usuário do sistema.

Como se trata de um sistema cujo foco é a modelagem e não efetivamente sua usabilidade, foram descartados aspectos importantes como persistência (armazenamento em disco) das informações entre outras tantas possíveis funcionalidades aderentes ao projeto.

Jogo da Velha

Diagrama de casos de uso

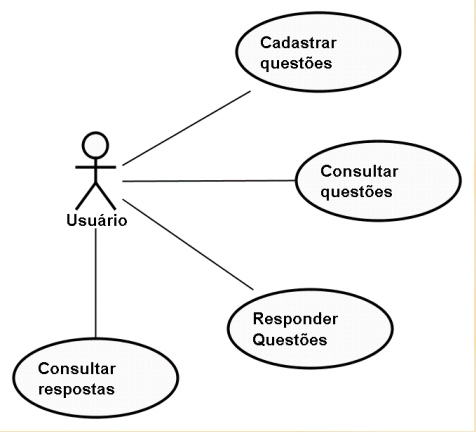


Cadastrar questões
Este caso de uso representa a necessidade de um usuário publicar uma questão de seu interesse para compartilhar com outros usuários e futuramente, poder consultar suas respostas. Cada questão é colocada de forma independente sem qualquer tipo de classificação ou conferência.

PUC • SP COGEAE

Jogo da Velha

Diagrama de casos de uso



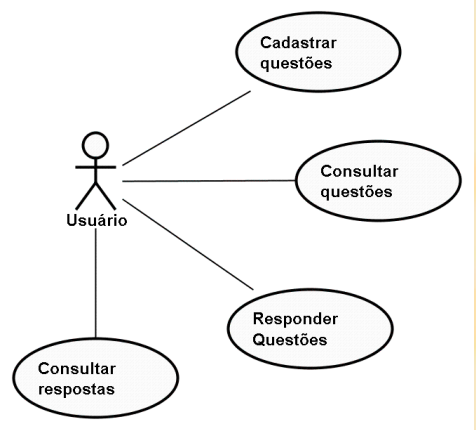
Consultar questões
Um usuário pode percorrer a lista de questões a fim de verificar temas de interesse de outros usuários e até verificar se alguma questão é de seu interesse tanto na leitura das respostas como até, eventualmente, contribuir com sua resposta no futuro.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Jogo da Velha

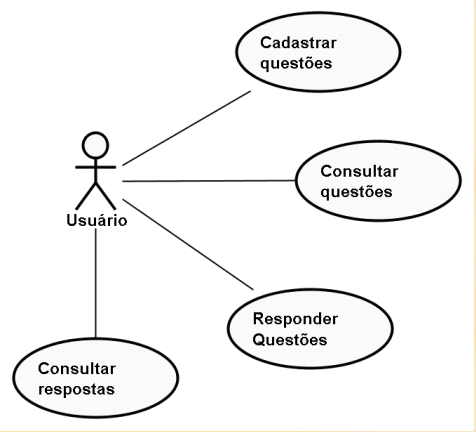
Diagrama de casos de uso



Responder questões
Um usuário pode registrar sua opinião a respeito de temas levantados por outros usuários do Fórum de Notícias. A sua resposta é anexada à questão e é disponibilizada para qualquer outro usuário.

Jogo da Velha

Diagrama de casos de uso



Consultar respostas
O usuário pode se interessar em ler as respostas de questões postadas ou não por ele. Caso tenha interesse, ele poderá navegar pelas questões do Fórum e consultar as várias possíveis respostas anexadas a cada questão.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Fórum de Discussões

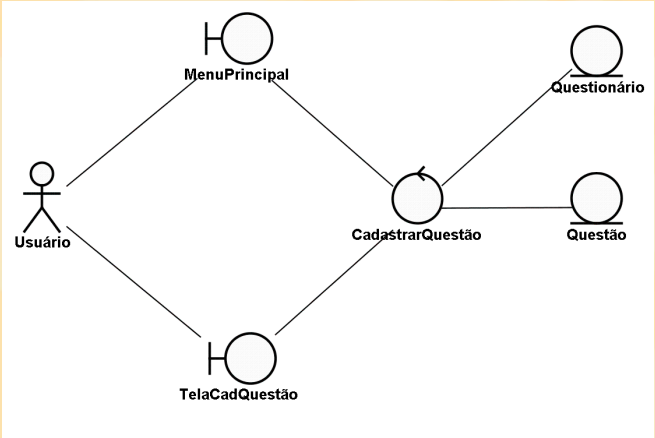
MN#01 – Caso de Uso Cadastrar Questões
Ator Principal: Usuário
Pré-condições Nenhuma
Pós-Condições: Questão Cadastrada

1. Seleciona a opção de cadastrar uma nova questão.	
	2. Exibe uma tela para a redação da nova questão com os campos: nome do usuário e questão. A data e a hora serão armazenadas automaticamente juntamente com os dados de entrada do usuário.
3. Digita as informações da tela e submete a questão para o Fórum de Discussão.	
	4. Acrescenta a questão do usuário no sistema.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Fórum de Discussões

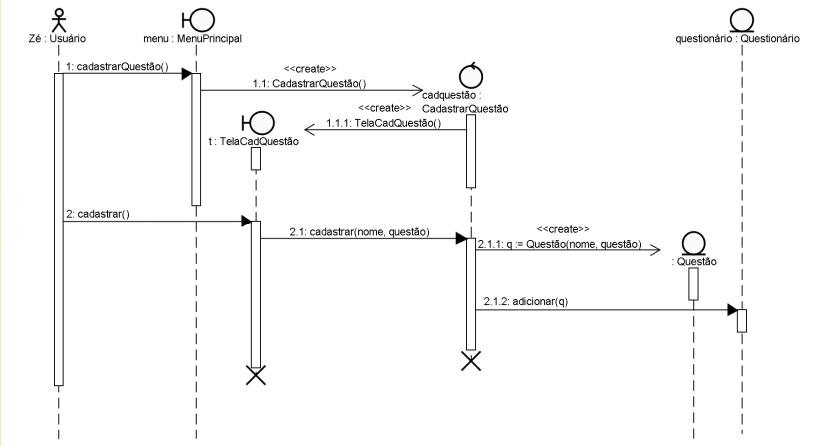
Diagrama de Robustez para o caso de uso Cadastrar Questões



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Fórum de Discussões

Diagrama de Seqüência para o caso de uso Cadastrar Questões



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Fórum de Discussões

MN#02 – Caso de Uso Consultar Questões
Ator Principal: Usuário
Pré-condições Nenhuma
Pós-Condições: Questões consultadas

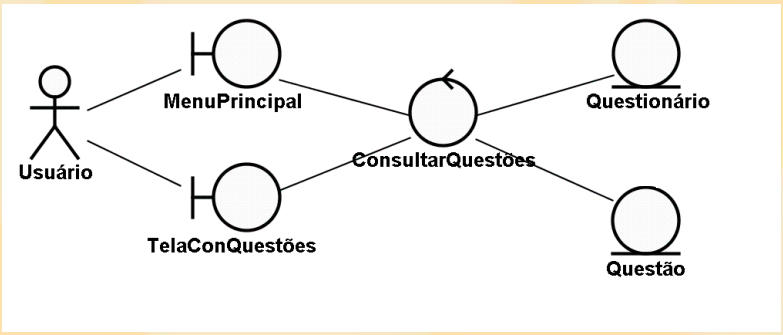
1. Seleciona a opção de consultar questões.	
	2. Exibe uma tela com a lista de todas as questões já cadastradas no sistema.
3. Consulta as questões cadastradas e depois solicita a volta ao menu principal.	
	4. Fecha a tela de consulta.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP COGEAE

Fórum de Discussões

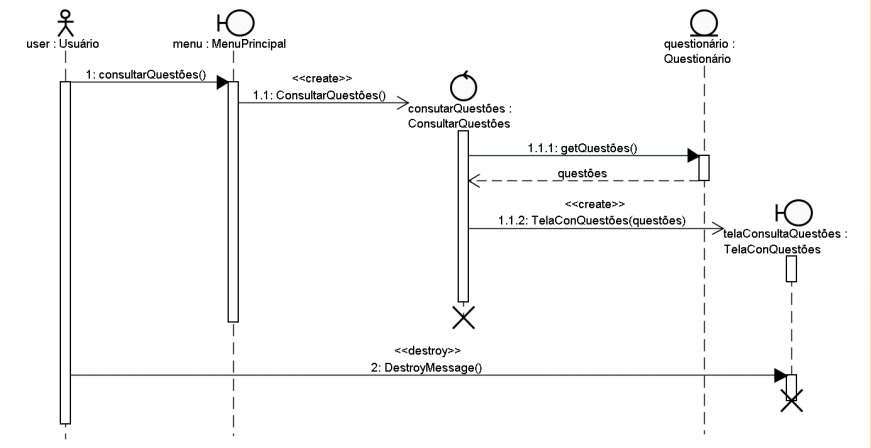
Diagrama de Robustez para o caso de uso Consultar Questões



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Fórum de Discussões

Diagrama de Seqüência para o caso de uso Consultar Questões



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Fórum de Discussões

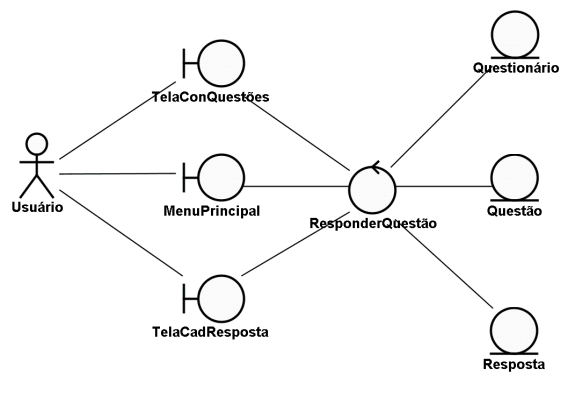
MN#03 – Caso de Uso Responder Questão
Ator Principal: Usuário
Pré-condições: Questão cadastrada
Pós-Condições: Questão com resposta anexada

1. Seleciona a opção de responder uma questão.	
	2. Exibe uma tela com todas as questões para que o usuário escolha a questão a ser respondida.
3. Escolhe uma das questões apresentadas na tela.	
	4. Apresenta uma tela com a pergunta escolhida e um espaço para que o usuário preencha seu nome e resposta para a questão.
5. Preenche seu nome e a resposta à questão.	
	6. Registra o nome, a resposta, a data e a hora do sistema e anexa à questão.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Fórum de Discussões

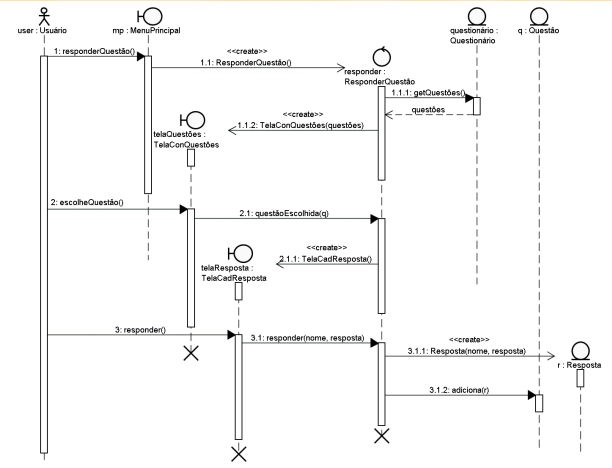
Diagrama de Robustez para o caso de uso Responder Questões



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Fórum de Discussões

Diagrama de Robustez para o caso de uso Responder Questões



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Fórum de Discussões

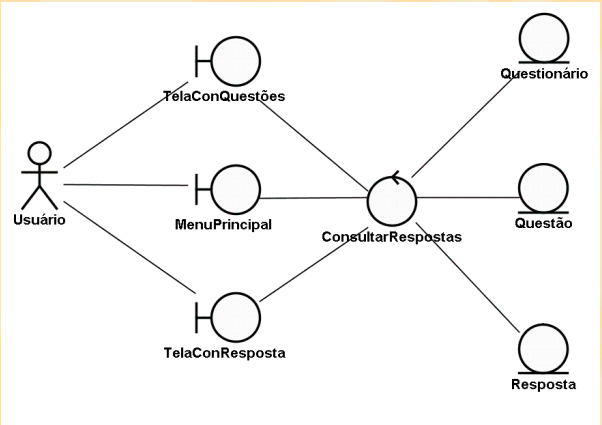
MN#04 – Caso de Uso Consultar Respostas das Questões
Ator Principal: Usuário
Pré-condições: Questão cadastrada
Pós-Condições: Respostas da questão consultadas

1. Seleciona a opção de consultar respostas de uma questão.	
	2. Exibe uma tela com todas as questões para que o usuário escolha a questão para a consulta de suas respostas.
3. Escolhe uma das questões apresentadas na tela.	
	4. Apresenta uma tela com a pergunta escolhida e todas as respostas associadas à esta questão.
5. Consulta as respostas associadas à questão e depois solicita a volta ao menu principal.	
	6. Fecha a tela de consulta de respostas.

Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Fórum de Discussões

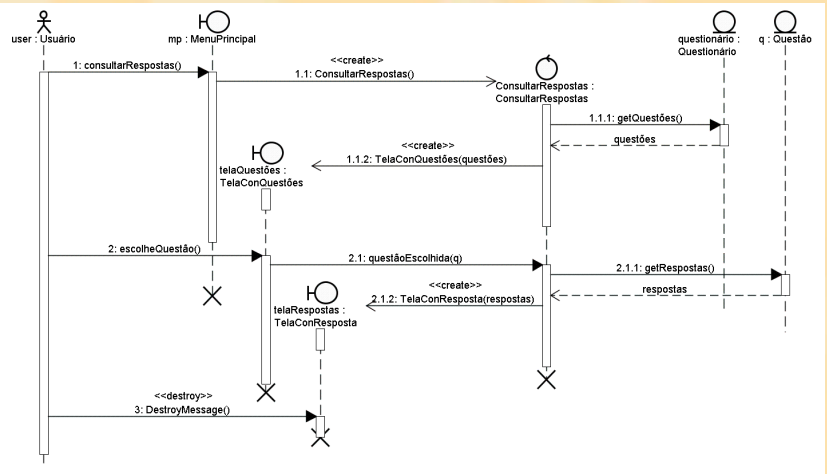
Diagrama de Robustez para o caso de uso Consultar Respostas das Questões



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Fórum de Discussões

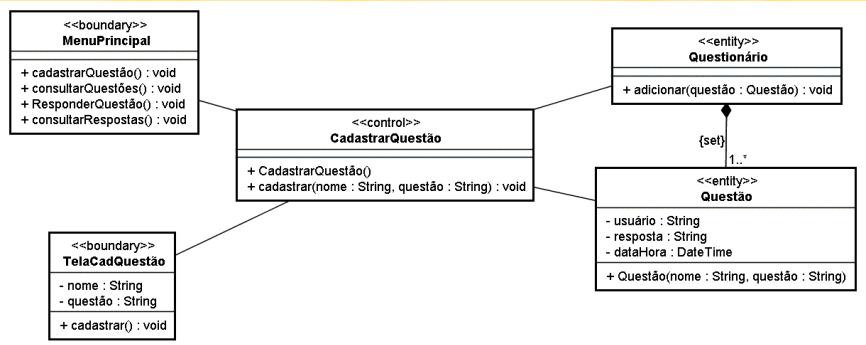
Diagrama de Seqüência para o caso de uso Consultar Respostas das Questões



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

Fórum de Discussões

Refinamento do diagrama de classes
Caso de Uso Cadastrar Questão



Modelagem de Software Orientado a Objetos
Análise de Modelos de Software

PUC • SP
COGEAE

Dúvidas?