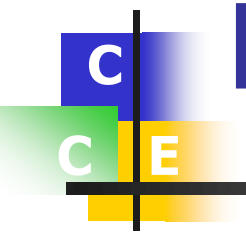
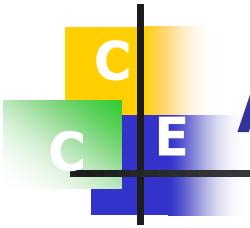


Tecnologias Atuais de Desenvolvimento de *Software*

Análise OO e UML

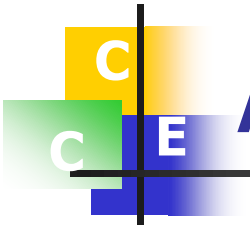
Prof. Luiz Antônio

lpereira@uninet.com.br



Agenda

- OO
  - Análise de sistemas, modelos e modelagem
  - Análise estruturada
  - Análise essencial
  - Motivação para OO
  - Objetos, o conceito central
- UML
  - (Breve) história
  - Características gerais
  - Propósitos
  - Aplicações
  - Principais diagramas da UML



Análise de sistemas

Comunicação entre analistas e usuários para definir o propósito e os requisitos de um sistema.

Modelo e modelagem de sistemas

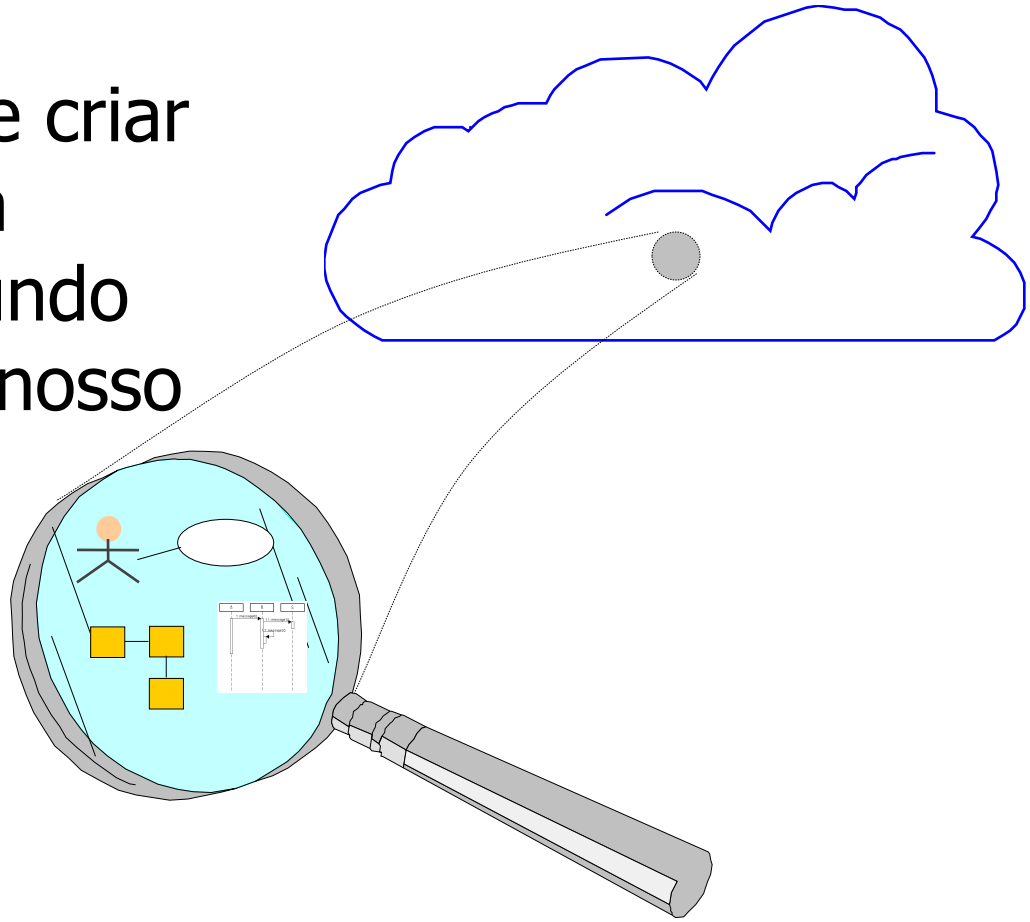
- **Modelo** é uma representação de um sistema (ou de um objeto qualquer). É uma abstração da realidade e representa uma seleção de características do mundo real que são relevantes para o propósito com o qual o modelo foi construído.

Modelo e modelagem de sistemas

- Detalhe:
  - Todo bom modelo precisa representar
    - A estrutura dos dados (a dimensão de dados);
    - As funções que transformam os dados (a dimensão funcional);
    - As sequências de aplicação das funções (a dimensão temporal) e as demais restrições.

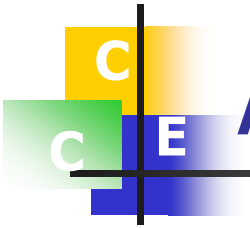
Modelo e modelagem de sistemas

- **Modelagem** consiste em se criar um modelo da parcela do mundo real que é de nosso interesse.



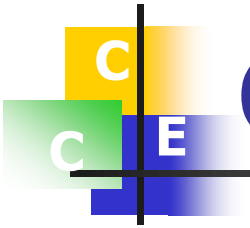
Modelo e modelagem de sistemas

- Por que modelar?
  - Possibilitar o estudo do comportamento do sistema;
  - Possibilitar a discussão de correções, modificações e validação com o usuário, a um custo baixo;
  - Facilitar a comunicação entre os membros da equipe;
  - Documentar o sistema, registrando todas as decisões tomadas durante o projeto.



Análise Estruturada e Essencial

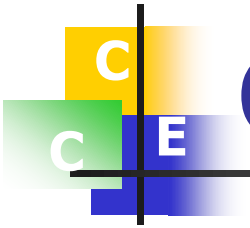
- Metodologias e técnicas muito usados até meados da década de 90:
  - Análise estruturada;
  - Análise essencial.
- Dificuldades:
  - Separação entre dados e processos;
  - Descontinuidade da análise para o projeto;
  - Diferenças de modelagem entre tipos de sistemas.



OOA&D

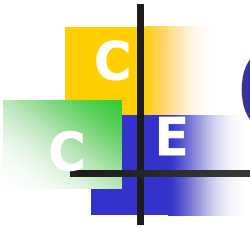
- Décadas de 70-80 surgiram linguagens OO e híbridas:
  - Smalltalk;
  - C++;
  - Object-Pascal.
- POO usado no contexto acadêmico somente;
- Métodos de projetos OO passaram a ser pesquisados para dar suporte “organizado” ao desenvolvimento de sistemas com linguagens OO.

Continua...



OOA&D

- Resolveram as dificuldades dos paradigmas anteriores:
  - Ok quanto à separação entre dados e processos
    - Diagramas de classes identificam as entidades, os relacionamentos, as operações e responsabilidades.
  - Ok quanto à descontinuidade da análise para o projeto
    - Os diagramas são refinados ao longo do ciclo de vida pelo acréscimo de detalhes em um único nível hierárquico.
  - Ok quanto às diferenças de modelagem entre tipos de sistemas
    - As metodologias e linguagem de especificação adotadas permitem o tratamento de casos em qualquer domínio, indistintamente.



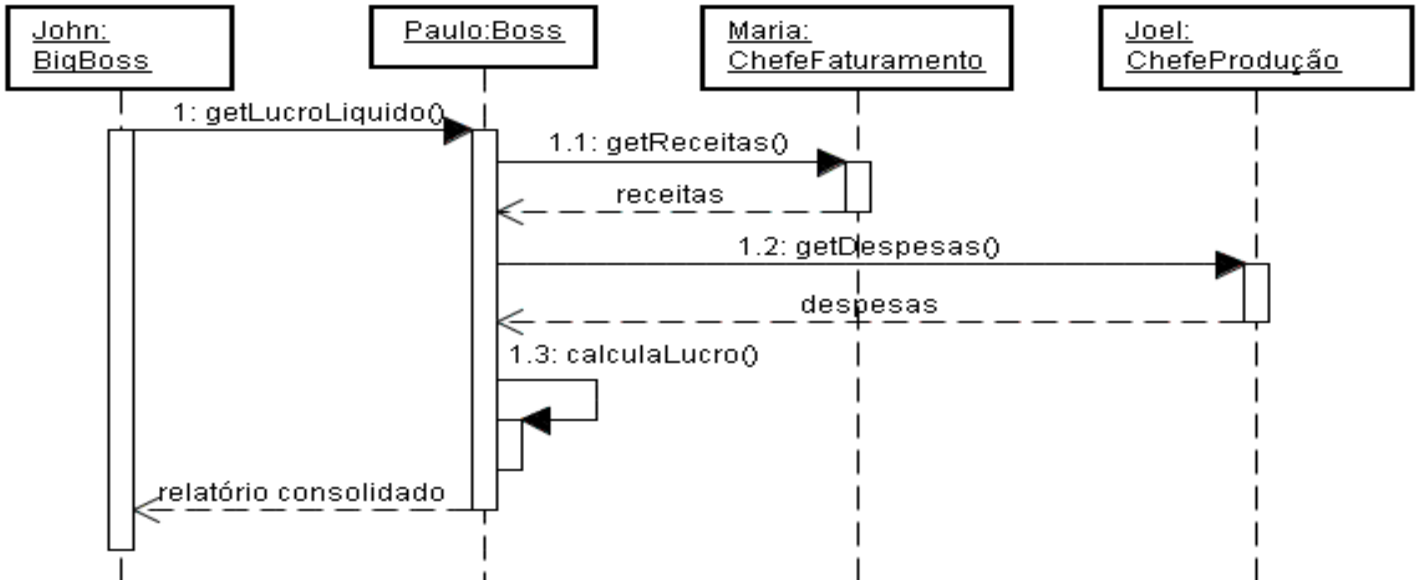
Objetos, o conceito central

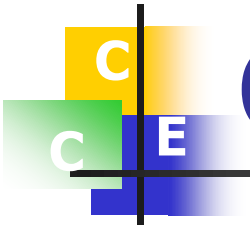
Objetos são entidades:

- Que representam coisas concretas ou abstratas do mundo real (um carro, um processo químico);
- Que se categorizam em *classes*;
- Que possuem *estados*;
- Que mantêm relacionamentos entre si;
- Têm responsabilidades e executam operações...

Objetos, o conceito central

- ... participando *colaborativamente*, em seqüências pré-definidas (programadas), da execução das funções do sistema;

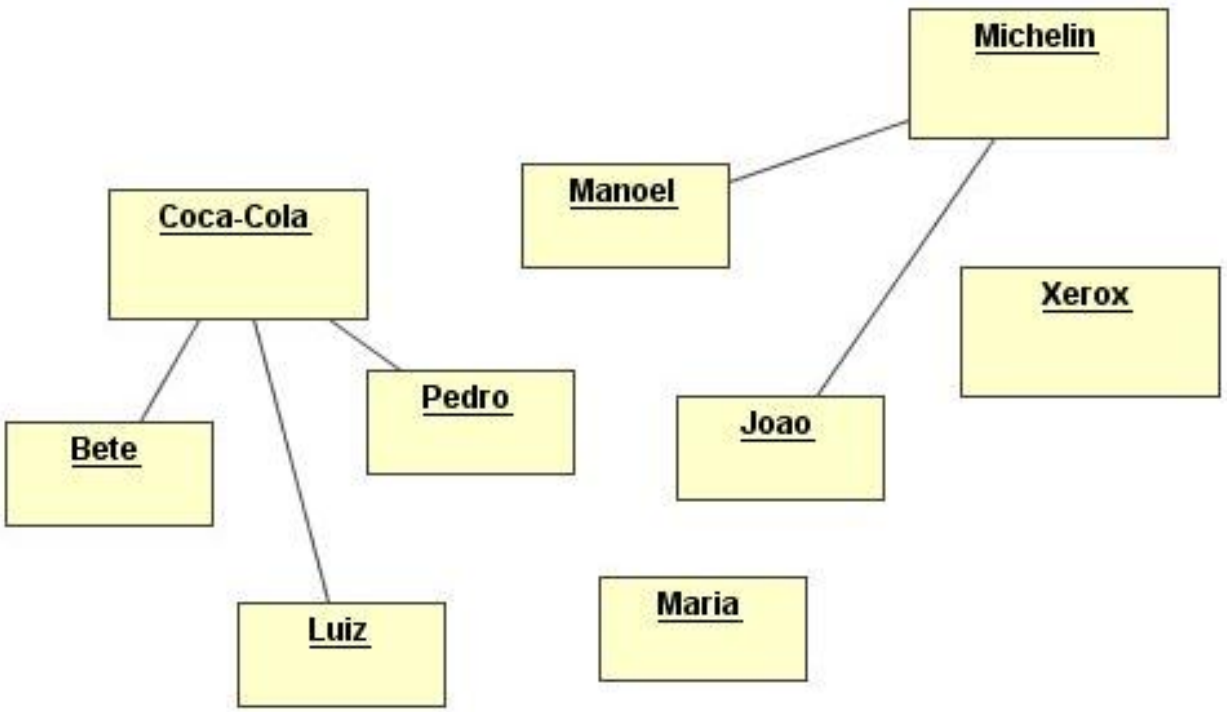


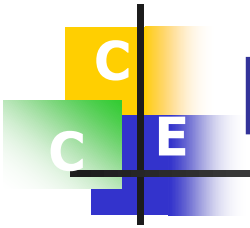


Objetos, o conceito central

- Objetos são entidades que os projetistas definem as características e como vão colaborar para a realização dos objetivos de um sistema.

Objetos, o conceito central





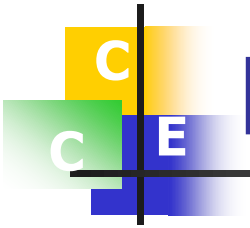
História da UML

- Meados da década de 90:
  - Massa crítica de idéias produzidas pelas várias metodologias;
  - Necessidade de estabilizar o mercado OO para viabilizar o desenvolvimento de ferramentas CASE OO.

História da UML

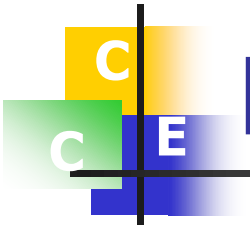
Guerra dos métodos

Iniciativa da comunidade no sentido de juntar forças para criar uma linguagem unificada



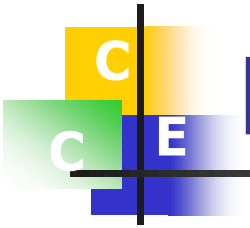
História da UML

- Dentre os métodos mais importantes destacavam-se:
  - Booch (Grady Booch - Rational Software), bom nas fases de projeto e construção;
  - OOSE (Ivar Jacobson - Objectory), bom na captura de requisitos e análise (abordagem em alto nível de abstração).
  - OMT (Jim Rumbaugh - GE), bom na análise de SIs com uso intensivo de dados.
- Cada um dos três passa a usar também idéias dos outros dois.



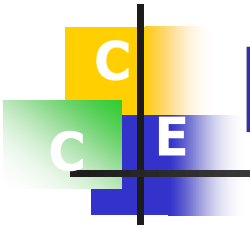
História da UML

- Os três – RJB - agora (96/97) juntos na Rational:
  - Iniciaram o processo de padronização da UML, criaram uma proposta inicial e ...
  - ... "passaram a bola" para o OMG, que passou a considerar outras opiniões;
  - Desenvolveram a metodologia unificada e software de apoio à mesma (Objectory) e software de case (Rose)



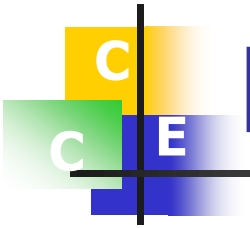
História da UML

- UML está, atualmente, na versão 2.2, e o desenvolvimento é gerido pelo OMG;
- Especificação disponível em pdf;
- Versões anteriores (i.e. 1.3, 1.4, 1.5 e 2.0) ainda são bastante empregadas, incluindo CASEs.



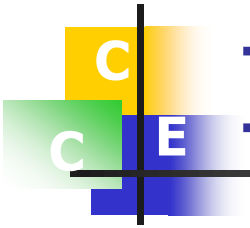
Principais Propósitos

- Permitir a modelagem de sistemas, do conceito ao artefato executável, utilizando técnicas OO;
- Contemplar as necessidades de modelagem de sistemas pequenos e simples a grandes e complexos;
- Prover uma linguagem que permita o entendimento e utilização por humanos e por máquinas;
- Ser independente da linguagem de programação e do processo de desenvolvimento;
- Construir modelos precisos, sem ambigüidades e completos;
- Linguagem para visualização do modelo, facilitando o entendimento pela equipe de desenvolvimento e pelos clientes;
- Servir para construir código, embora não seja uma linguagem de programação;
- Servir para documentar sistemas (requisitos, arquitetura, projeto, etc.).



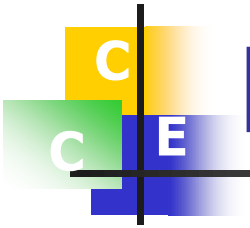
Referências

- Referências importantes:
  - Fowler & Scott, “UML Essencial”, Bookman
  - Booch, Rumbaugh & Jabobson, “UML: Guia do Usuário”, Campus
  - Larman, “Utilizando UML e Padrões-Uma Introdução à Análise e Projeto Orientados a Objetos, Bookman.
  - ...
- ...Referência Básica (centenas de páginas):
  - Infra-estrutura;
  - Superestrutura;
  - OCL.



Importante

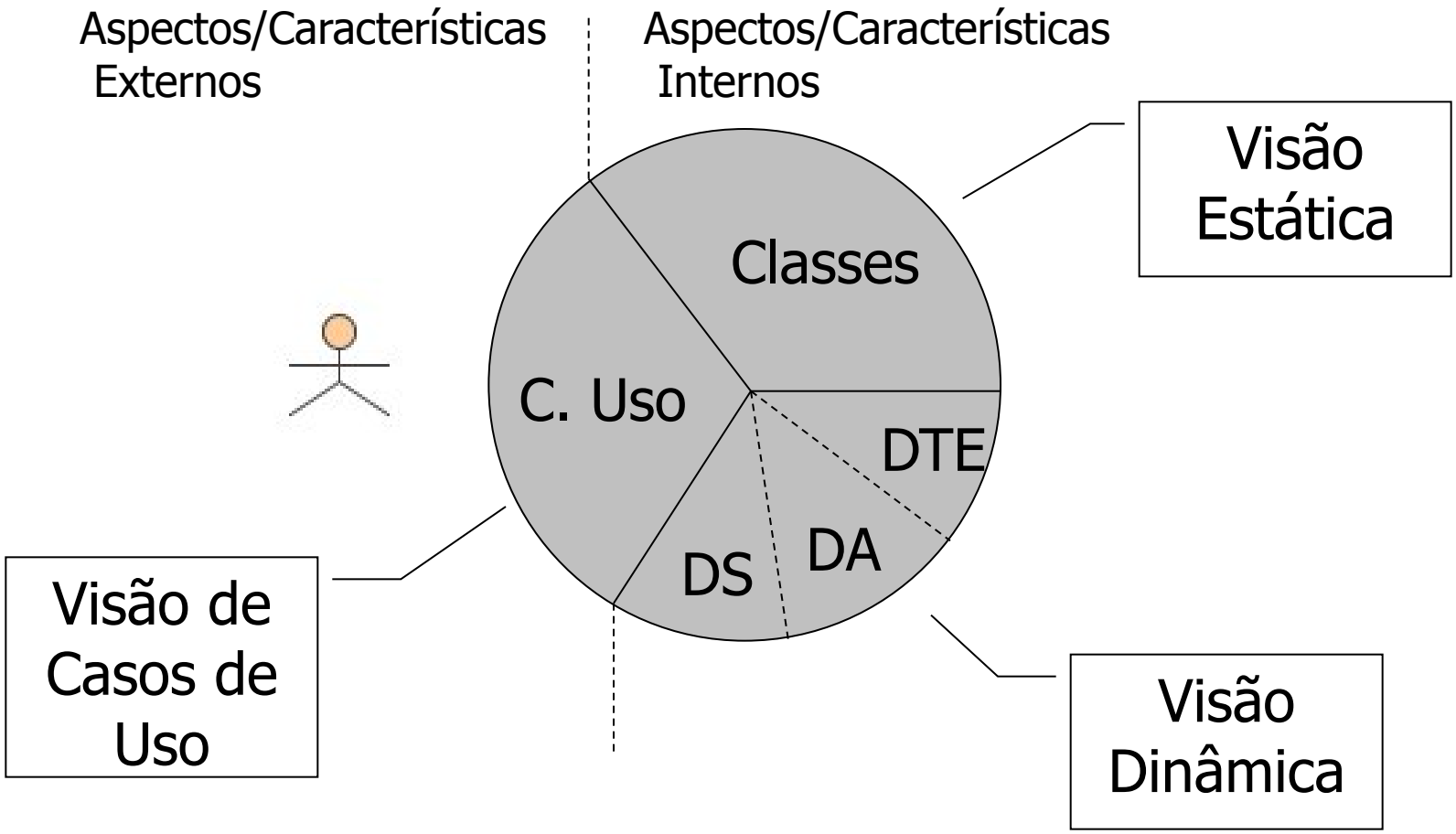
- Aproximadamente 40% da linguagem cobre 98% das necessidades de um projeto comum (Fowler).



Diagramas da UML

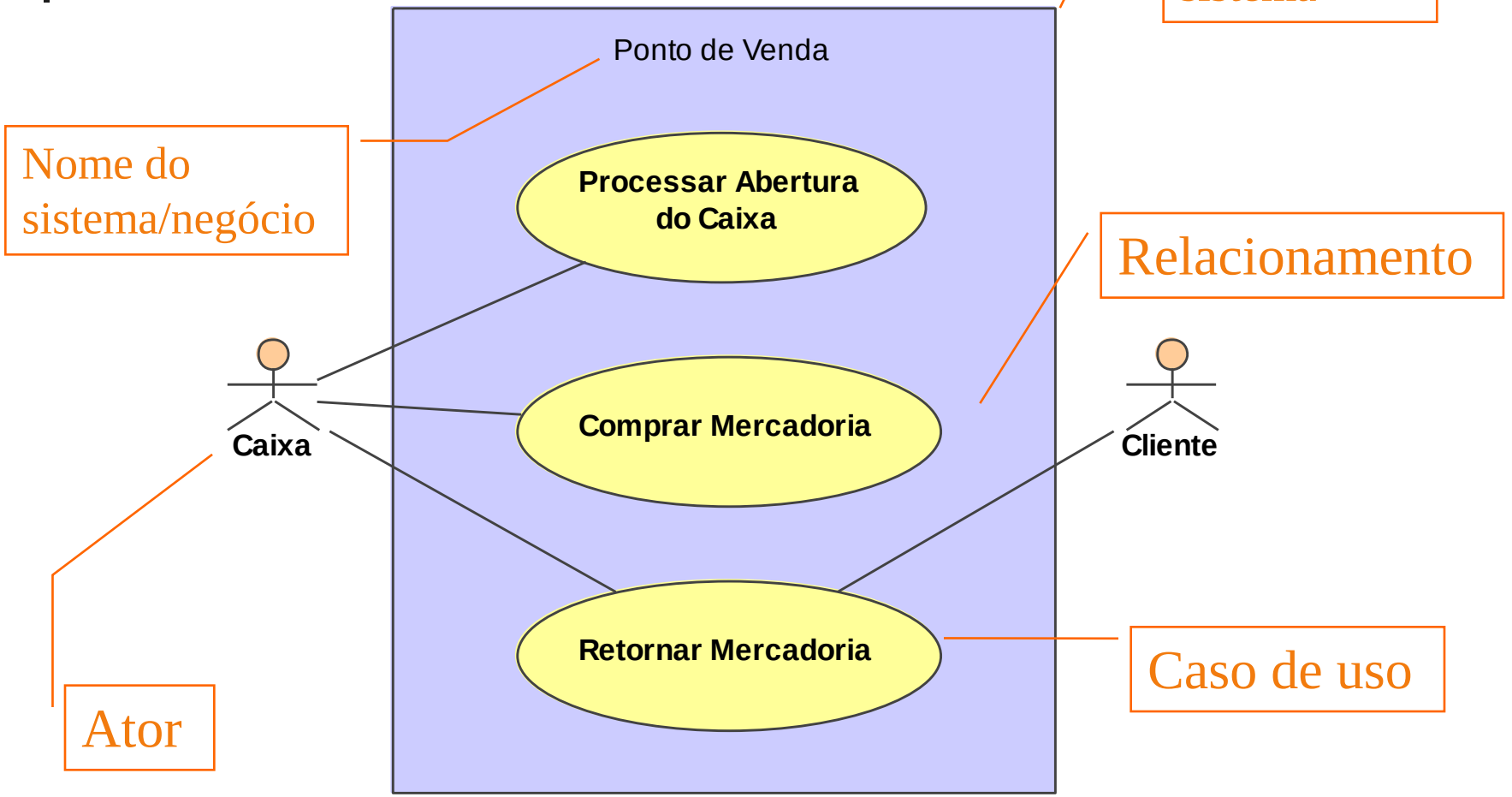
- A UML define 13 diagramas, divididos em três categorias:
  - Diagramas estruturais
    - Classes, objetos, componentes, pacotes, estruturas compostas e implantação.
  - Diagramas comportamentais
    - Casos de uso (usados por algumas metodologias durante a captura de requisitos), atividade e máquina de estados.
  - Diagramas de interação
    - Seqüência, comunicação, *temporização* (de tempo), visão geral de interação.

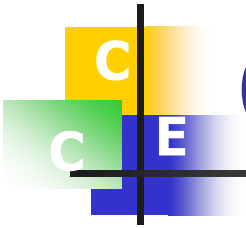
Modelo UML Básico



Casos de Uso

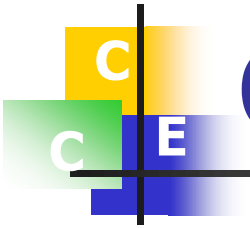
-


Limites do sistema
Ponto de Venda
Nome do sistema/negócio
Processar Abertura do Caixa
Relacionamento
Comprar Mercadoria
Caixa
Cliente
Retornar Mercadoria
Caso de uso
Ator



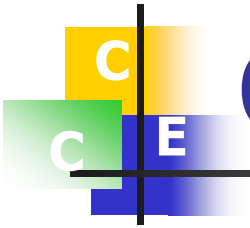
Casos de Uso

- Atores são pessoas, sistemas ou *hardware* que interagem com o negócio ou sistema em estudo;
- Atores podem participar de um ou mais casos de uso;
- Um único usuário pode interpretar o papel de vários atores;
- Vários usuários podem interpretar o papel de um único ator;
- A implementação interna dos atores não é relevante.



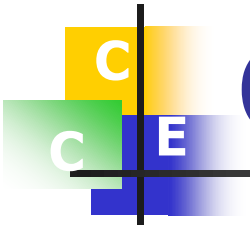
Casos de Uso

- Casos de uso (ovais/elipses) representam funcionalidades de um sistema ou transações de um negócio;
- Relacionamentos
  - Entre atores e casos de uso especificam quais atores participam de quais casos de uso;
  - Entre casos de uso, representando dependências.



Casos de Uso

- Fronteira (ou limite) do sistema/negócio:
  - Opcional, segundo a UML;
  - Colocamos a fronteira quando queremos e podemos (às vezes não conseguimos definir uma fronteira retangular com os U.C. dentro e os atores fora).



Casos de Uso

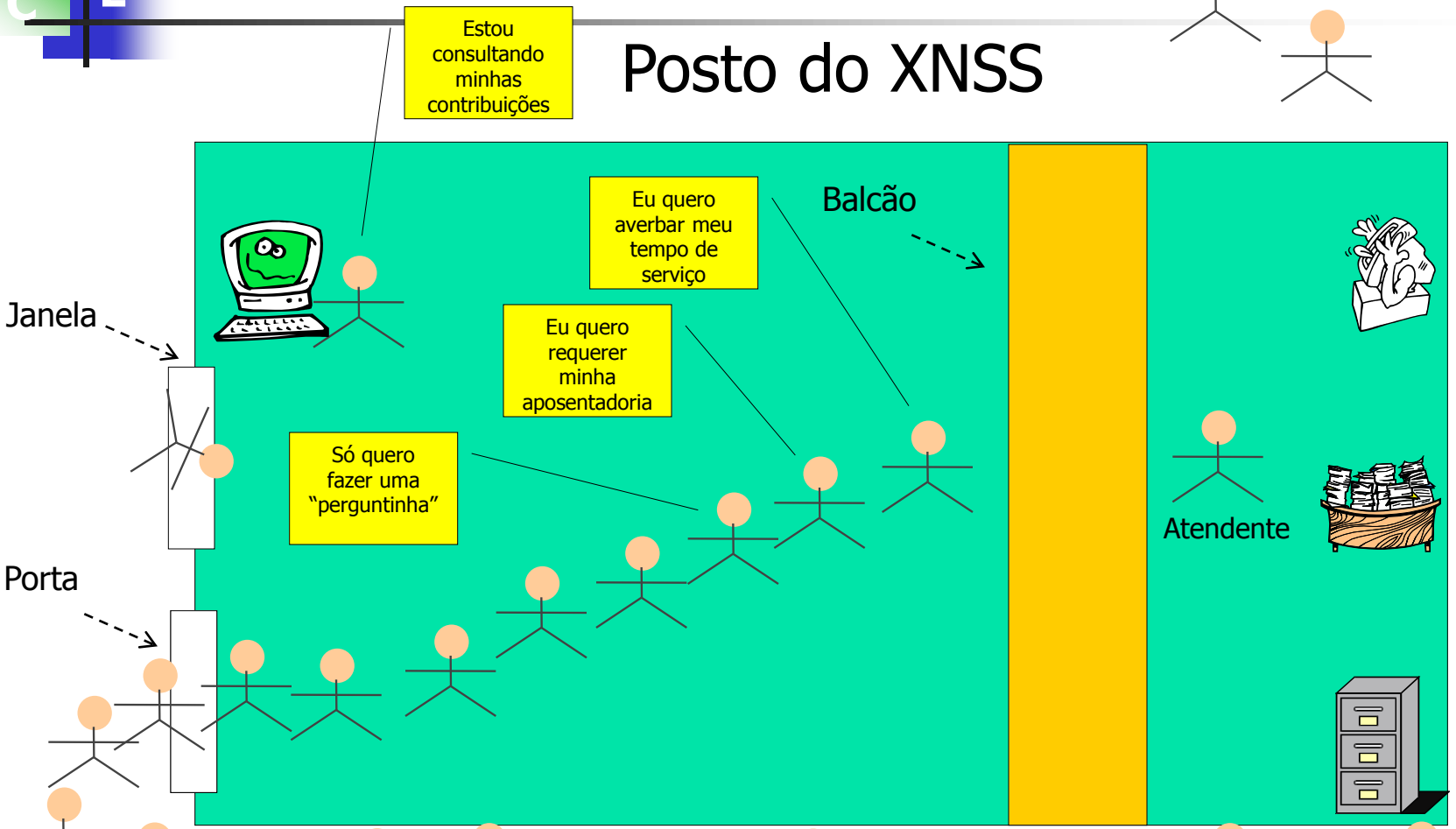
Dois enfoques:

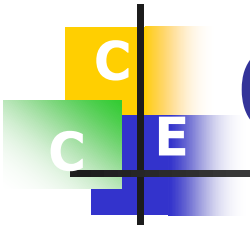
- No negócio:
  - Concentra-se nas relações entre participantes e processos de negócios
  - $\Rightarrow$ casos de uso do negócio
- No sistema:
  - Concentram-se nas relações usuários/sistema
  - Evidenciam a interação com o software
  - $\Rightarrow$ casos de uso do sistema

Ver ilustração a seguir...

Casos de Uso

Posto do XNSS



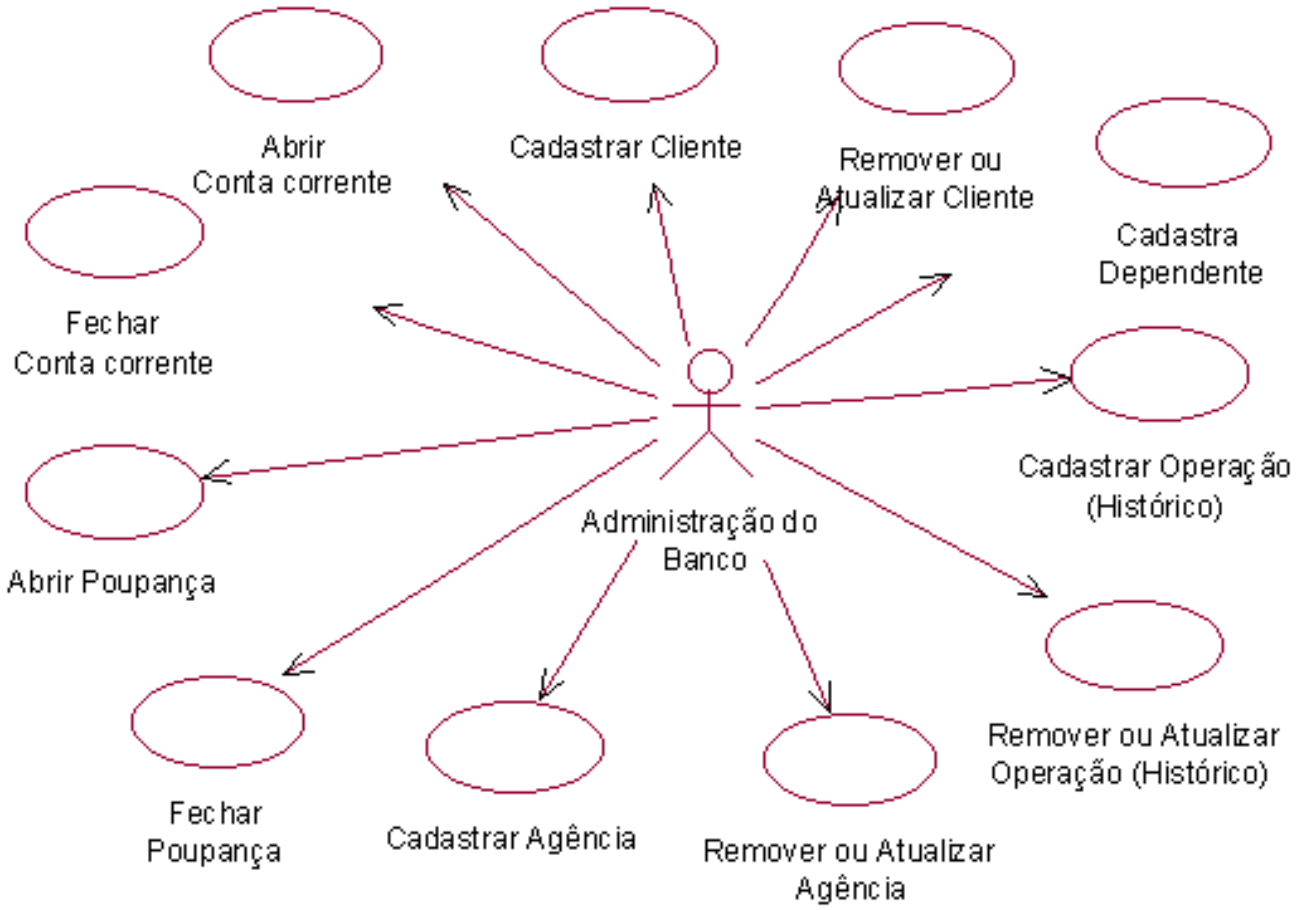


Casos de Uso

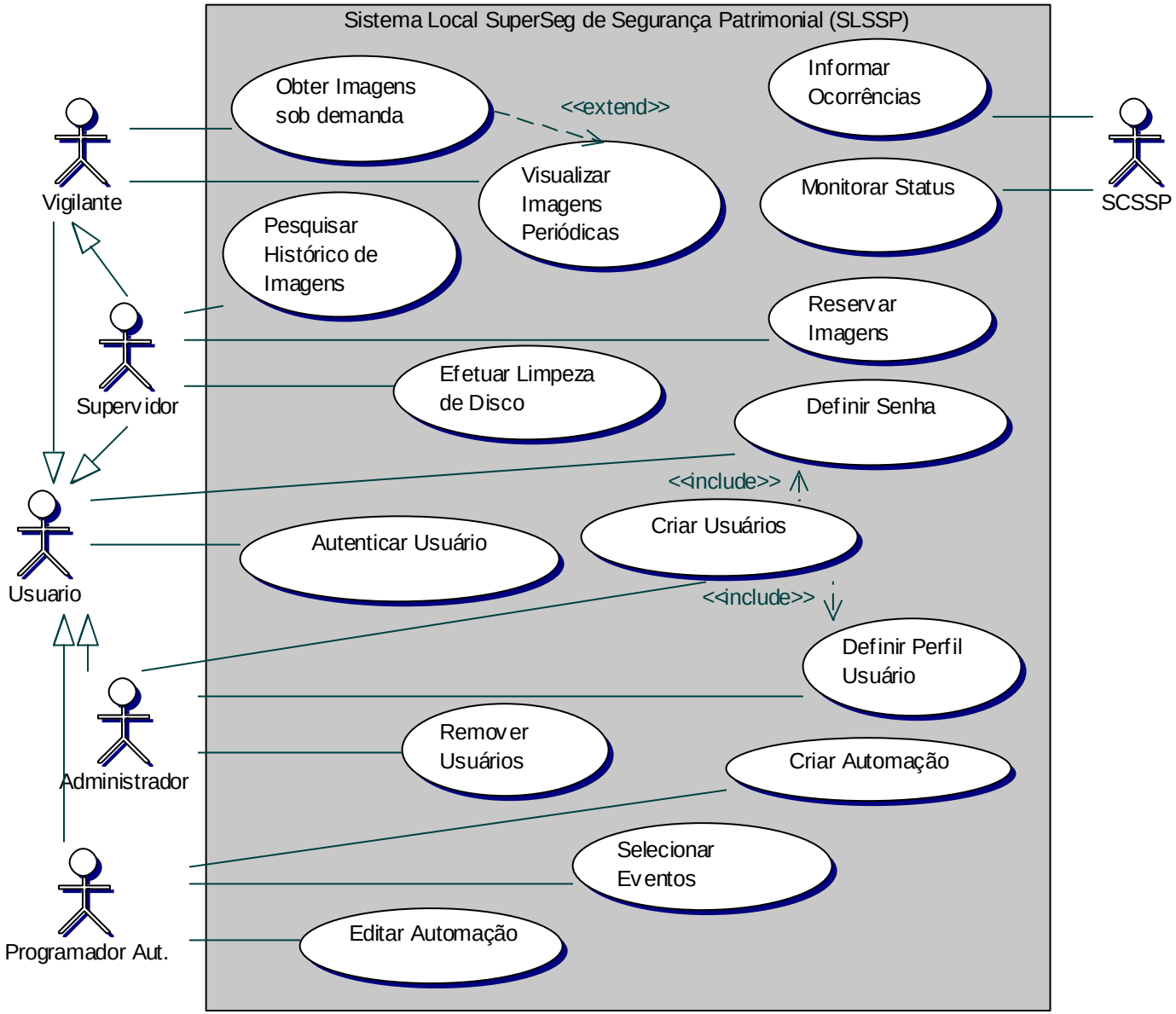
- Casos de uso de sistema
  - Capturam o comportamento de um sistema tal como observado pelos *usuários* externos (atores);
  - Um caso de uso é uma unidade coerente de funcionalidade expressa como uma *transação* entre os usuários e o sistema;
  - Muito usados na definição dos requisitos do sistema;
  - Cada caso de uso especifica uma seqüência de ações, incluindo suas possíveis variações, executadas durante as interações com os respectivos atores.

Casos de Uso

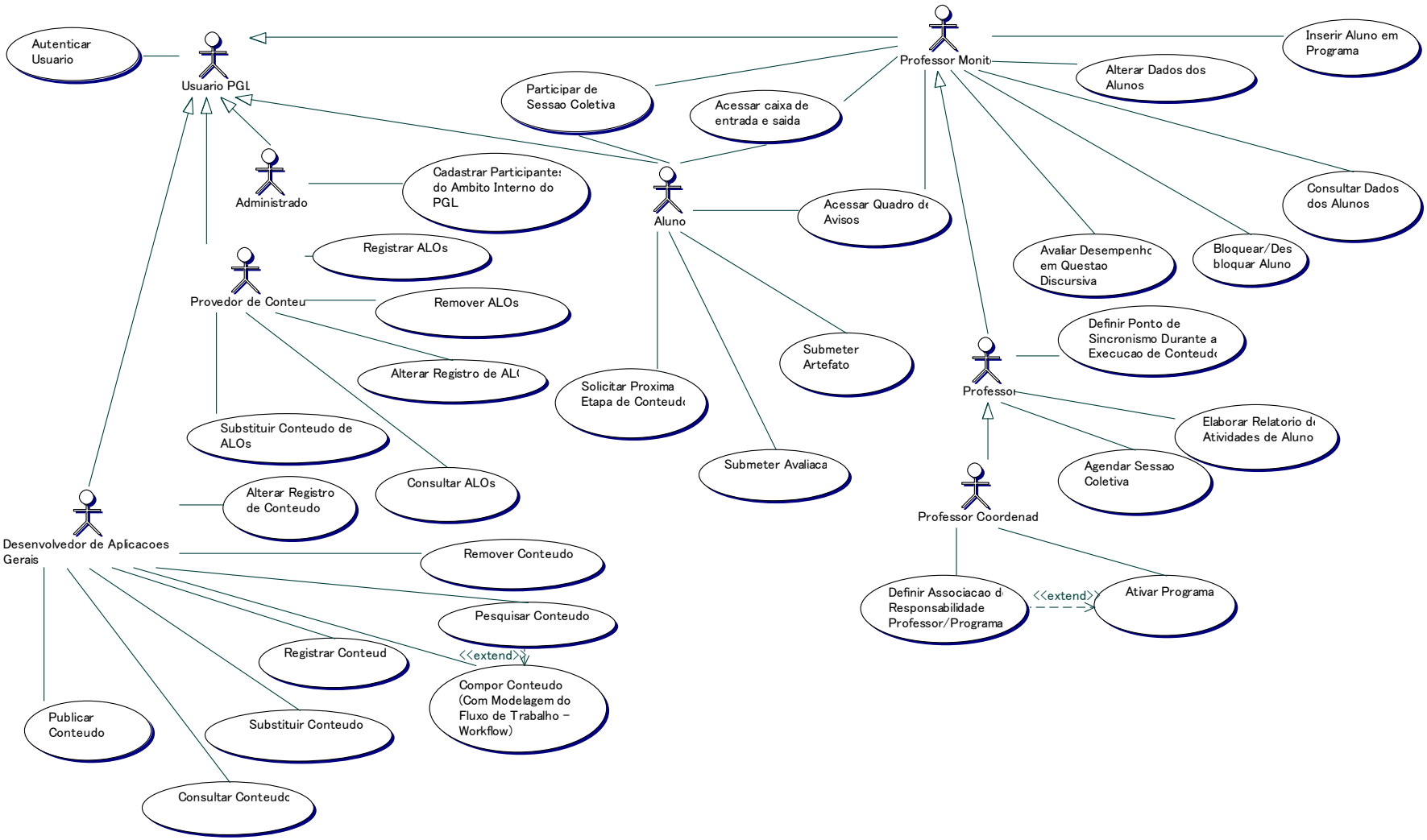
Exemplo de Diagrama

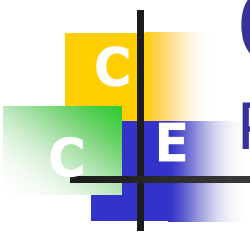


Exemplo 2:


Sistema Local SuperSeg de Segurança Patrimonial (SLSSP)
Obter Imagens sob demanda
<<extend>>
Informar Ocorrências
Visualizar Imagens Periódicas
Monitorar Status
Pesquisar Histórico de Imagens
Reservar Imagens
Efetuar Limpeza de Disco
Definir Senha
<<include>>
Criar Usuários
Autenticar Usuário
<<include>>
Definir Perfil Usuário
Remover Usuários
Criar Automação
Selecionar Eventos
Editar Automação
Vigilante
SCSSP
Supervidor
Usuario
Administrador
Programador Aut.

Exemplo 3:


Autenticar Usuario
Usuario PGL
Professor Monit
Inserir Aluno em Programa
Alterar Dados dos Alunos
Participar de Sessao Coletiva
Acessar caixa de entrada e saida
Administrado
Cadastrar Participantes do Ambito Interno do PGL
Aluno
Acessar Quadro de Avisos
Consultar Dados dos Alunos
Registrar ALOs
Avaliar Desempenho em Questao Discursiva
Bloquear/Desbloquar Aluno
Provedor de Conteu
Remover ALOs
Definir Ponto de Sincronismo Durante a Execucao de Conteud
Submeter Artefato
Alterar Registro de AL
Solicitar Proxima Etapa de Conteud
Professor
Substituir Conteudo de ALOs
Elaborar Relatorio de Atividades de Aluno
Consultar ALOs
Submeter Avaliaca
Agendar Sessao Coletiva
Alterar Registro de Conteudo
Professor Coordenad
Desenvolvedor de Aplicacoes Gerais
Remover Conteudo
Definir Associacao d Responsabilidade Professor/Programa
<<extend>>
Ativar Programa
Pesquisar Conteudo
Registrar Conteud
<<extend>>
Compor Conteudo (Com Modelagem do Fluxo de Trabalho – Workflow)
Publicar Conteudo
Substituir Conteudo
Consultar Conteudo



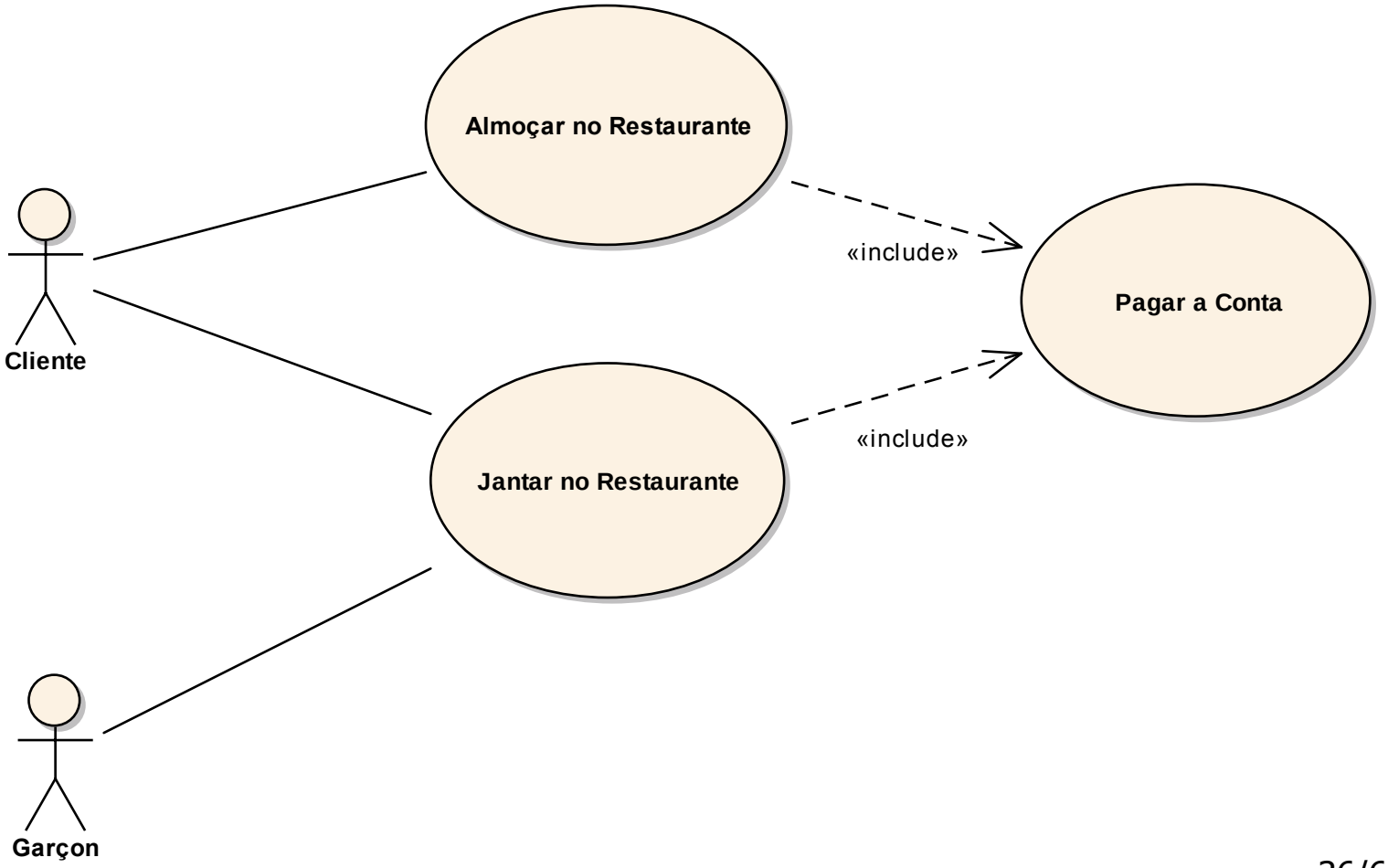
Casos de Uso

Principais Relacionamentos

- Entre casos de uso:
  - Ocorrem quando há uma parte do comportamento que é semelhante em mais de um caso de uso e você não quer ficar copiando a descrição desse comportamento (fatoração);
  - Inclusão:
    - Ocorre obrigatoriamente.
  - Extensão:
    - Ocorre opcionalmente.
- Entre atores:
  - Generalização/especialização

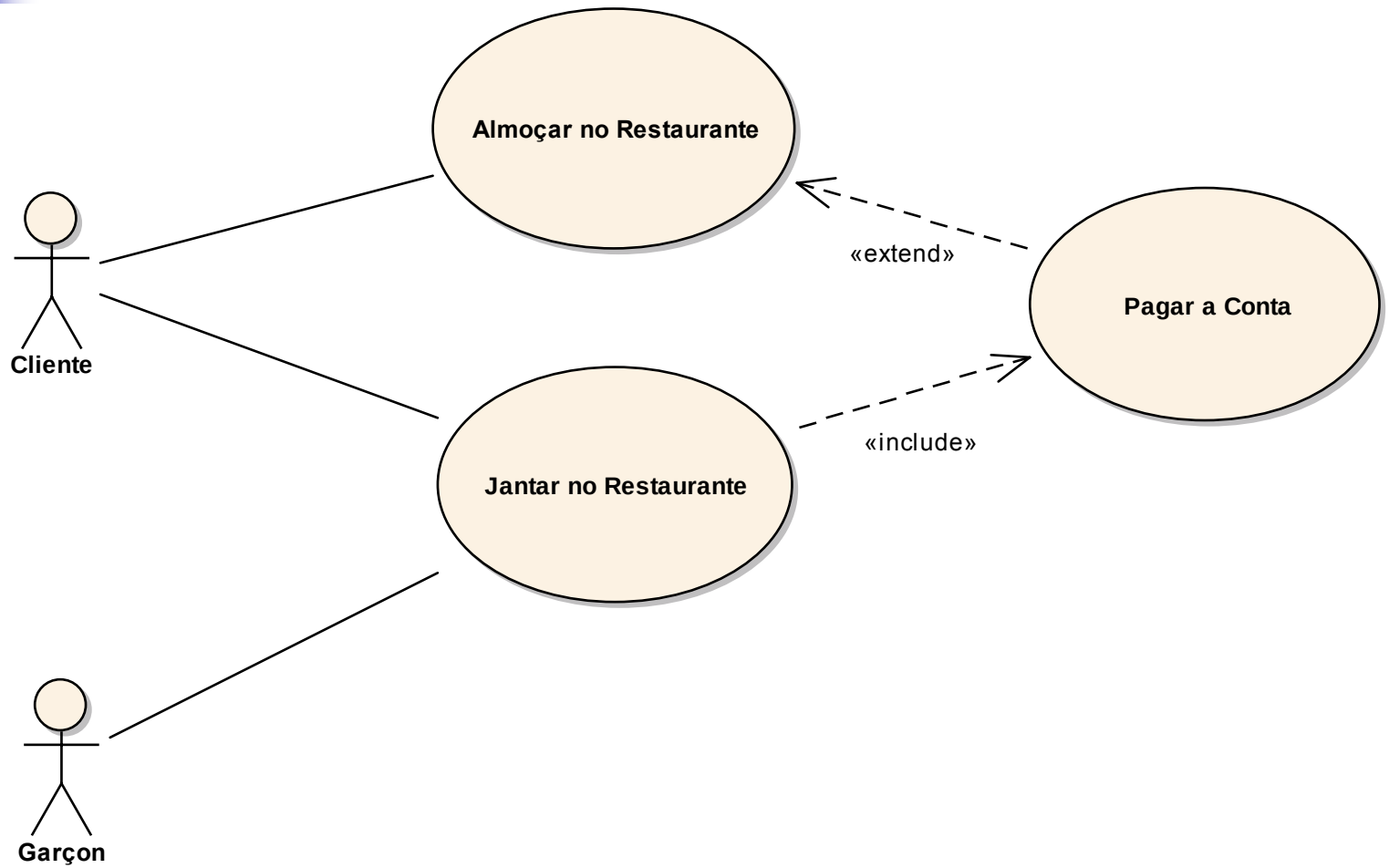
Casos de Uso

Inclusão



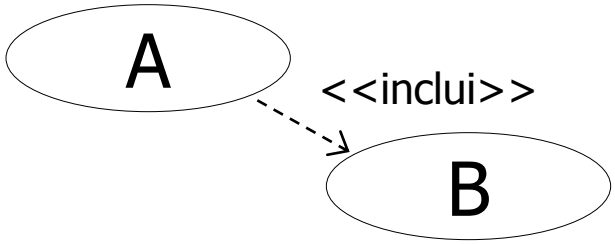
Casos de Uso

Extensão

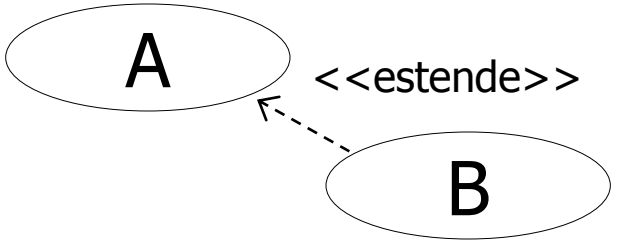


Casos de Uso
Inclui ou estende?

- Na prática fazemos a pergunta:

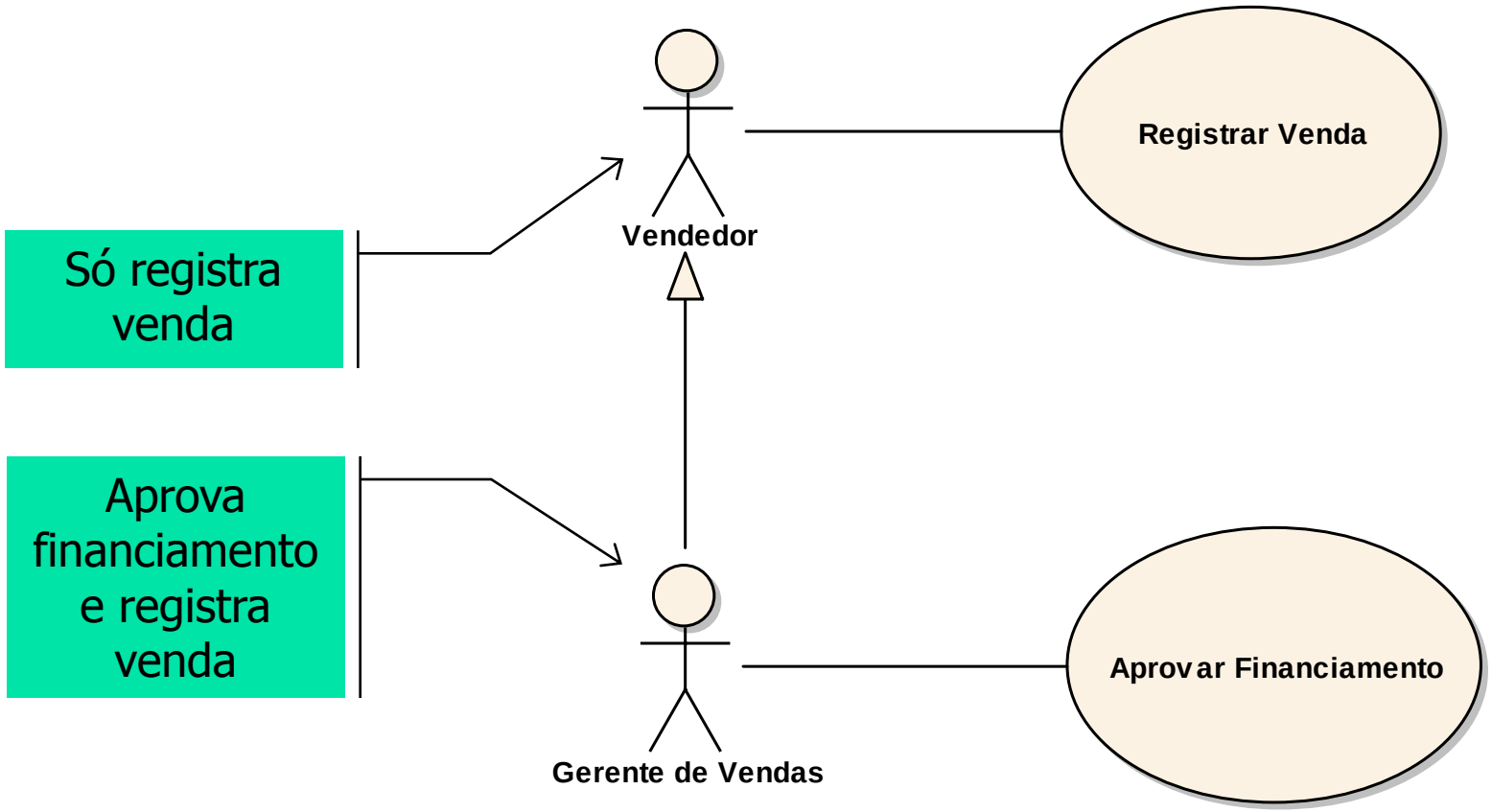


A inclusão ocorre sempre?
Se a resposta for "Sim", deixamos como está (com o <<include>>).
Se a resposta for "Não", trocamos o sentido da seta e substituímos o <<include>> pelo <<estende>>.



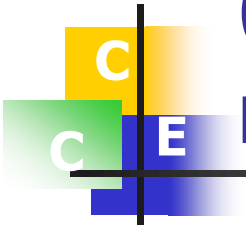
Casos de Uso

Generalização/especialização de atores



Casos de Uso
Descrições

- U.C. precisam ser descritos.As descrições podem ser:
  - Em alto nível (descrição geral, resumida), geralmente feita no início do processo de captura dos requisitos, ou
  - Detalhada (ou expandida), não procedimental, e é refinada ao longo do restante do projeto, ou quando há riscos maiores de erros de definição.



Casos de Uso
Descrição

- Existem muitas formas de descrição das seqüências de operações de um caso de uso. Você pode inventar e (idealmente) padronizar a sua. A UML não especifica uma forma correta.
  - Existem *templates* prontos na Internet (e.g. Alistair Cockburn em
    - http://alistair.cockburn.us/usecases/uctempla.doc
    - http://alistair.cockburn.us/usecases/uctempla2.dot
  - Em geral as empresas adotam seus próprios padrões.

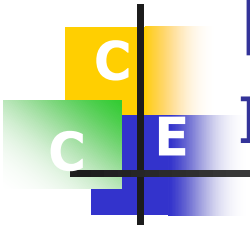


Diagrama de Classes
Introdução

- É uma visão estática do sistema.
- Descreve relações atemporais entre elementos do domínio.
- Compõe-se de classes, relacionamentos entre elas, restrições, etc.
- Podem modelar o domínio sob três perspectivas:
  - Conceitual.
  - Especificação.
  - Implementação.
- Cada perspectiva representa o domínio com graus diferentes de abstração

Diagrama de Classes

Classes, atributos, operações e relacionamentos

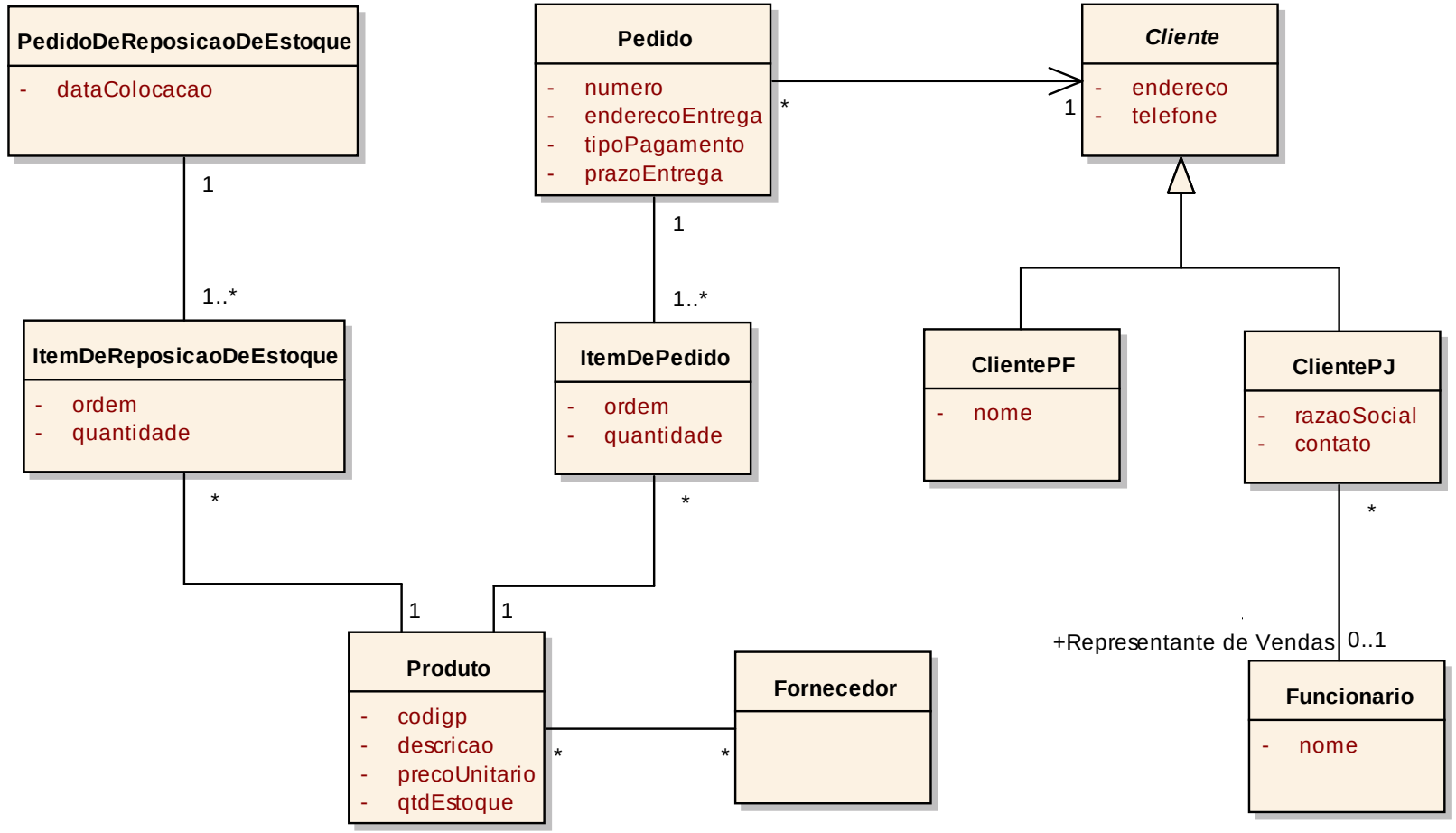


Diagrama de Classes

Associações: leitura dos rótulos e multiplicidades

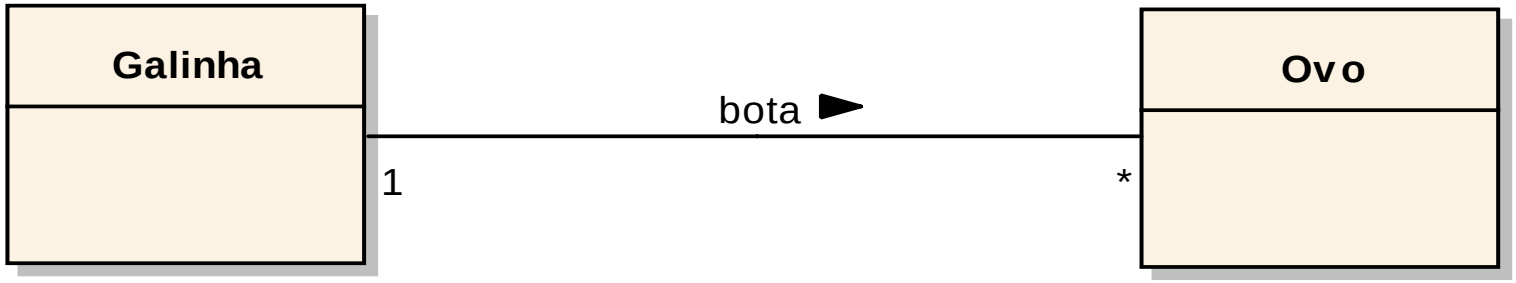
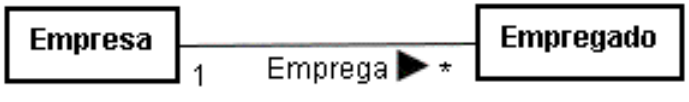


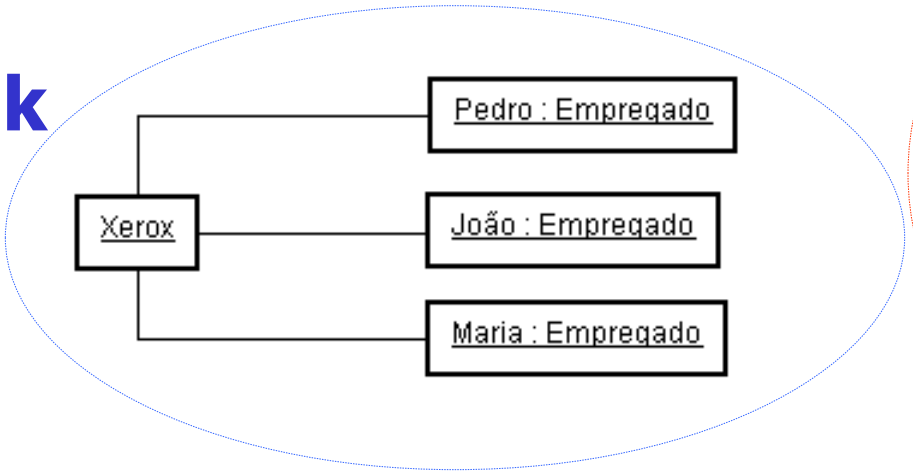
Diagrama de Classes

Associações: semântica das multiplicidades

Representam regras com respeito às relações entre os objetos.



Ok



~Ok

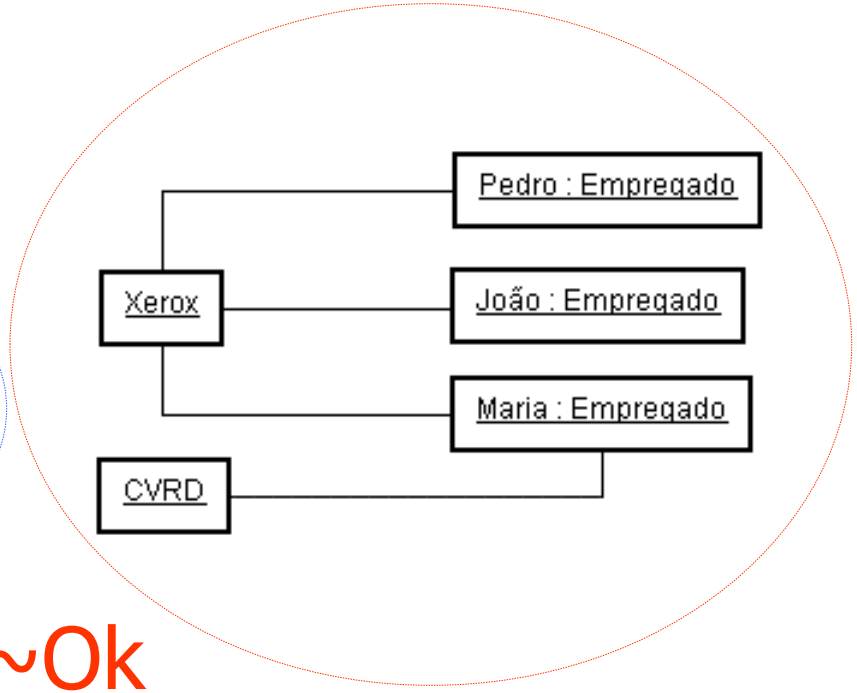


Diagrama de Classes

Auto-associações

Pode haver auto-associação. Exemplos:

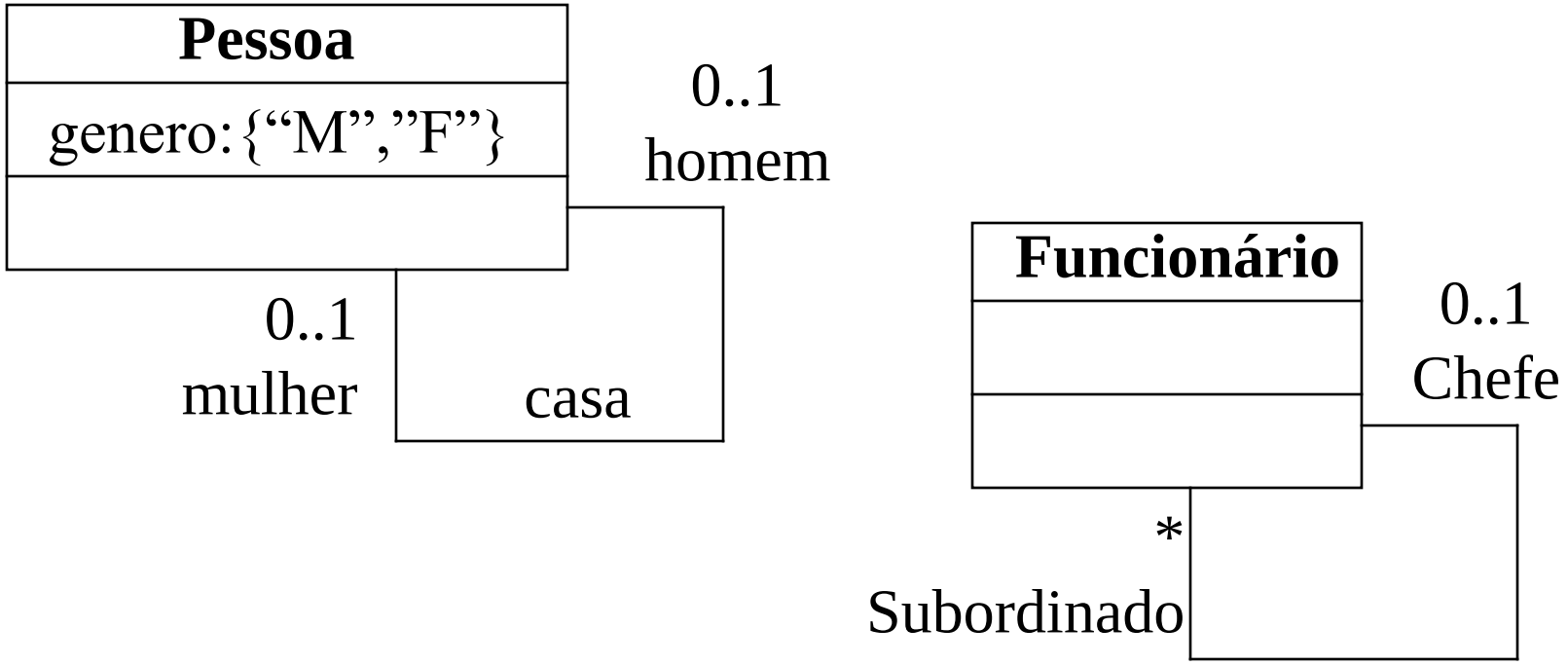


Diagrama de Classes

Classes de associação

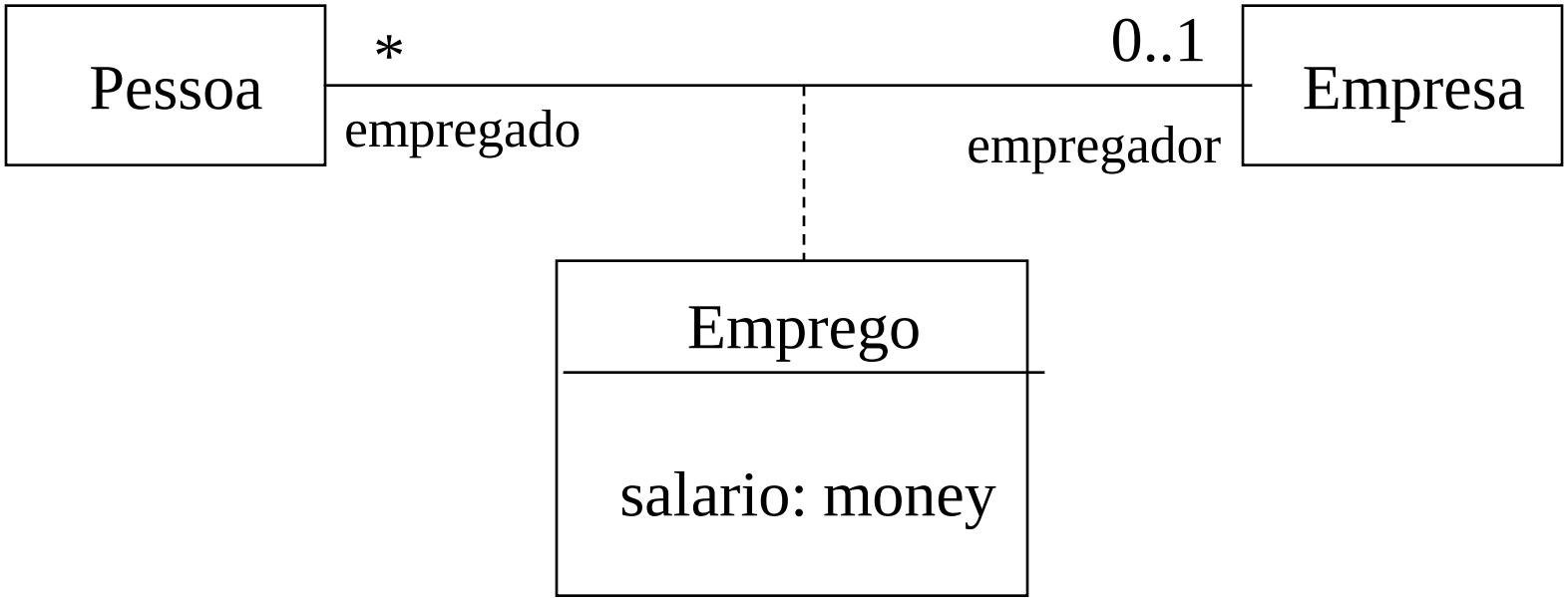
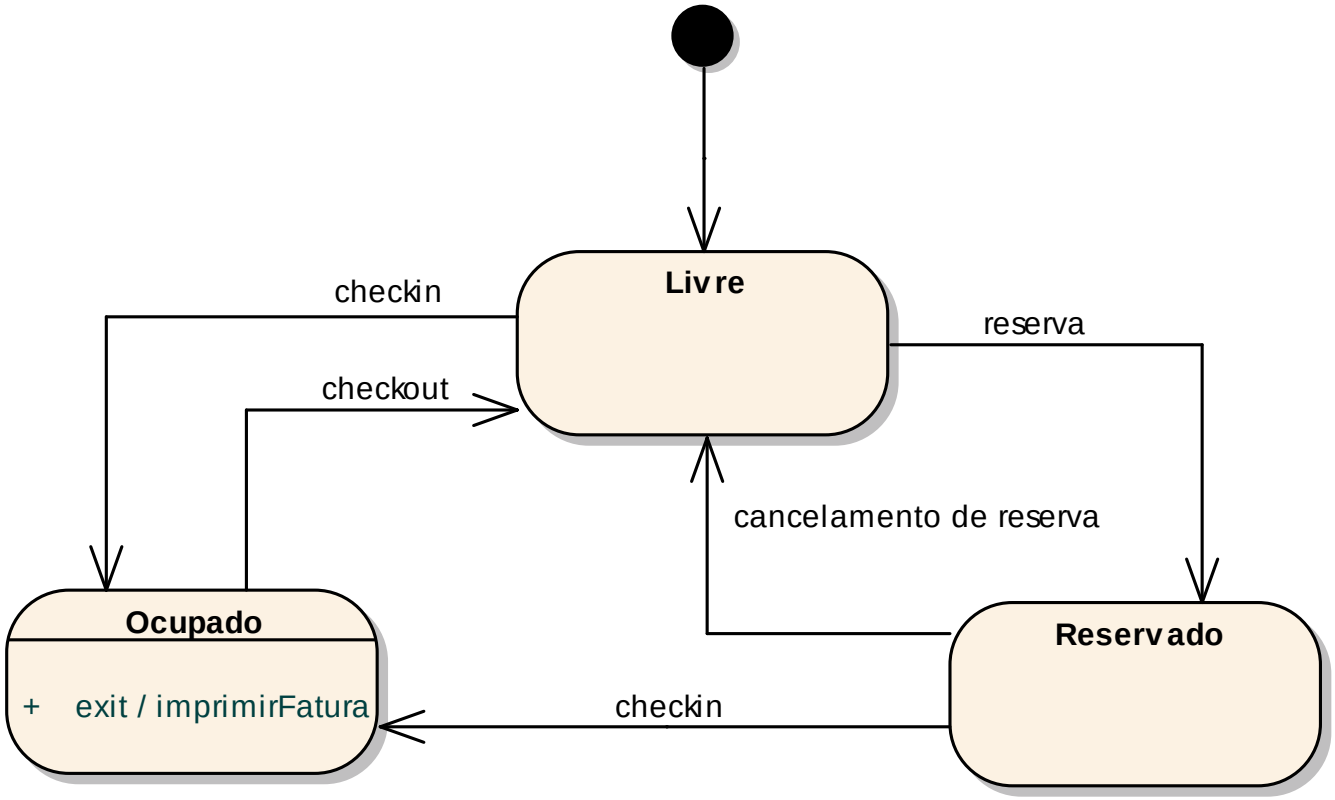


Diagrama de Máquina de Estados



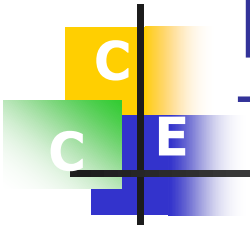


Diagrama de Máquina de Estados

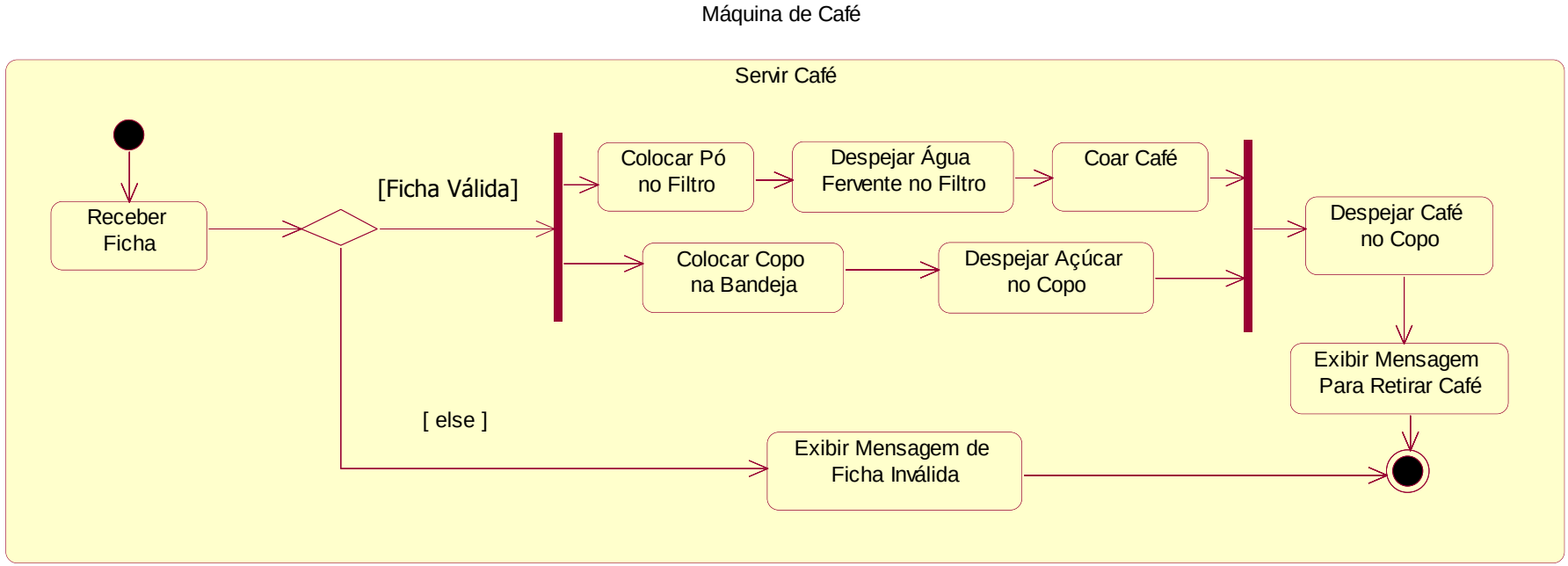
Tipos de Estados

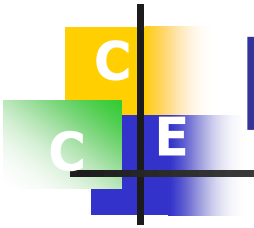
- Estados podem ser:
  - De atividade: nome bom » verbo no gerúndio (lendo, calculando ...)
  - De satisfação de condição: nome bom » verbo no particípio (autenticado, lançado...)
  - De espera: nomes bons » aguardando ou esperando

Diagrama de Máquina de Estados
Transições

- Transições:
  - Passagem de um estado para outro;
  - Ocorrem a partir de *eventos;*
  - Envolvem, em geral, ações de curta duração;
  - Ações **não** podem ser interrompidas;
  - São rotuladas.

Diagrama de Atividade





DA

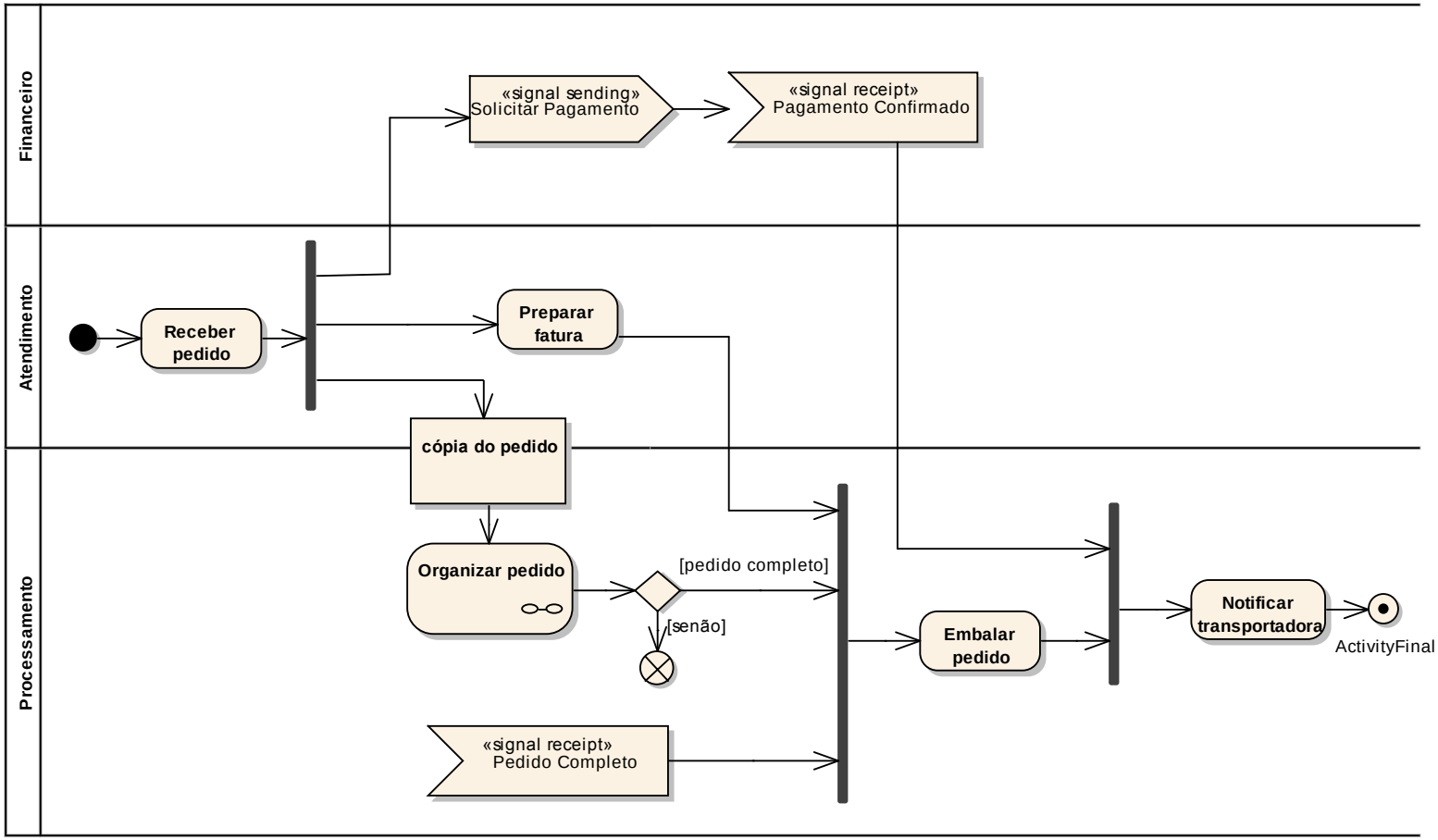
- Enfocam o fluxo de controle entre ações do sistema (visão dinâmica);
- Úteis para descrição de comportamentos com muito processamento em paralelo;
- Úteis para modelagem de programas concorrentes, onde se projetam graficamente as *threads* e pontos de sincronismo.
- Úteis para a modelagem de processos de negócio.

DA

Partições ou Raias de natação

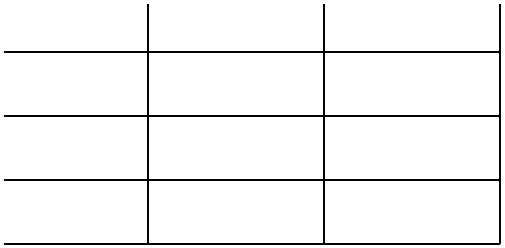
- Partições:
  - Raias (*swimlanes*) são usadas quando há necessidade de se indicar quem executa as atividades.

DA
Raias de natação



DA
Raias de natação

Raias podem ser hierarquizadas/multidimensionadas:



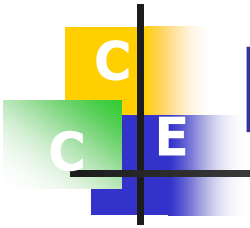
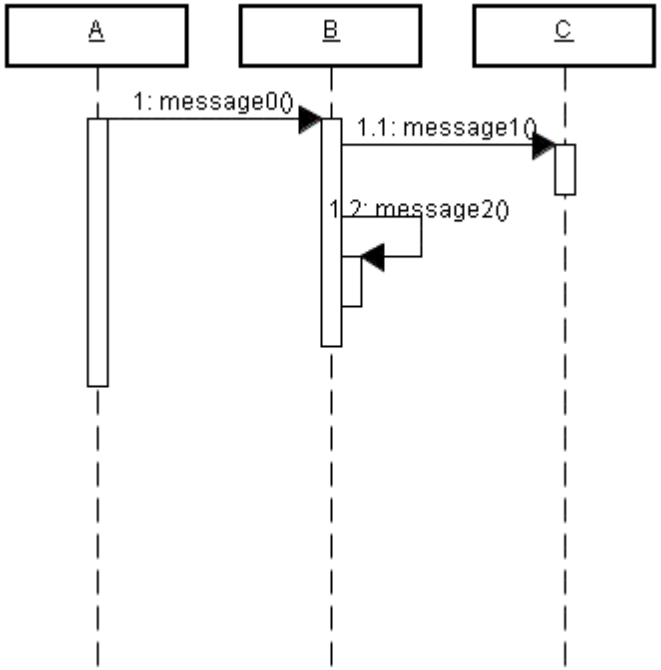


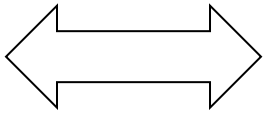
Diagrama de Sequência

- Descrevem como grupos de objetos colaboram em algum comportamento do sistema;
- Servem para modelar o "funcionamento" (programação) do sistema.

Diagrama de Sequência

- Melhores que o Diagrama de Comunicação para apresentar as responsabilidades de cada objeto, especialmente quando o aspecto da ordenação temporal é relevante.





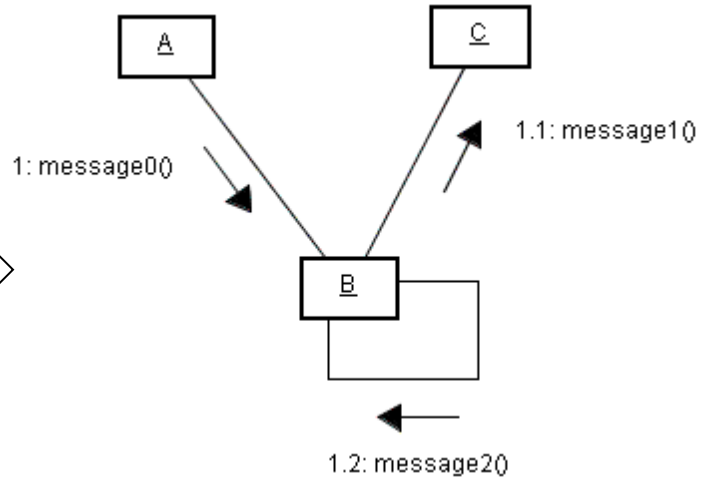


Diagrama de Sequência

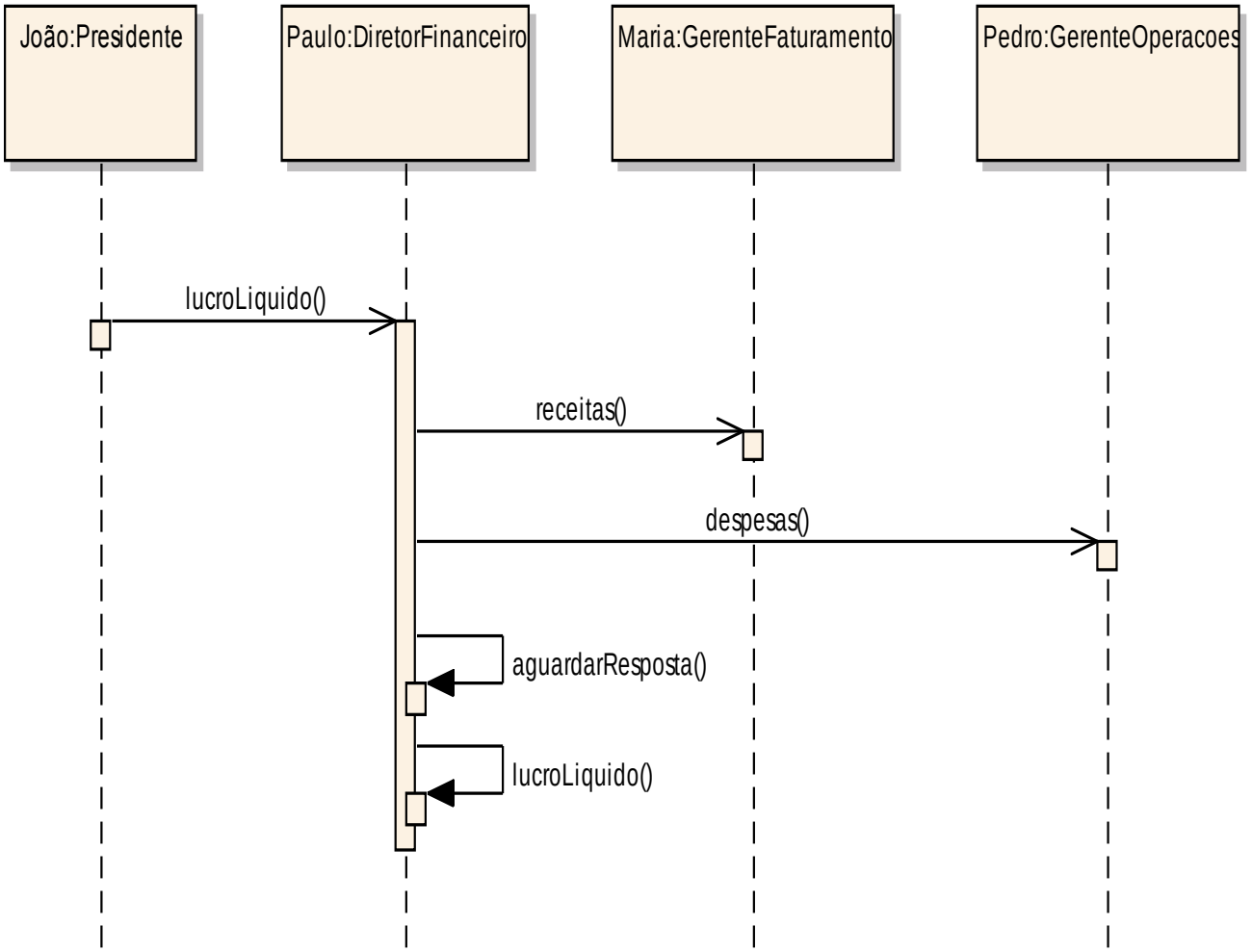
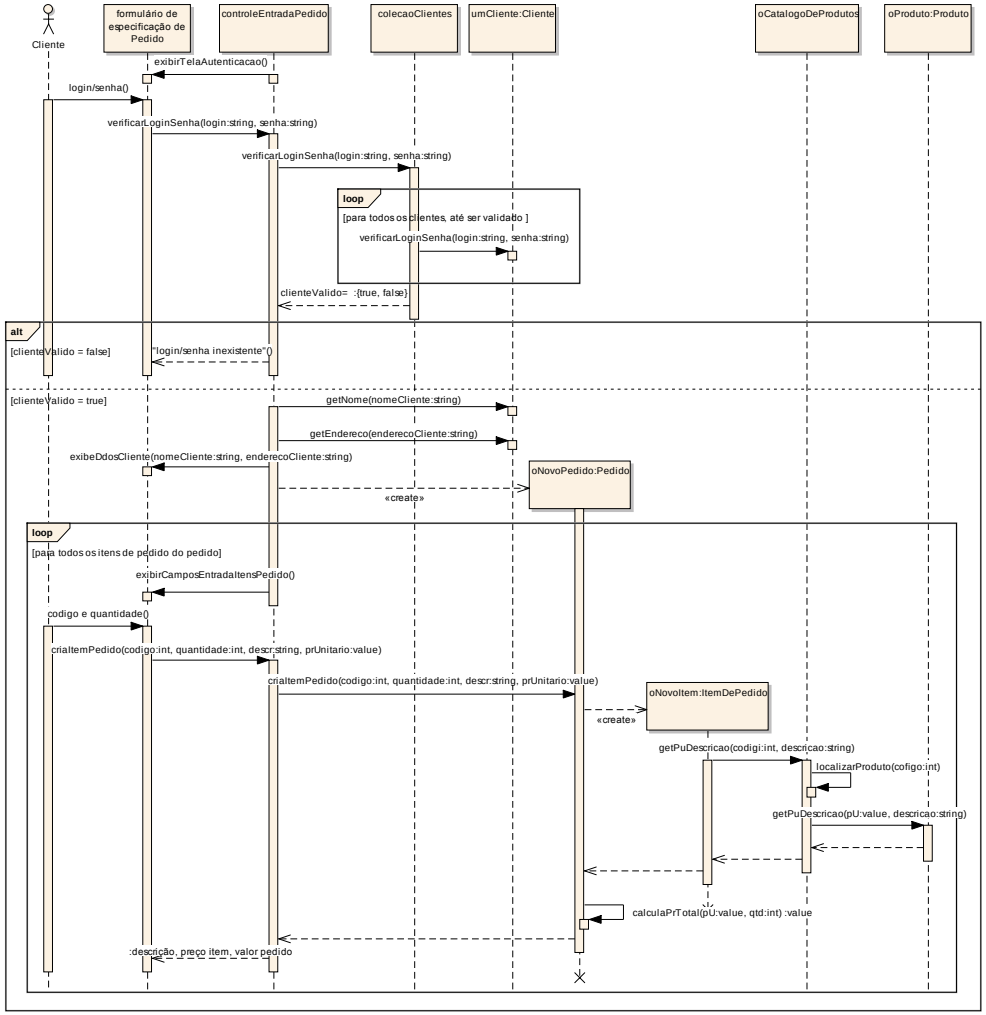
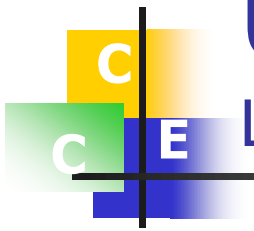


Diagrama de Sequência

Em geral são visualmente complexos





UML
Lembrete

Próxima aula: Arquitetura de Software
Patterns e Frameworks
MDA