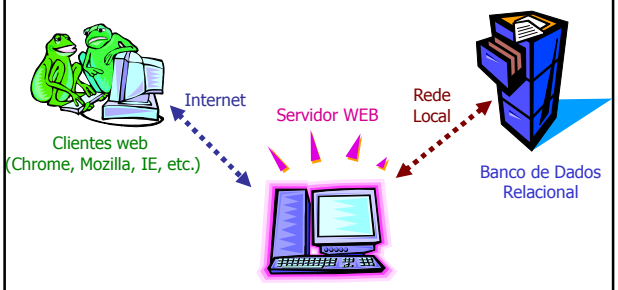
Arquitetura de Software

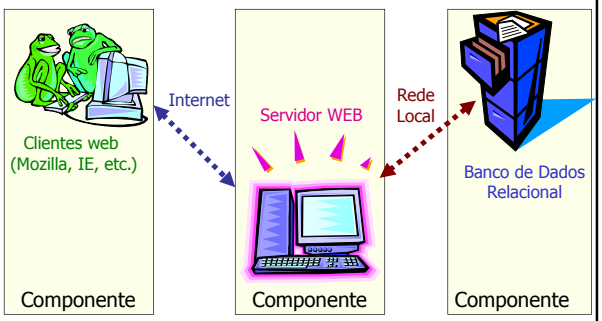
Arquitetura de Software

- Corresponde à **estrutura** de um sistema de software, que engloba
  - componentes de software;
  - suas propriedades visíveis externamente;
  - e os relacionamentos e interações entre os componentes (ou seja, os conectores)
- Esta relacionada às primeiras decisões tomadas no projeto de um sistema
  - **As mais importantes!**

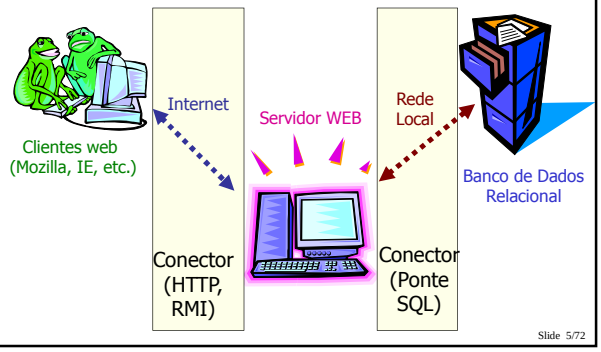
Exemplo: Uma Arquitetura em Camadas



Exemplo: Uma Arquitetura em Camadas



Exemplo: Uma Arquitetura em Camadas



Projeto Arquitetural

- É o processo de projeto que estabelece
  - Os subsistemas (componentes) que constituem um sistema
  - A maneira como esses componentes interagem
- Inclui algumas decisões tecnológicas
  - Ex. Plataforma dos componentes, SGBD
- A saída desse processo de projeto é uma descrição da arquitetura de software.
- A arquitetura de software lida com os requisitos não-funcionais do sistema

Projeto Arquitetural

- É o primeiro estágio do projeto do sistema
- Representa a ligação entre os processos de especificação e de projeto
- É frequentemente conduzido em paralelo com algumas atividades de especificação
  - Às vezes junto com a elicitação de requisitos
- Envolve a identificação dos componentes principais do sistema e sua interação
  - Componentes => unidades de **modularidade**

Vantagens de uma Arquitetura Explícita

- Comunicação com os *stakeholders*
  - A arquitetura pode ser usada como um foco de discussão pelos *stakeholders* do sistema.
- Análise de sistema
  - Se há possibilidade de o sistema atender a seus requisitos de qualidade (não-funcionais)
- Reuso em larga escala
  - A arquitetura pode ser reusável em uma variedade de sistemas
    - Suas partes também!

Decisões de projeto

- Projeto de arquitetura é um processo criativo
  - Cada sistema envolve diferentes decisões/requisitos/conflitos/restrições
  - Envolve solucionar os problemas representados pelos requisitos
- Decisões de projeto:
  - Escolhas feitas durante o projeto de um sistema
  - Afetam a capacidade do sistema de fornecer seu serviço
  - Normalmente resultam em compromissos
  - É importante avaliar as opções existentes
  - Não estão restritas ao projeto arquitetural!

Características de um Sistema que decorrem de sua Arquitetura

- Desempenho
  - Localizar operações críticas e minimizar comunicações.
- Proteção (*security)*
  - Usar uma arquitetura em camadas com itens críticos nas camadas mais internas.
- Segurança (*safety*)
  - Localizar características críticas de segurança em um pequeno número de subsistemas.
- Disponibilidade
  - Incluir componentes redundantes e mecanismos para tolerância à falhas.
- Facilidade de manutenção
  - Usar componentes facilmente trocáveis.

Conflitos de arquitetura (exemplos)

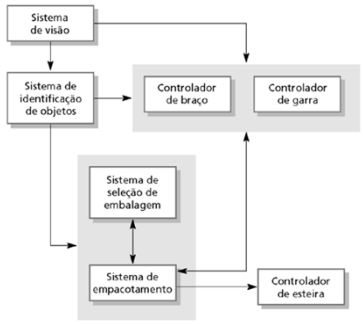
- A introdução de dados redundantes aprimora a disponibilidade, mas torna a proteção mais difícil
  - E cria dificuldades para tornar o sistema confiável em outras partes
- Localizar as funcionalidades críticas de segurança em poucos locais pode criar gargalos de desempenho

Representação de Arquiteturas

- Arquiteturas são um ativo importante no desenvolvimento
  - Podem ser a diferença entre o sucesso e o fracasso
- Representá-las é importante
  - Torna possível “falar” sobre ela
  - O projeto de arquitetura é normalmente expresso como um diagrama de blocos
- Modelos mais específicos também podem ser desenvolvidos.

Exemplo de Arquitetura: Sistema de controle robotizado de empacotamento

Figura 11.1
Diagrama de blocos de um sistema de controle robotizado de empacotamento.



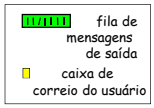
Visões Arquiteturais

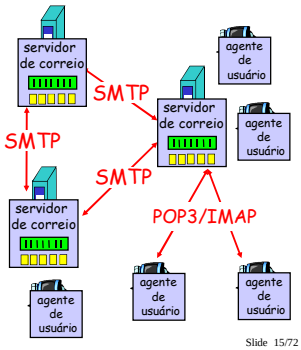
- A arquitetura de um sistema de software normalmente é representada através de várias **visões**
- Visões são maneiras diversas de se enxergar uma mesma arquitetura
  - Enfocando diferentes aspectos de interesse
  - Ex.: as várias plantas de uma casa
- Arquiteturas de software são especificadas através de uma ou mais visões. Exemplos:
  - Lógica (ex: classes, pacotes)
  - De casos de uso
  - De interação (diagramas de sequência)
  - Operacional, Física ou de Alocação (diagramas de componentes)

Correio Eletrônico – Visão 1

Três principais elementos:

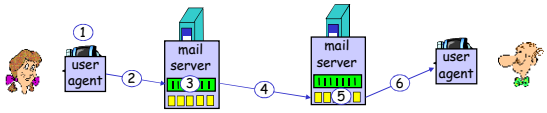
- agentes de usuário (UA).
- servidores de correio.
- simple mail transfer protocol: SMTP.





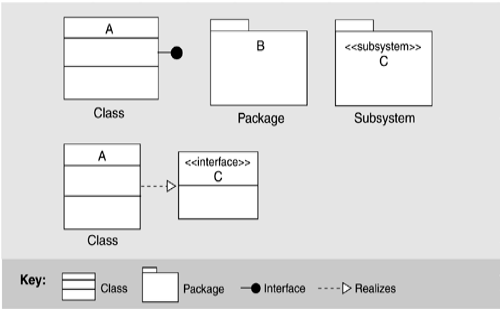
Correio Eletrônico – Visão 2

1) Alice usa o UA para compor uma mensagem "para" `bob@someschool.edu`
2) O UA de Alice envia a mensagem para o seu servidor de correio; a mensagem é colocada na fila de mensagens.
3) O lado cliente do SMTP abre uma conexão TCP com o servidor de correio de Bob.
4) O cliente SMTP envia a mensagem de Alice através da conexão TCP.
5) O servidor de correio de Bob coloca a mensagem na caixa de entrada de Bob.
6) Bob chama o seu UA para ler a mensagem.

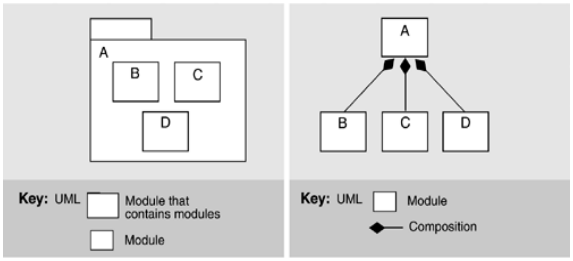


Documentando Visões em UML

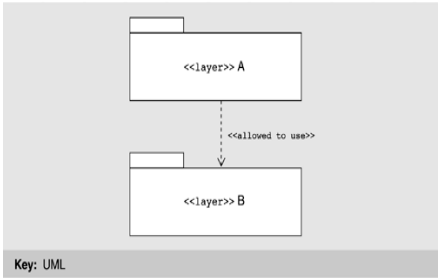
UML: Visão de Módulos



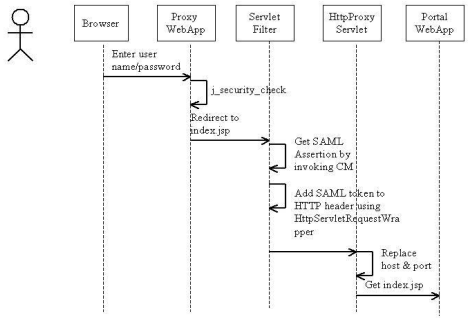
UML: Visão de Módulos



UML: Visão de Módulos



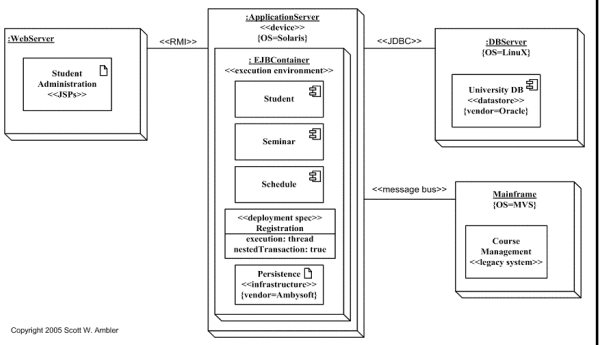
UML: Visão de Interação



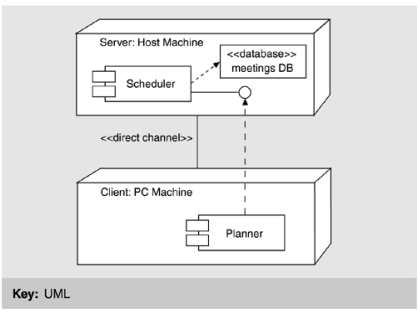
UML: Visão de Alocação

- Diagramas de Implantação mostram a alocação dos componentes de software do sistema aos elementos de hardware
- Incluem os protocolos de interação entre as partes do sistema
- Podem também indicar informações adicionais, como:
  - Ambientes de execução (como máquinas virtuais e servidores de aplicação)
  - Sistemas operacionais
  - Tecnologias específicas

Diagramas de Implantação: Exemplo 1



Diagramas de Implantação: Exemplo 2

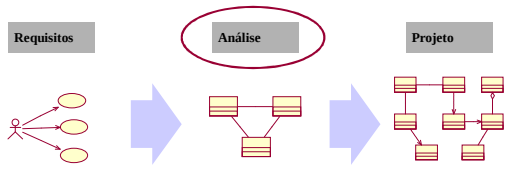


Criação do Modelo de Análise no OpenUP – Análise de Casos de Uso

Contexto

- Com base nos casos de uso, queremos agora gerar um primeiro modelo de arquitetura do sistema.
- Este modelo é chamado de modelo de análise.

Contexto



A Atividade de Análise

- Vai proporcionar um método que permita que criemos um modelo de classes do sistema a partir dos casos de uso
- Trará a resposta para a pergunta:
  - Quais classes preciso para implementar estes casos de uso?

Análise no OpenUP

- A maneira como vamos realizar a etapa de análise se baseia no processo usado no OpenUP
- A análise será orientada a casos de uso, ou seja, os casos de uso servirão de guia para a etapa de análise

Casos de Uso X Análise

casos de uso	análise
Descritos na linguagem do cliente	Descrito na linguagem dos desenvolvedores
Visão externa do sistema	Visão interna do sistema
Captura as funcionalidades do sistema	Mostra, de modo abstrato, como a funcionalidade pode ser realizada
Estruturado por casos de uso	Estruturado por classes e pacotes

Passos da Atividade de Análise

- Identificar as classes
  - Identificar persistência
- Identificar responsabilidades das classes
- Identificar relacionamentos
- Identificar atributos

Identificando as classes

- No primeiro passo de análise, identificaremos três tipos de classes:
  - Fronteira
  - Entidade
  - Controle
- Tais classes são identificadas separadamente para cada de uso

Classes de Fronteira

- Utilizada para modelar a interação entre um ator e o sistema
- Para cada interação entre um ator e caso de uso, é criada uma classe de fronteira
- Possuem o estereótipo <<*boundary*>>

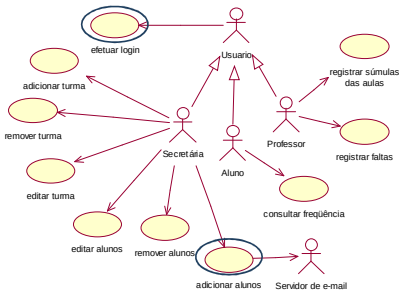
Classes de Entidade

- Utilizadas para modelar a informação manipulada pelo sistema
- Podem ser persistentes ou não
- Conceito análogo às entidades dos diagramas ER
- São identificadas a partir do fluxo de eventos do caso de uso
- Possuem o estereótipo <<*entity*>>

Classes de Controle

- Classes responsáveis por controlar o fluxo de execução do caso de uso
- Normalmente é criada uma classe de controle para cada caso de uso
- Possuem o estereótipo <<*control*>>

Exemplo



Exemplo

Efetuar Login

Fluxo de eventos:

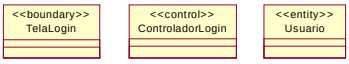
1. Usuário informa login e senha
2. Sistema checa se o login e senha conferem
3. Sistema registra a sessão do aluno e a tela principal do sistema é exibida

Exemplo

- Que classes preciso criar?
  - uma classe de fronteira para lidar com a interação dos atores com o sistema
  - uma classe de entidade para representar as informações relevantes do aluno
  - uma classe de controle para gerenciar o fluxo de execução do caso de uso

Exemplo

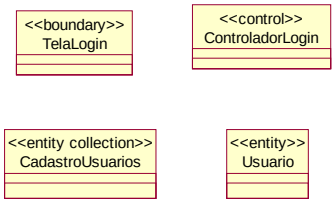
Há diferentes opções de visualização dos estereótipos. A opção padrão é mostrada acima - os estereótipos são visualizados através da mudança dos ícones das classes. Há também a opção de se visualizar os estereótipos do modo convencional (<<estereótipo>>).



Persistência

- Mas caso alguma classe de entidade precise ser persistente?
- Que classe será responsável por realizar as tarefas de persistência?
- Para cada classe de entidade que precise ser persistente, é criada uma nova classe com o estereótipo <<entity collection>>

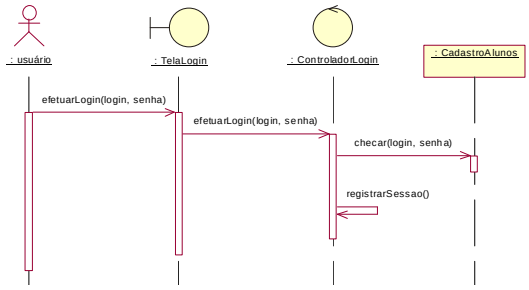
Exemplo



Diagramas de interação

- Após a identificação das classes, é necessário descobrir quais são as responsabilidades de cada classe, o que cada uma precisa fazer.
- Os diagramas de interação (sequência e colaboração) são muito úteis nesta tarefa

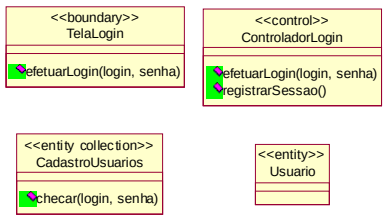
Exemplo



Alocando responsabilidades

- Após identificarmos as responsabilidades (métodos) pelos diagramas de interação, devemos acrescentar os métodos nas classes previamente identificadas (1º passo)

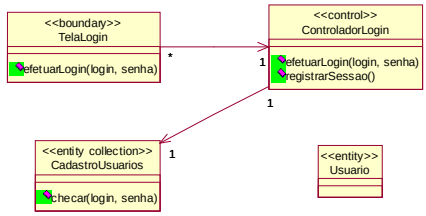
Classes com métodos



Identificando relacionamentos

- As classes identificadas não funcionam isoladamente, elas se relacionam com as demais classes
- Os diagramas de interação são muito úteis nesta tarefa
- Para cada ligação presente nos diagramas de interação, é necessário um relacionamento no diagrama de classes

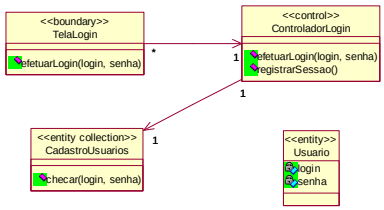
Classes com relacionamentos



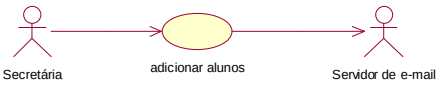
Identificando Atributos

- Também é necessário identificar quais os atributos das classes
- Um bom conhecimento do domínio do problema é bastante importante para esta tarefa, principalmente na identificação de atributos das classes de entidade
- Nesta etapa ainda não precisamos indicar quais os tipos dos atributos

Diagrama final

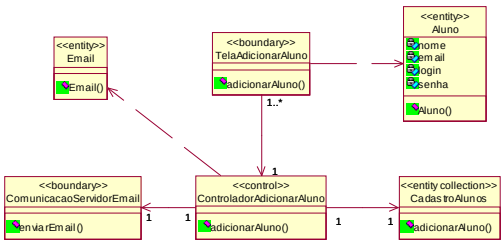


Exemplo – adicionar aluno



Fluxo de eventos:
1. Secretária informa dados do aluno
2. Secretária seleciona a opção "confirmar cadastro"
3. Sistema checa se os dados são válidos
4. Sistema adiciona o aluno à base de dados
5. Sistema envia um e-mail para o aluno, informando-o seu login e senha
6. Sistema exibe uma mensagem de confirmação de cadastro

Exemplo – Análise adicionar aluno

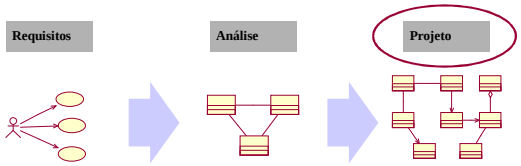


Modelo de Projeto – um Refinamento do Modelo de Análise

Contexto

- Após a etapa de análise temos um primeiro modelo do sistema
- Queremos agora melhorar esse modelo, a ponto de gerarmos facilmente a implementação do sistema
- Este modelo é chamado de Modelo de Projeto

Contexto



Análise X Projeto

- Abstrato X Concreto
- Independente X dependente da tecnologia de implementação
- Simples X detalhado
- Modelos por caso de uso X unificação em um único modelo

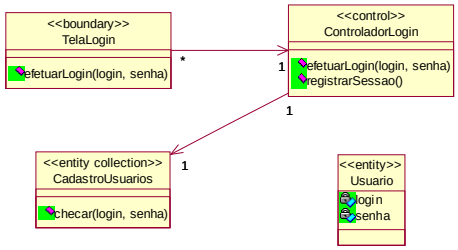
Atividades – Projeto

- Refinar o modelo de classes
- Projetar arquitetura
  - Camadas
  - Separação em pacotes
- Projetar Banco de Dados

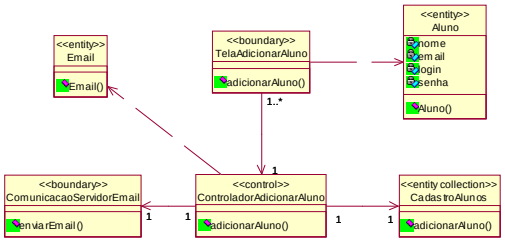
Refinar o modelo de classes

- Juntar todas as classes em um só diagrama
- Analisar se é necessário criar novas classes ou remover classes existentes
- Eliminar os estereótipos de análise
- Adicionar modificadores de visibilidade aos métodos e atributos
- Definir os tipos dos atributos

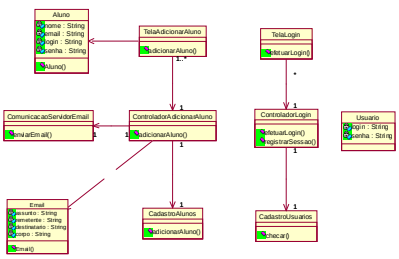
Exemplo – Análise login



Exemplo – Análise adicionar aluno



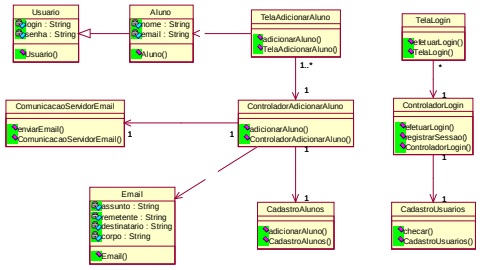
Exemplo – diagrama único



Refinar o modelo de classes

- Detalhar assinatura dos métodos
  - definir todos os parâmetros dos métodos, seu tipos e o tipo de retorno dos métodos
- Mapear associações em atributos
- Analisar a possibilidade de utilizar herança

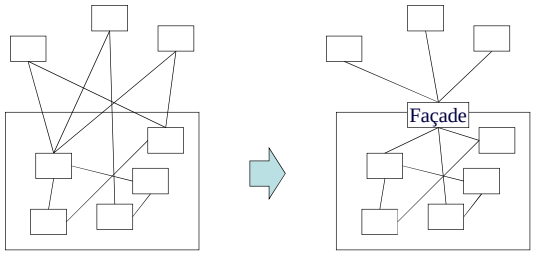
Exemplo – diagrama melhorado



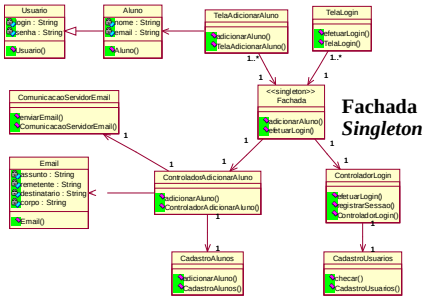
Refinar o modelo de classes

- Identificar padrões de projeto a serem aplicados
  - Abstrações de uma solução de projeto recorrente
  - Envolvem dependências, estruturas, interações e convenções relativos a classes e objetos
    - Ex: Singleton, Fachada
- Revisar as classes

Exemplo de Padrão – Fachada (*Façade*)



Modelo Melhorado com Padrões de Projeto



Projetar arquitetura

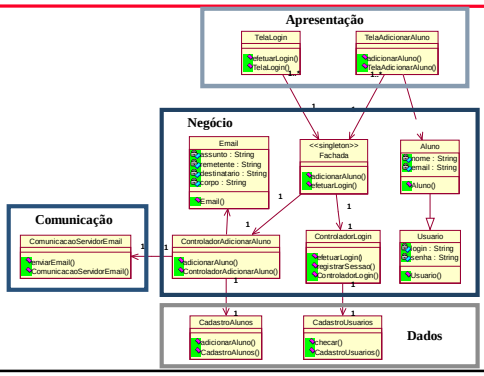
- Dividir o sistema em camadas.

Apresentação	Interface com o usuário
Comunicação	Comunicação entre apresentação e negócio e com outros sistemas
Negócio	Regras de negócio inerentes à aplicação
Dados	Código relacionado ao mecanismo de persistência utilizado

Projetar arquitetura

- Por que dividir em camadas?
  - Aumentar modularidade
  - Diminuir dependências
  - Facilitar possível troca de camadas

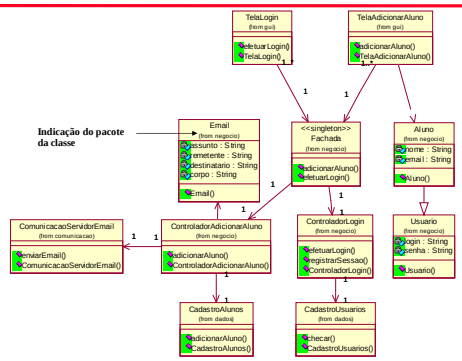
Camadas



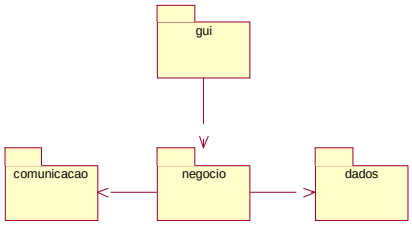
Divisão do sistema em pacotes

- Agrupar classes em pacotes
- Possíveis critérios:
  - Camadas
  - Lógica do sistema
- Critérios escolhidos devem minimizar a dependência entre os pacotes
- Criar um diagrama de pacotes indicando as dependências entre os pacotes

Pacotes



Pacotes



Referências

- The Unified Software Development Process - Jacobson, Rumbaugh, Booch
- The UML Reference Manual - Rumbaugh, Jacobson, Booch