

# Engenharia de Software

**Curso Superior de Tecnologia em Desenvolvimento de  
Software Multiplataforma**

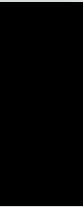
**Aula 08**

**Prof. Claudio Benossi**




# 3. Unidade

## Fundamentos da Modelagem de Software





# Engenharia de Software

- As economias de todas as nações desenvolvidas são dependentes de softwares.
  - Mais e mais sistemas são controlados por software.
  - A engenharia de software se preocupa com teorias, métodos e ferramentas para desenvolvimento de softwares profissionais.
- 



# Engenharia de Software

## Visão da Estrutura

### Diagramas recomendados:

- Diagrama de Classe
- Diagrama de Estado
- Gráficos
- Diagrama de Atividade



# Engenharia de Software

## Visão dos Processos

### Diagramas recomendados:

- Mapas de Processos e/ou Diagramas de Processos
- Diagrama de Atividade
- Diagrama de Sequencia
- Diagrama de Processo
- Diagrama de Caso de Uso

# Introdução ao UML

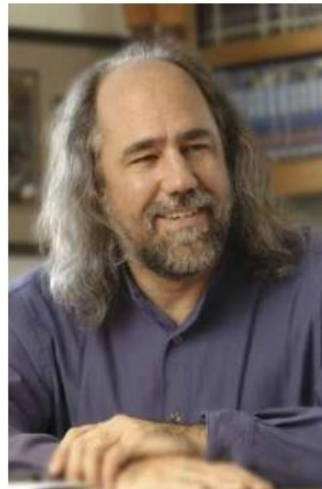
Originou-se da necessidade de produzir sistemas orientado a objetos, pois haviam linguagem OO e todas as técnicas até o momento eram voltadas para a análise estruturada.



# Introdução ao UML

Foram criadas algumas técnicas:

- Grady Booch – técnica Booch.
- James Rumbaugh – técnica OMT.
- Ivar Jacobson – técnica OOSE.



grady booch




ivar jacobson



james rumbaugh



# Introdução ao UML

- Em 1994, Booch, Rumbaugh e Jacobson uniram forças para combinar suas metodologias populares – Booch, OMT e OOSE.
  - Em 1996 foi apresentada a UML (Linguagem de Modelagem Unificada).
  - UML é a junção do que havia de melhor nas três metodologias.
- 



# Introdução ao UML

*Unificação dos  
métodos para a  
criação de um novo  
padrão*

BOOCH

OOSE

UML

OMT

- Diagrama de Estados
- Diagrama de Objetos (colaboração).
- Diagrama de Processos (desenvolvimento).
- Diagrama de Módulos (componentes).

- Caso de Uso.
- Subsistemas (package).
- Diagrama de Interações.
- Miniespecificação.

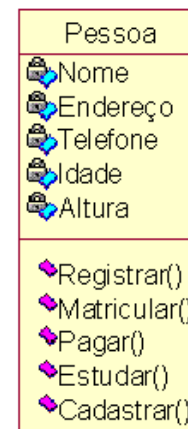
- Diagrama de Estados.
- Diagrama de Classes.

# Introdução ao UML

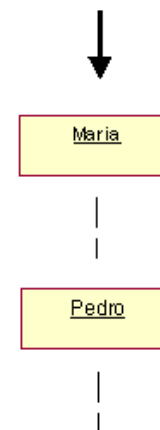
A UML é uma tentativa de padronizar a modelagem orientada a objetos de uma forma que qualquer sistema, seja qual for o tipo, possa ser modelado corretamente, com consistência, fácil de se comunicar com outras aplicações, simples de ser atualizado e compreensível.



## Classe



## Objetos



ATRIBUTOS  
MÉTODOS

# Introdução ao UML

Existem várias metodologias de modelagem orientada a objetos que até o surgimento da UML causavam uma guerra entre a comunidade de desenvolvedores orientados a objetos.



# Introdução ao UML

A UML acabou com esta guerra trazendo as melhores ideias de cada uma destas metodologias, e mostrando como deveria ser a migração de cada uma para a UML.



# Introdução ao UML

**Os objetivos da UML são:**

1. A modelagem de sistemas (não apenas de software) usando os conceitos da orientação a objetos;



# Introdução ao UML

**Os objetivos da UML são:**

2. Estabelecer uma união fazendo com que métodos conceituais sejam também executáveis;



# Introdução ao UML

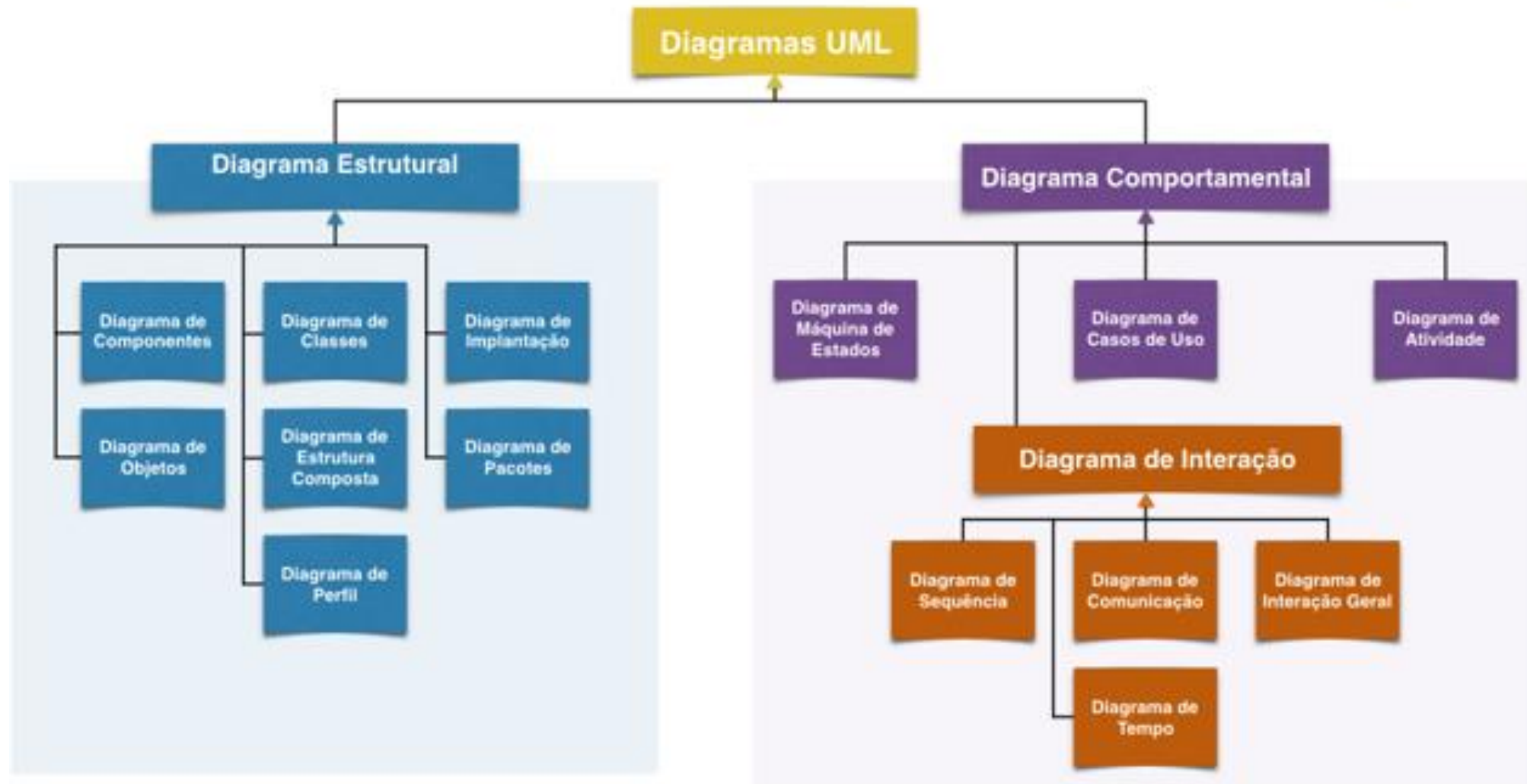
**Os objetivos da UML são:**

3. Criar uma linguagem de modelagem usável tanto pelo homem quanto pela máquina.





# Introdução ao UML







# Diagramas Estáticos e Dinâmicos

Todos os sistemas possuem uma estrutura estática e um comportamento dinâmico.

A UML suporta modelos estáticos (estrutura estática), dinâmicos (comportamento dinâmico) e funcional.



# Diagramas Estruturais – Estático

- ▷ Diagrama de classe.
- ▷ Diagrama de objeto.
- ▷ Diagrama de componentes.
- ▷ Diagrama de implementação.



# Diagramas Comportamentais - Dinâmico

- ▷ Diagrama caso de uso.
- ▷ Diagrama de sequência.
- ▷ Diagrama de colaboração.
- ▷ Diagrama de gráfico de estados.
- ▷ Diagrama de atividades.



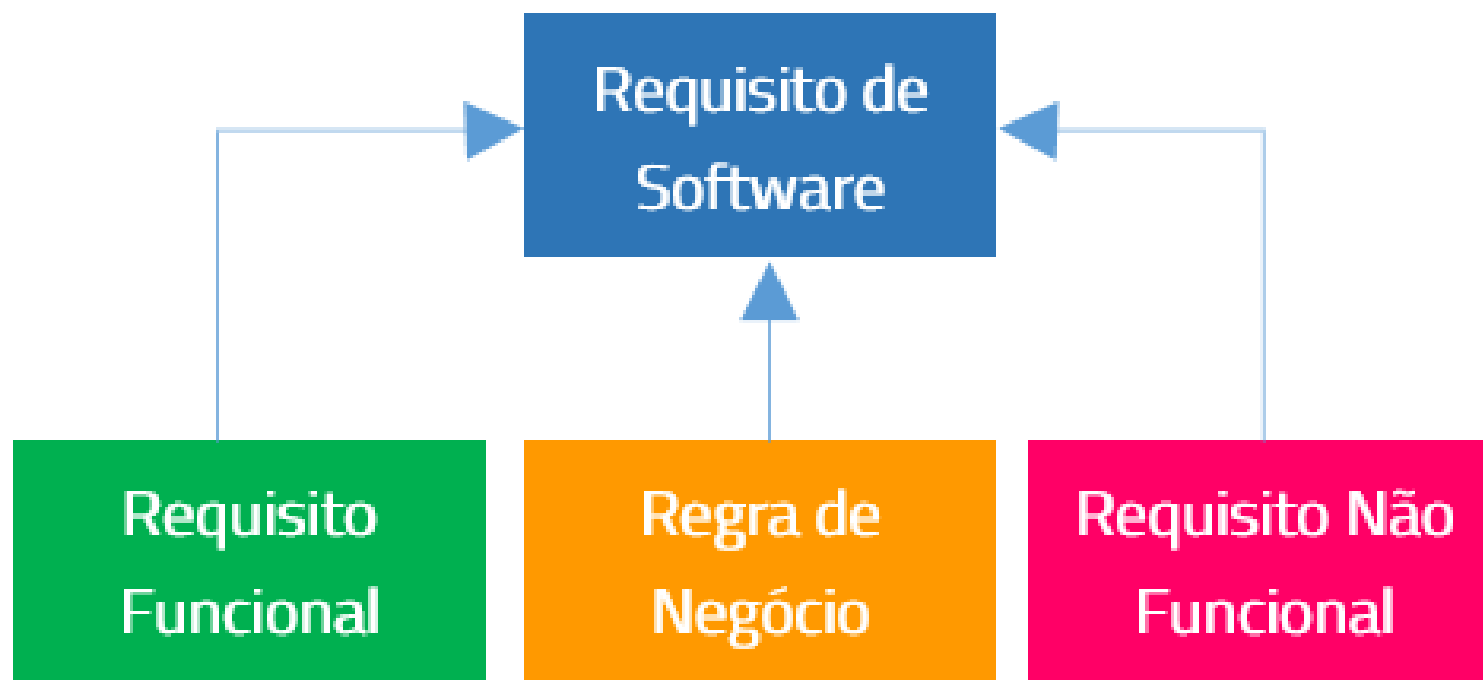
# Quais Diagramas utilizar?

## **Depende da complexidade do seu sistema**

- De forma incremental
  - Ampliando os diagramas uma parte de cada vez.
- De forma interativa
  - Repetindo o processo de projetar uma pequena parte e construí-la.

## Levantamento e especificação dos requisitos

Significa buscar todas as informações possíveis sobre aquilo que se espera do sistema.





# Levantamento e especificação dos requisitos

## Informações fornecidas por:


- Usuários;
- Análise de documentos;
- Outros sistemas;
- Observação dos usuários ao interagirem com o sistema atual.



# Requisitos Funcionais

- São aqueles relacionados aos serviços que o sistema deve fornecer.
- Especificam o que o sistema deve fazer.

## Exemplos:

- O sistema deve realizar venda.
  - O sistema deve permitir devolver filme.
  - O sistema deve permitir cancelar pedido.
- 




# Regras de Negócio

- Normas condições ou imposições de como o sistema deve funcionar;
- Regra de negócio é o que define a forma de fazer o negócio, refletindo a política interna, o processo definido e/ou as regras básicas de conduta, ou seja, é um conjunto de instruções que os usuários já seguem e que o sistema a ser desenvolvido deve contemplar.





# Requisitos Não Funcionais

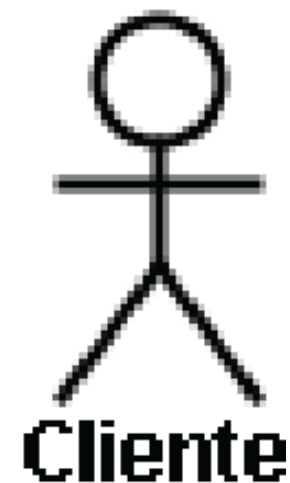
- Referem-se às restrições sobre as funções e as operações que o sistema deve fornecer ou realizar.
  - Eles tratam de rotinas de backup, autenticação no sistema, desempenho, interface etc.
  - O sistema deve ter interface web ou se o sistema deve realizar backup diário.
- 

# Atores

- São as entidades que interagem com o sistema.
- É sempre o ator que causa o estímulo.
- Sempre está fora do sistema.

Atores podem ser:

- Pessoas.
- Outros sistemas.
- Hardware periférico.





# Objetivos do Caso de Uso

Descrever os requisitos funcionais do sistema, mostrando que desenvolvedores e clientes estão de acordo sobre o que será desenvolvido.

Fornecer uma visão clara sobre o que o sistema deve fazer.

Fornecer uma base para a execução de testes que verifiquem se o sistema trabalha apropriadamente.





# Caso de Uso

Os casos de usos representam as interações dos atores com o sistema.

Um caso de uso captura uma funcionalidade do sistema.

Não revelam a estrutura e o comportamento interno do sistema.



# Caso de Uso

Há várias formas para descrever casos de uso:

- um texto contínuo;
- uma sequência numerada;
- a utilização de tabelas.



## Texto Contínuo

O cliente chega à livraria. No terminal de consulta, o sistema mostra as formas de pesquisa (por título da obra, pelo nome do autor, pelo nome da editora).



# Sequencia Numerada

1. Cliente chega à livraria e dirige-se a um terminal de consulta.
2. O sistema exibe as formas de pesquisa (por título da obra, pelo nome do autor, pelo nome da editora).
3. O cliente escolhe a forma de pesquisa que lhe interessa.
4. O sistema exibe as informações sobre o produto desejado.



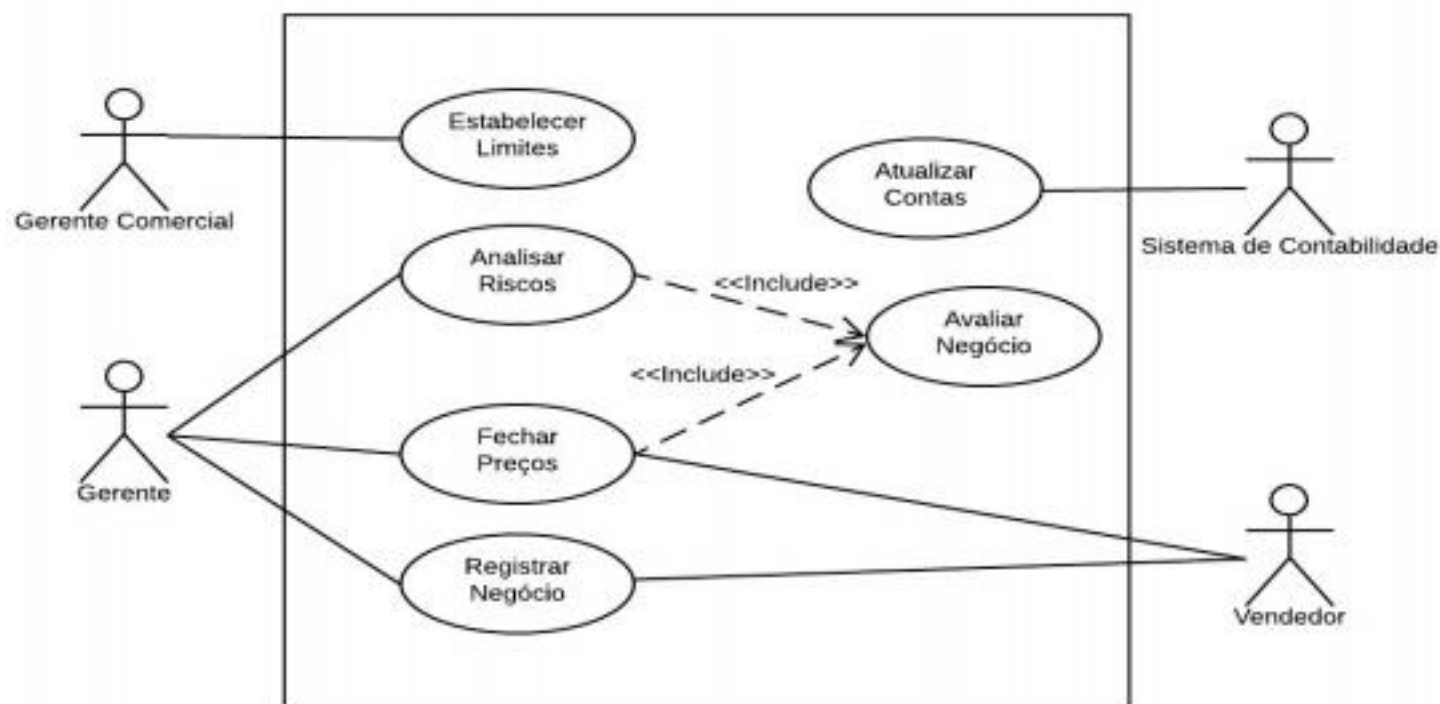
# Tabelas

CLIENTE	SISTEMA
1. Cliente chega à livraria e dirige-se a um terminal de consulta.	2. Sistema exibe as formas de pesquisa (por título da obra, pelo nome do autor, pelo nome da editora).
3. Cliente escolhe a forma de pesquisa que lhe interessa.	4. Sistema exibe as informações sobre o produto desejado.



# Diagrama de Caso de Uso

A modelagem de um diagrama use-case é uma técnica usada para descrever e definir os requisitos funcionais de um sistema.





# Diagrama de Caso de Uso

Eles são escritos em termos de atores externos, use-cases e o sistema modelado.

Os atores representam o papel de uma entidade externa ao sistema como um usuário, um hardware, ou outro sistema que interage com o sistema modelado.



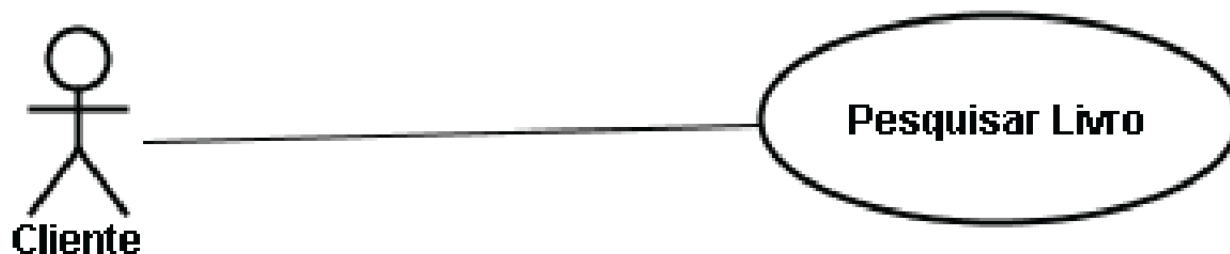
# Diagrama de Caso de Uso

Os atores iniciam a comunicação com o sistema através dos use-cases, onde o use-case representa uma sequência de ações executadas pelo sistema e recebe do ator que lhe utiliza dados tangíveis de um tipo ou formato já conhecido, e o valor de resposta da execução de um use-case (conteúdo) também já é de um tipo conhecido, tudo isso é definido juntamente com o use-case através de texto de documentação.

# Diagrama de Caso de Uso

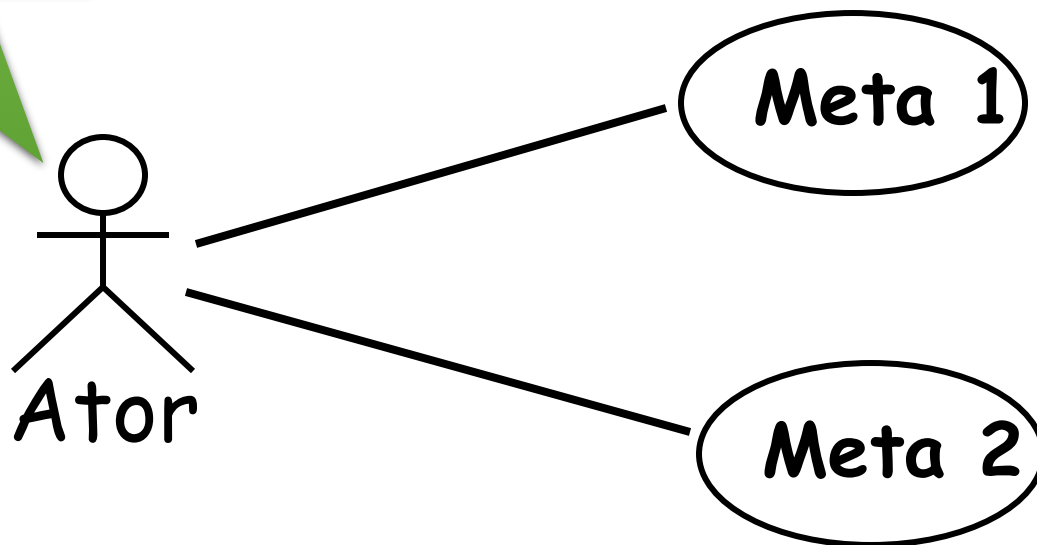
Representa, graficamente, todos os casos de uso de um sistema, utilizando a linguagem UML.

Por meio dele é possível visualizar, em um alto nível de abstração, quais os elementos (atores) interagem com o sistema em cada funcionalidade.



# Diagrama de Caso de Uso

Quais metas eu quero  
atingir ao utilizar o  
sistema?





# Diagrama de Caso de Uso

- O nome do caso de uso deve ser único.
- Deve estar na perspectiva do ator que dispara o caso de uso.
- Deve iniciar com o verbo no infinitivo.



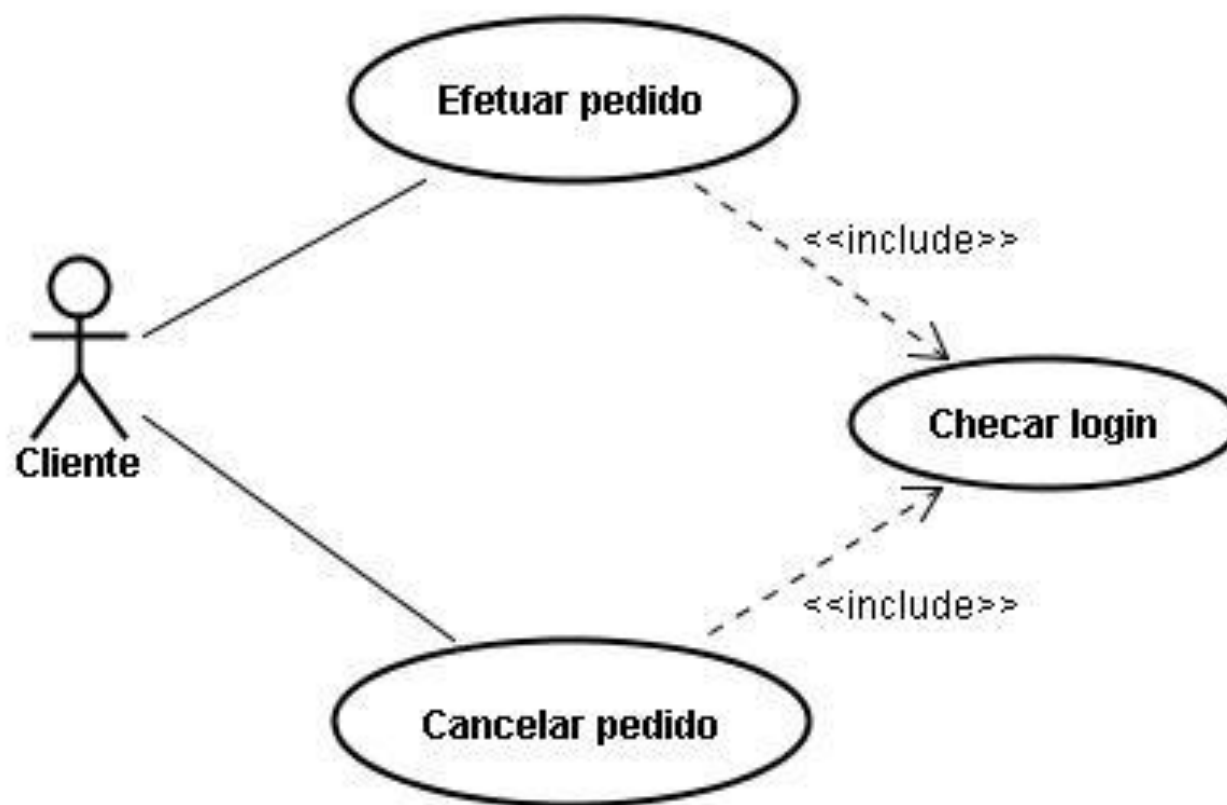
Fazer  
Matrícula

Consultar  
Notas

Realizar  
Saque

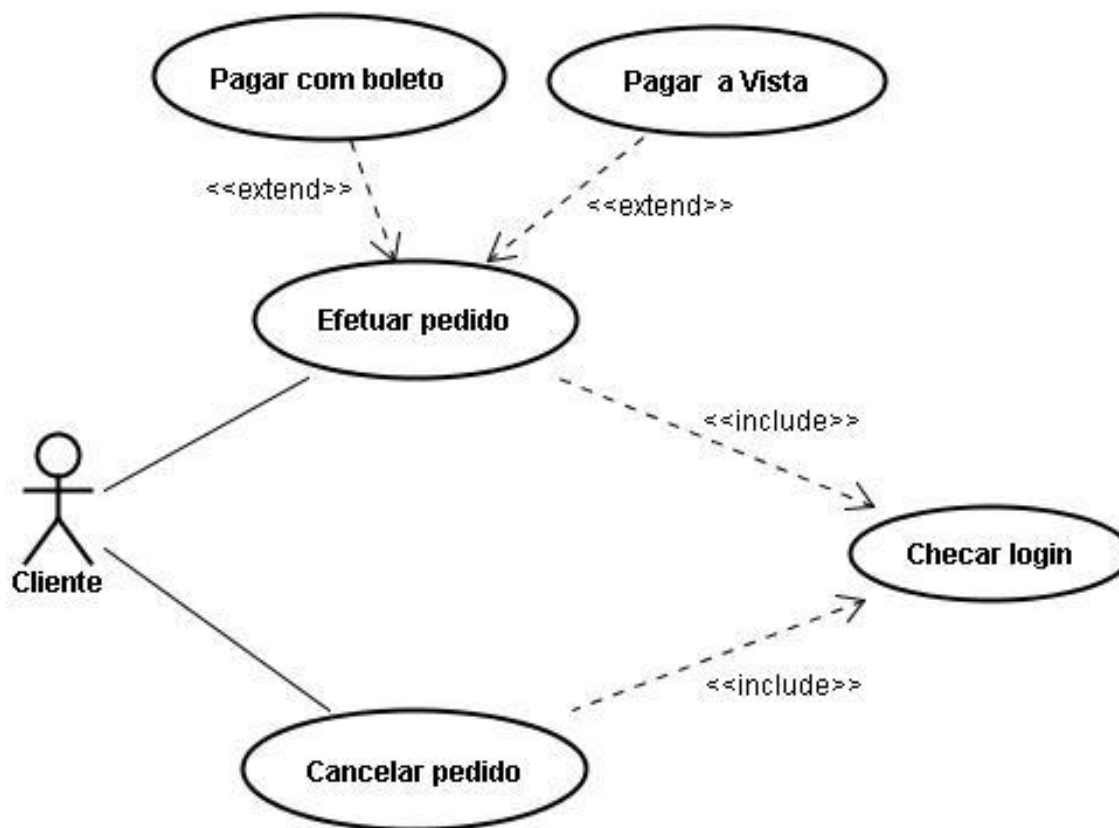
# Relações em Caso de Uso

**<<include>>**: incorpora o comportamento de um caso de uso à outro caso de uso.



# Relações em Caso de Uso

**<<extend>>**: indica que o comportamento estendido poderá ou não ser usado. O uso do comportamento estendido é opcional.





# Relações em Caso de Uso

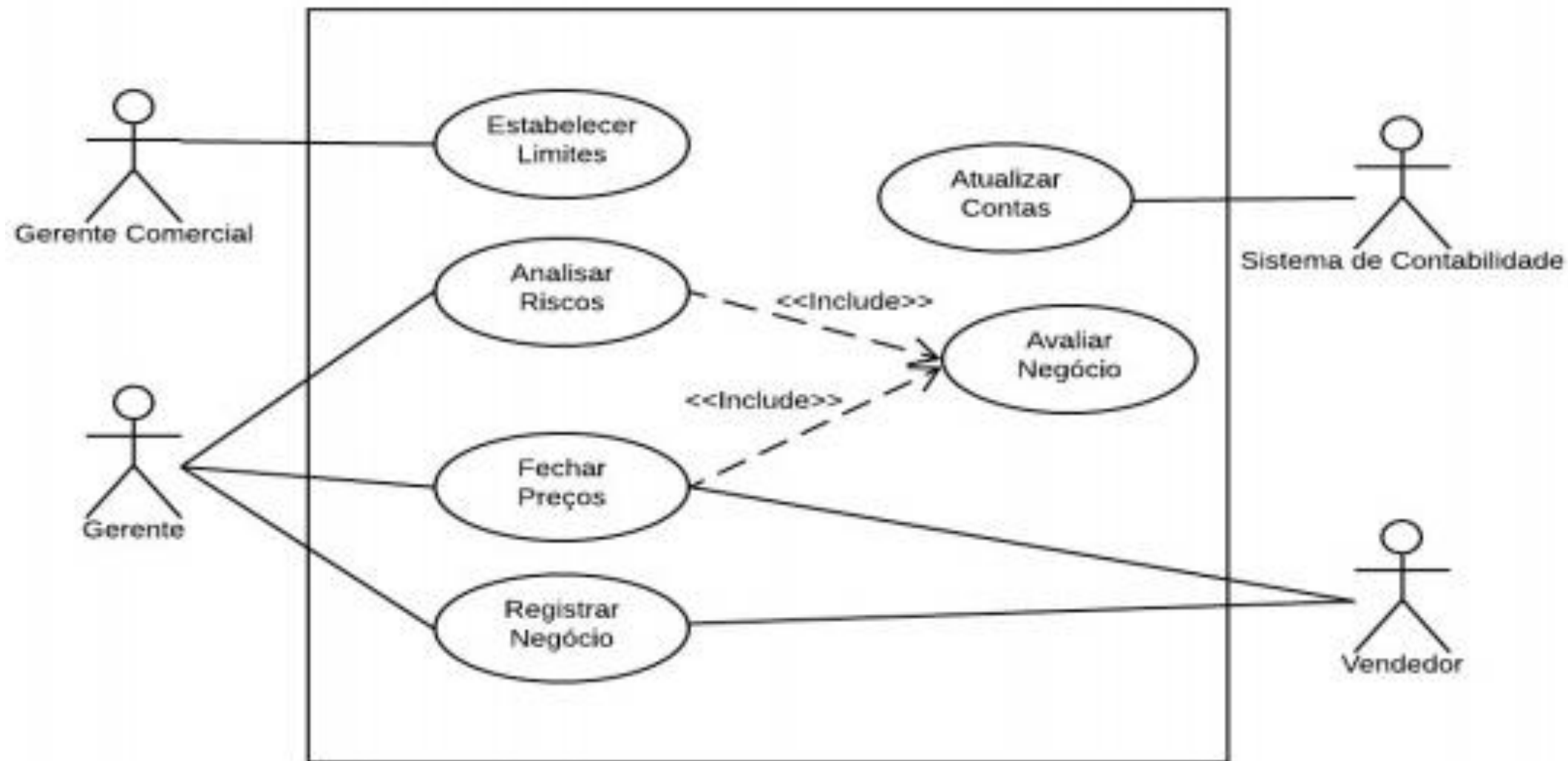
**Generalização:** utilizado para criar um caso de uso específico baseado em um caso de uso geral.



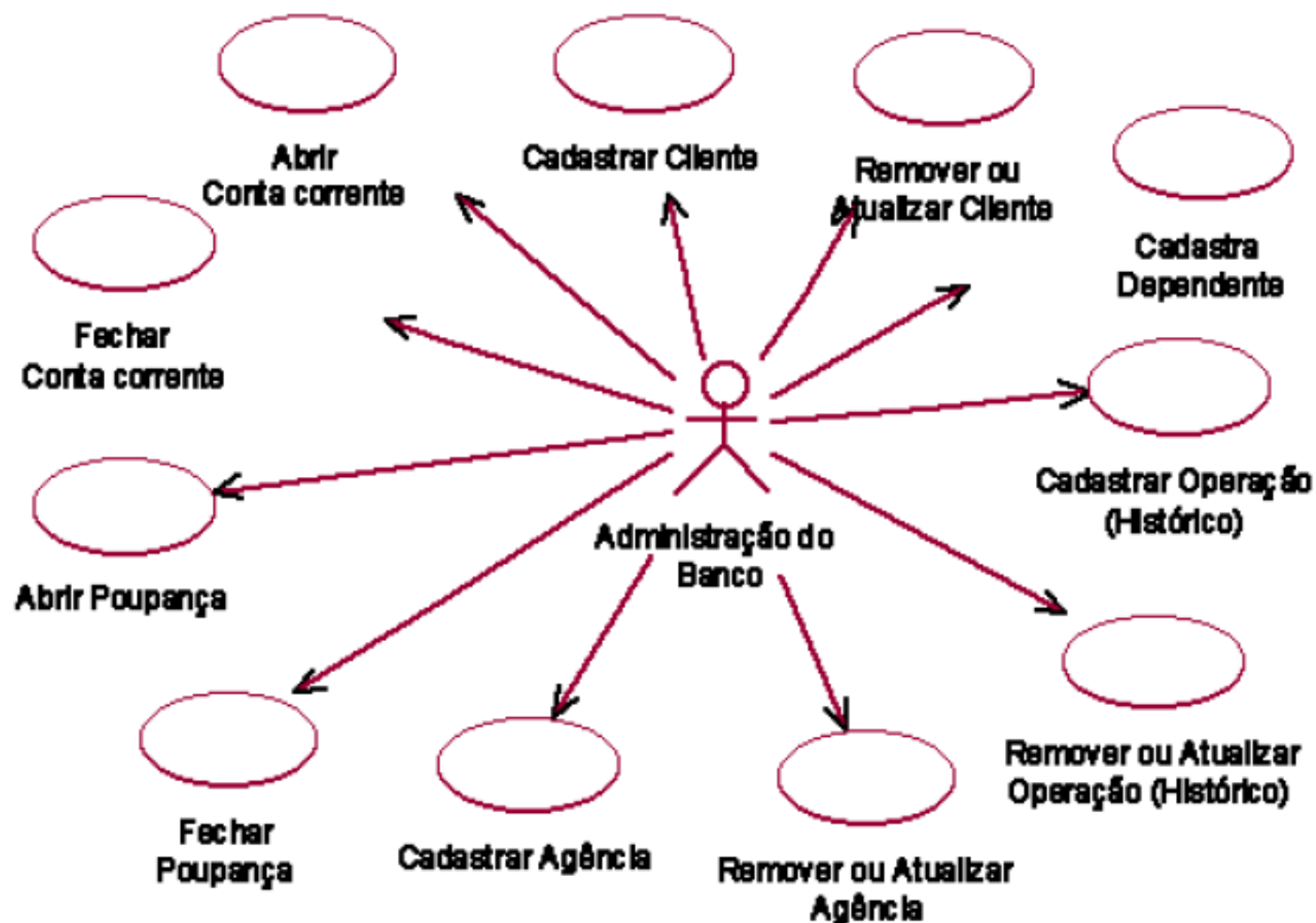
# Detalhamento do Caso de Uso

Caso de Uso: Encomendar Livro	
<b>Atores:</b> Vendedor, Cliente	
<b>Descrição:</b> Vendedor informa o título do livro desejado pelo Cliente. Sistema solicita dados do Cliente. Sistema gera pedido de encomenda do livro.	
Sequência Típica de Eventos	
Atores	Sistema
1. Cliente informa ao Vendedor o título do livro desejado.	
2. Vendedor informa título do livro ao Sistema.	3. Solicita dados do Cliente.
4. Informa dados do Cliente.	5. Abre pedido de encomenda do livro.
	6. Informa a data de previsão de chegada do livro.
	7. Encerra operação.
Sequência Alternativa de Eventos	
Não há.	

# Exemplo de Caso de Uso



# Exemplo de Caso de Uso





## Exemplo de Caso de Uso

O diagrama de use-cases anterior (com o ator **Administração do Banco**) demonstra as funções de um ator externo de um sistema de controle bancário de um banco fictício que foi modelado no estudo de caso no final deste trabalho.

O diagrama especifica que funções o administrador do banco poderá desempenhar.



## Exemplo de Caso de Uso

Pode-se perceber que não existe nenhuma preocupação com a implementação de cada uma destas funções, já que este diagrama apenas se resume a determinar que funções deverão ser suportadas pelo sistema modelado.



# Exemplo de Caso de Uso

## CONTROLE DE TAREFAS

Arnaldo deseja escrever uma aplicação de controle de tarefas para colocar em seu Smartphone.

As especificações da aplicação são as seguintes:

- O cadastro de cada tarefa contém o número da prioridade, representado por um valor real. Isso permite entrar com intervalos intermediários. Além da prioridade, o cadastro deve conter: o nome da tarefa, a data limite de execução (se houver), o percentual já concluído e o detalhamento da tarefa.



# Exemplo de Caso de Uso

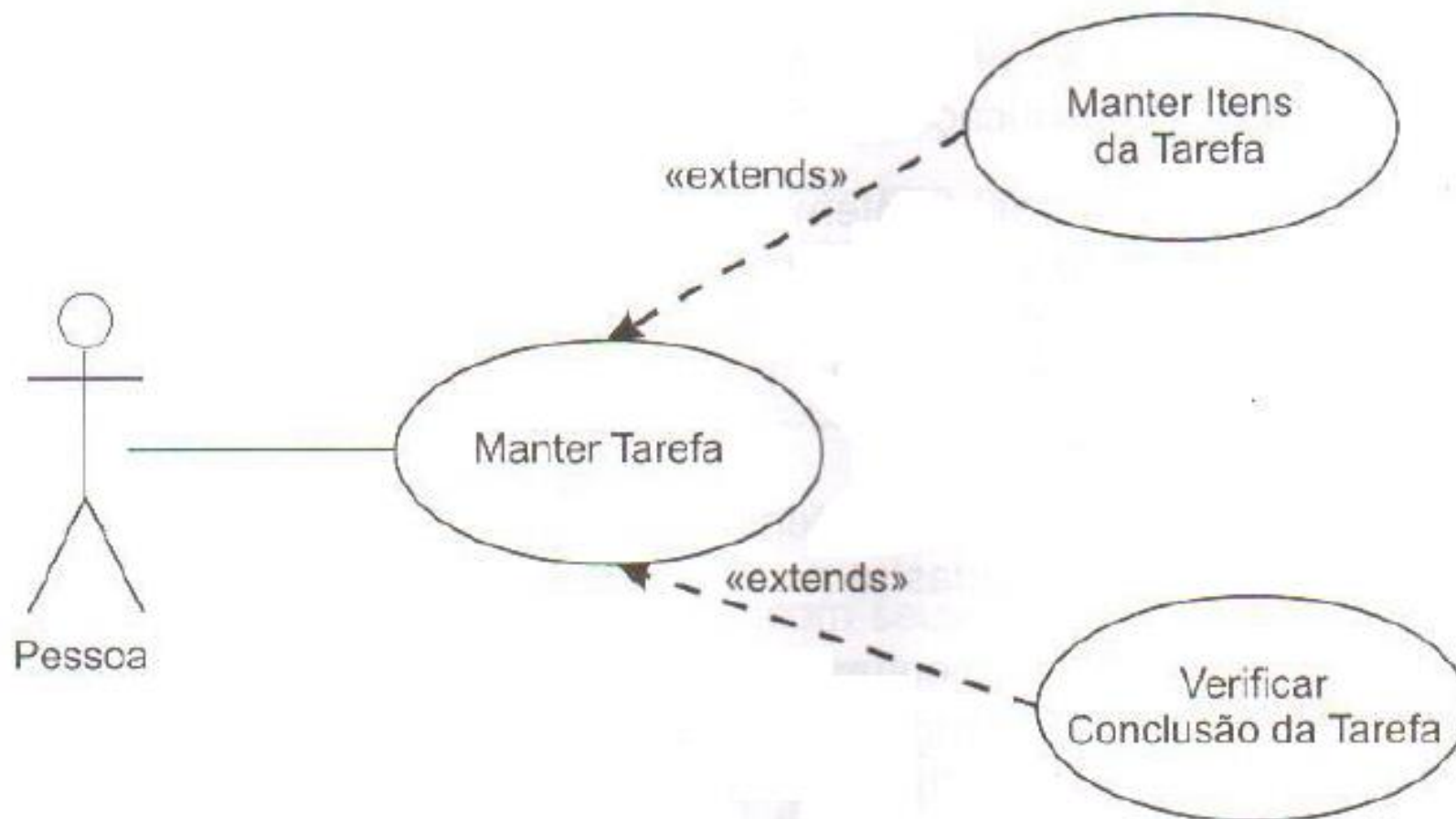
## CONTROLE DE TAREFAS

- Para cada tarefa há uma lista de itens que descrevem sua execução.
- Para cada item de execução, cadastram-se:
  - O percentual correspondente
  - A descrição da execução
  - A data da execução (quando for concluída)
  - Quando uma tarefa receber 100% de execução, esta deve ser movida automaticamente para a lista de tarefas concluídas, podendo ser apagada, se for o caso.



# Exemplo de Caso de Uso

## CONTROLE DE TAREFAS





# Exemplo de Caso de Uso

## LIGAÇÕES TELEFÔNICAS

Bruna resolveu desenvolver uma aplicação para controlar as ligações telefônicas de sua casa, a fim de checar se o valor que paga mensalmente está correto.

Assim, sempre que desejar, poderá listar as ligações efetuadas num determinado período, contabilizando o valor a pagar.



# Exemplo de Caso de Uso

## LIGAÇÕES TELEFÔNICAS

Para que isso seja possível, toda ligação será feita pelo computador. A cada solicitação de ligação, a aplicação deverá registrar: a data da ligação, a hora da ligação, quantidade de minutos gastos (que deve ser registrado no momento que a ligação for encerrada), o número de pulsos (que deve ser calculado pela aplicação) e o telefone para onde se discou.



# Exemplo de Caso de Uso

## LIGAÇÕES TELEFÔNICAS

A aplicação permitirá o controle de uma agenda de telefones, com número do telefone e o nome da pessoa de contato.

O usuário poderá escolher, no momento da ligação, se deseja um dos registros da agenda ou se digitará diretamente o número do telefone.



# Exemplo de Caso de Uso

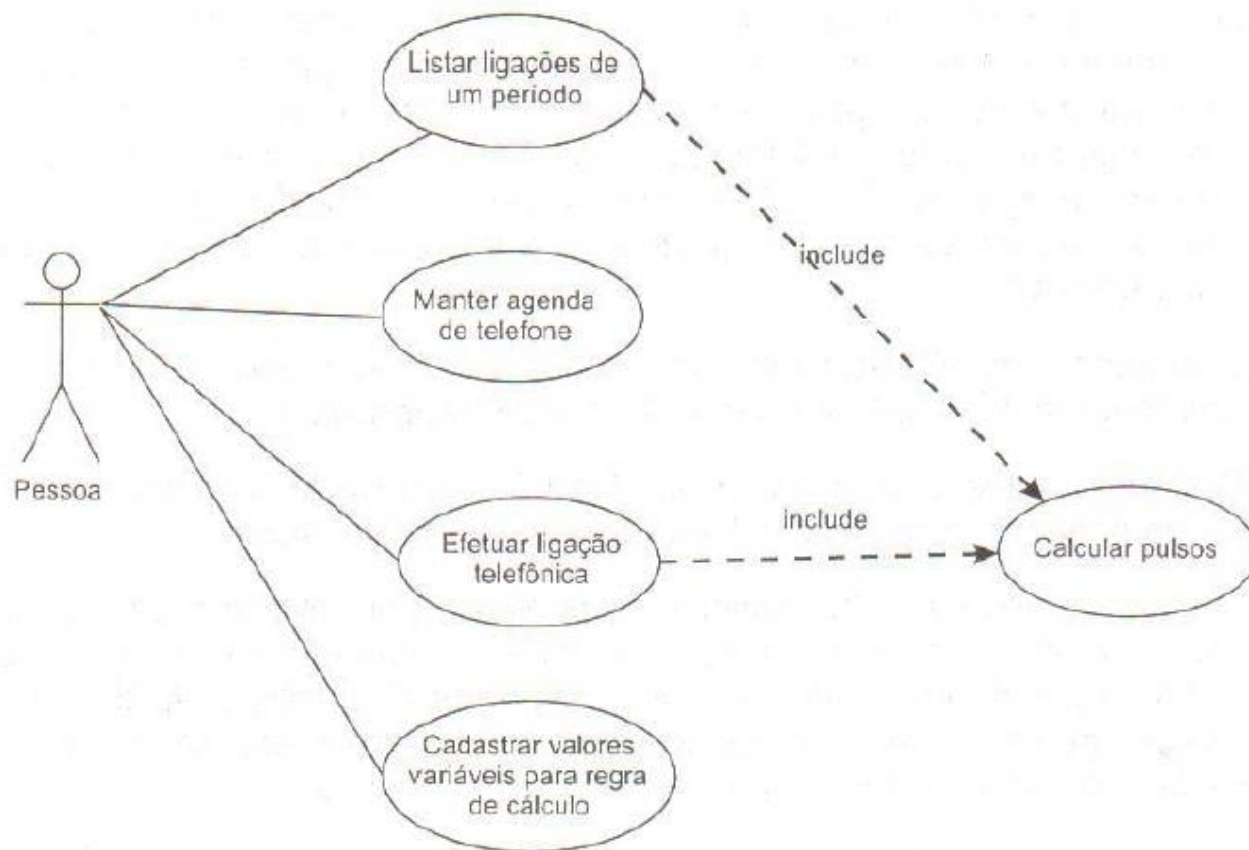
## LIGAÇÕES TELEFÔNICAS

A forma de cálculo dos pulsos considera os seguintes critérios:

- A ligação ao ser completada já conta um pulso. A partir daí, a cada 4 minutos de conversação concluída, cobra-se mais 1 pulso.
- Cada pulso custa R0.08 para ligações locais.
- Os finais de semana possuem uma promoção: cada ligação contabiliza somente 1 pulso, independente do número de minutos de conversação.

# Exemplo de Caso de Uso

## LIGAÇÕES TELEFÔNICAS





# Exemplo de Caso de Uso

## TESTES DE FIXAÇÃO

Mariana prepara diversos exercícios para suas filhas que estão na 1ª e 2ª séries.

Ela gostaria de informatizar esses exercícios, para gerar testes aleatórios.



# Exemplo de Caso de Uso

## TESTES DE FIXAÇÃO

Cada teste gerado deve ser guardado (acompanhado de suas questões), com a indicação de sua data de geração.

Na geração de um teste, é preciso informar o número de questões desejadas e qual a disciplina pertence o teste.





# Exemplo de Caso de Uso

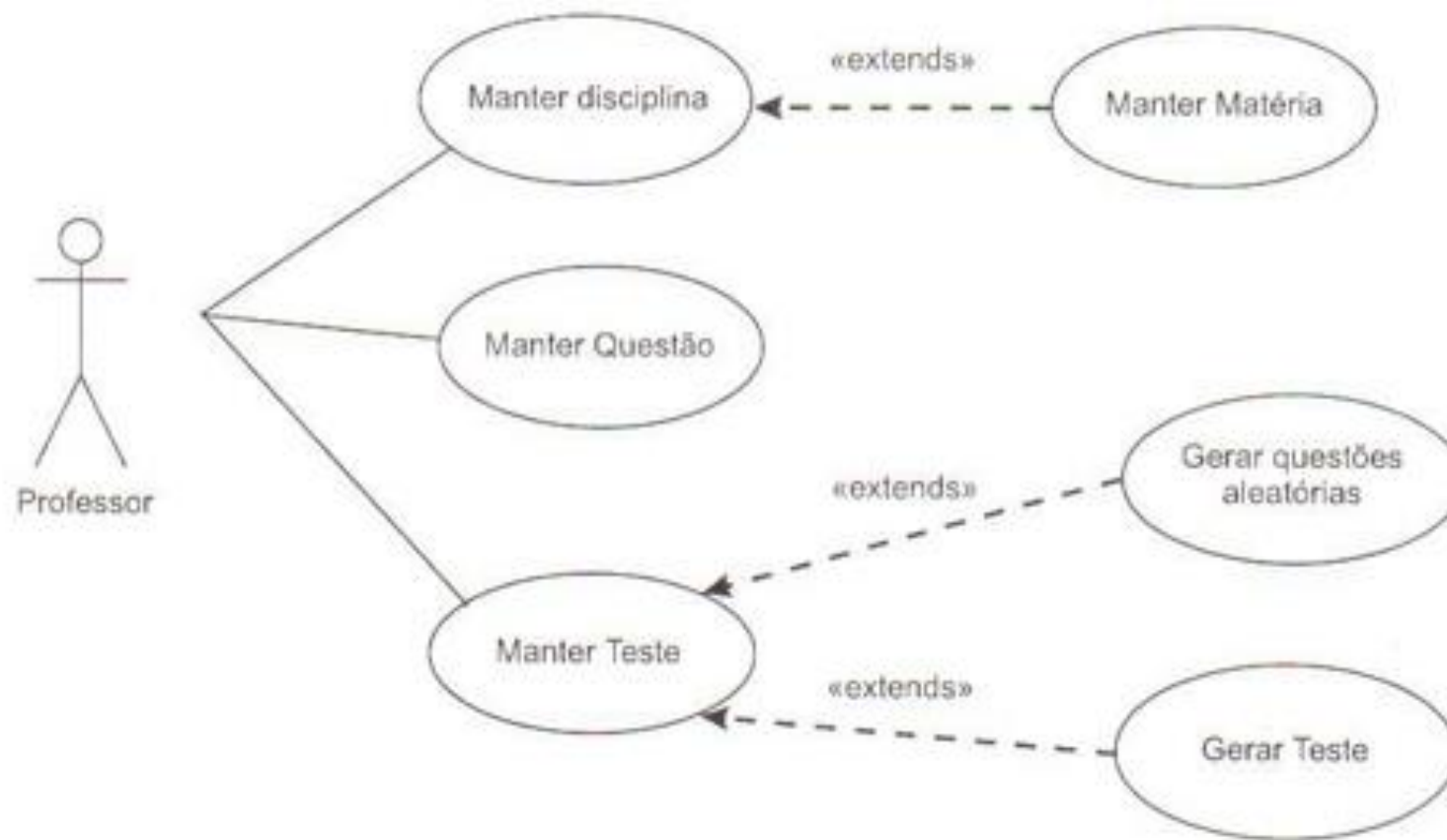
## TESTES DE FIXAÇÃO

Para cada disciplina, cadastra-se: uma lista de questões objetivas, identificando de que bimestre é cada questão e a que matéria pertence.

O gabarito também é cadastrado a fim de facilitar a correção do teste. Cada matéria faz parte de uma única disciplina. A série está ligada à matéria.

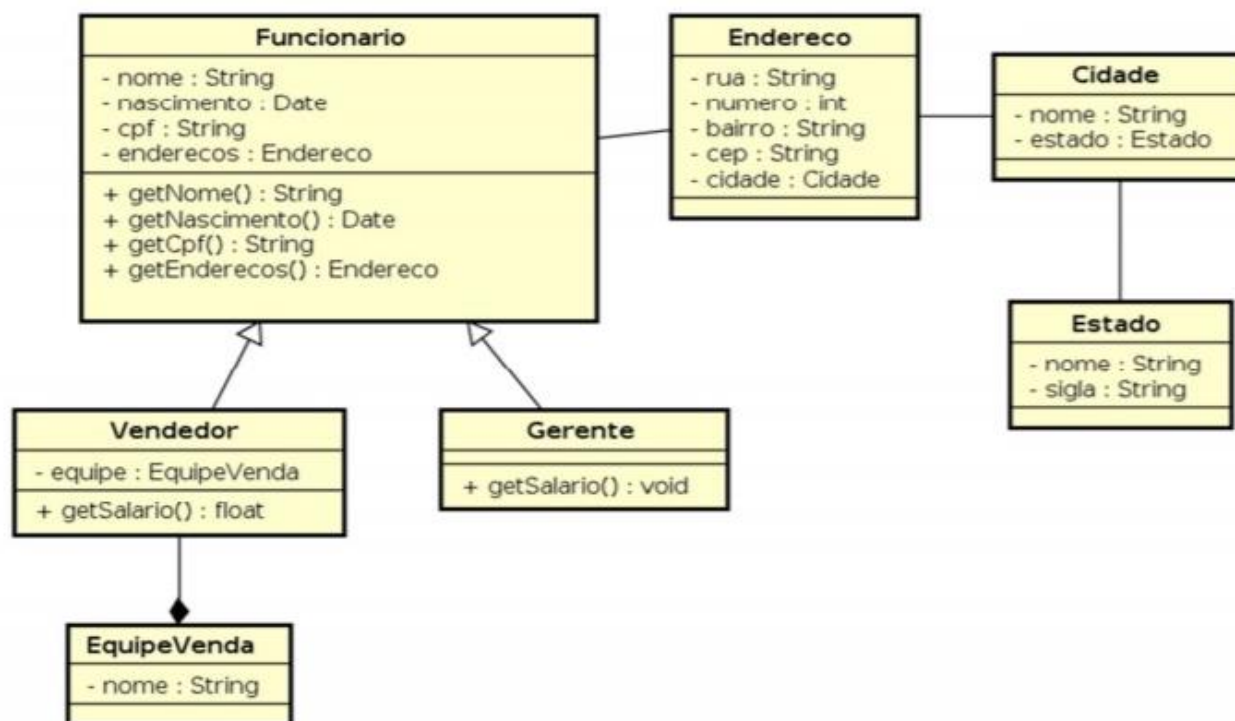
# Exemplo de Caso de Uso

## TESTES DE FIXAÇÃO



# Diagrama de Classe

O diagrama de classes demonstra a estrutura estática das classes de um sistema onde estas representam as "coisas" que são gerenciadas pela aplicação modelada.





# Diagrama de Classe

Classes podem se relacionar com outras através de diversas maneiras: **associação** (conectadas entre si), **dependência** (uma classe depende ou usa outra classe), **especialização** (uma classe é uma especialização de outra classe), ou em **pacotes** (classes agrupadas por características similares).

Todos estes relacionamentos são mostrados no diagrama de classes juntamente com as suas estruturas internas, que são os atributos e operações.



# Diagrama de Classe

O diagrama de classes é considerado estático já que a estrutura descrita é sempre válida em qualquer ponto do ciclo de vida do sistema.

Um sistema normalmente possui alguns diagramas de classes, já que não são todas as classes que estão inseridas em um único diagrama e uma certa classes pode participar de vários diagramas de classes



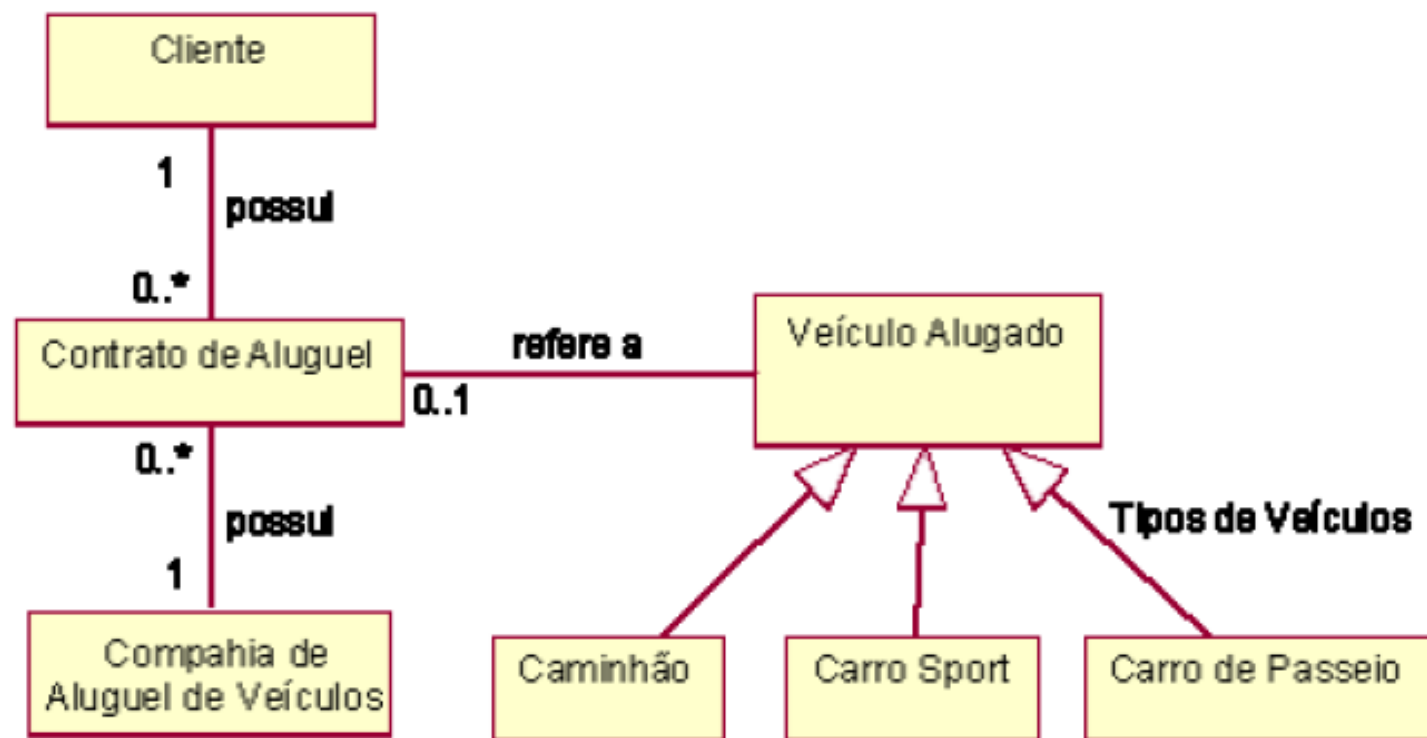
# Diagrama de Classe

Uma classe num diagrama pode ser diretamente implementada utilizando-se uma linguagem de programação orientada a objetos que tenha suporte direto para construção de classes.

Para criar um diagrama de classes, as classes têm que estar identificadas, descritas e relacionadas entre si.

# Diagrama de Classe

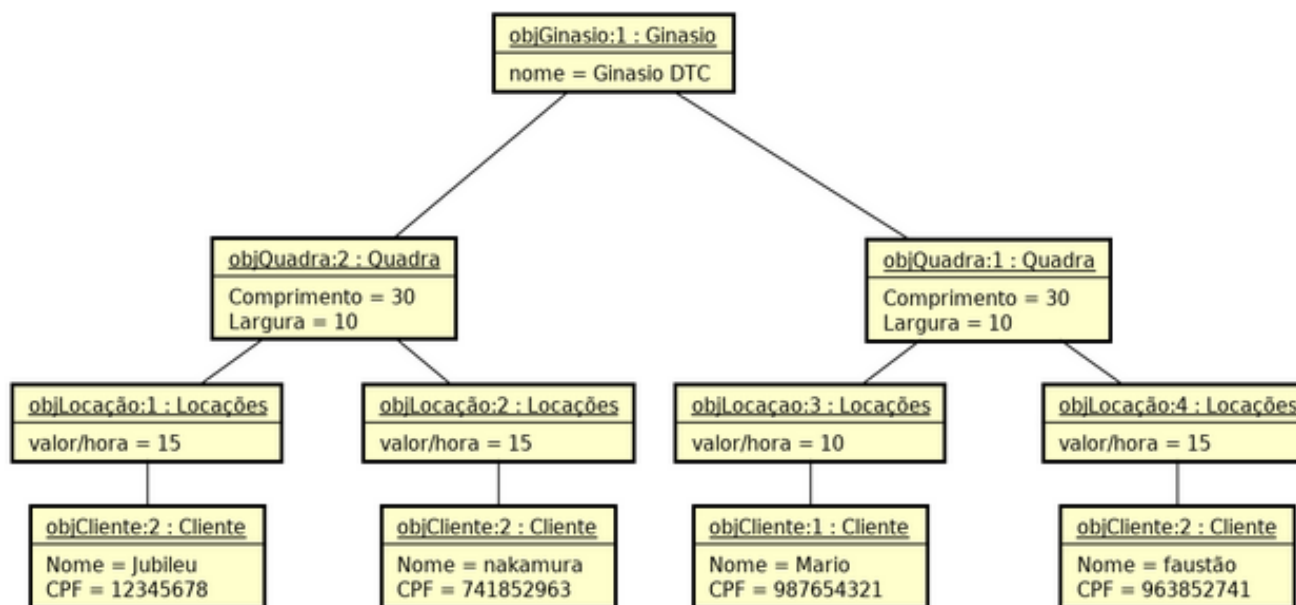
Exemplo:



# Diagrama de Objetos

O diagrama de objetos é uma variação do diagrama de classes e utiliza quase a mesma notação.

A diferença é que o diagrama de objetos mostra os objetos que foram instanciados das classes.







# Diagrama de Objetos

O diagrama de objetos é como se fosse o perfil do sistema em um certo momento de sua execução.

A mesma notação do diagrama de classes é utilizada com 2 exceções: os objetos são escritos com seus nomes sublinhados e todas as instâncias num relacionamento são mostradas.

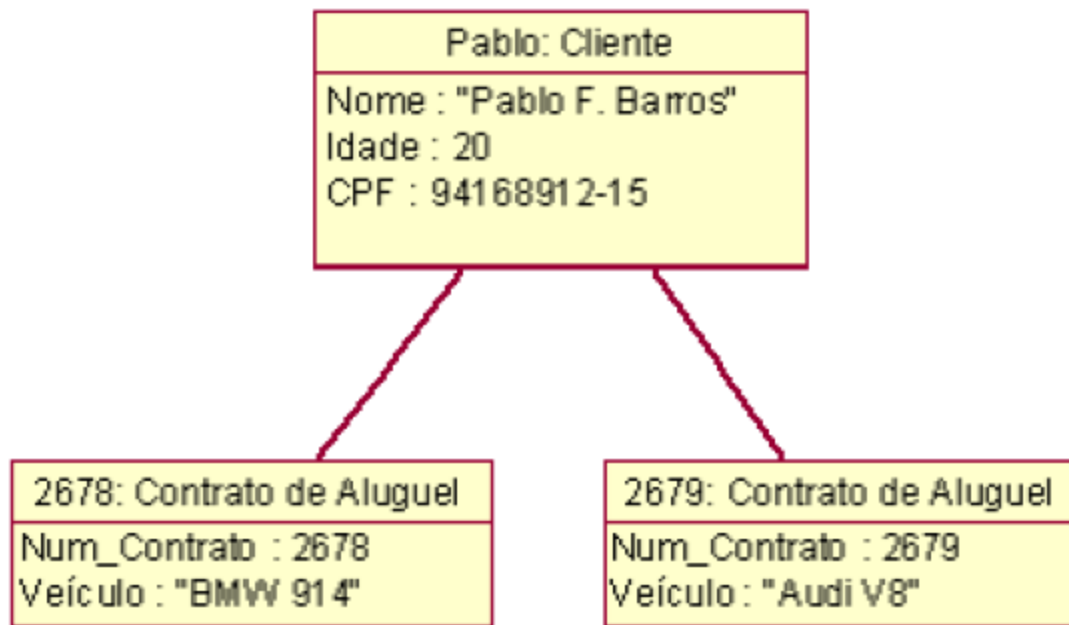


# Diagrama de Objetos

Os diagramas de objetos não são tão importantes como os diagramas de classes, mas eles são muito úteis para exemplificar diagramas complexos de classes ajudando muito em sua compreensão.

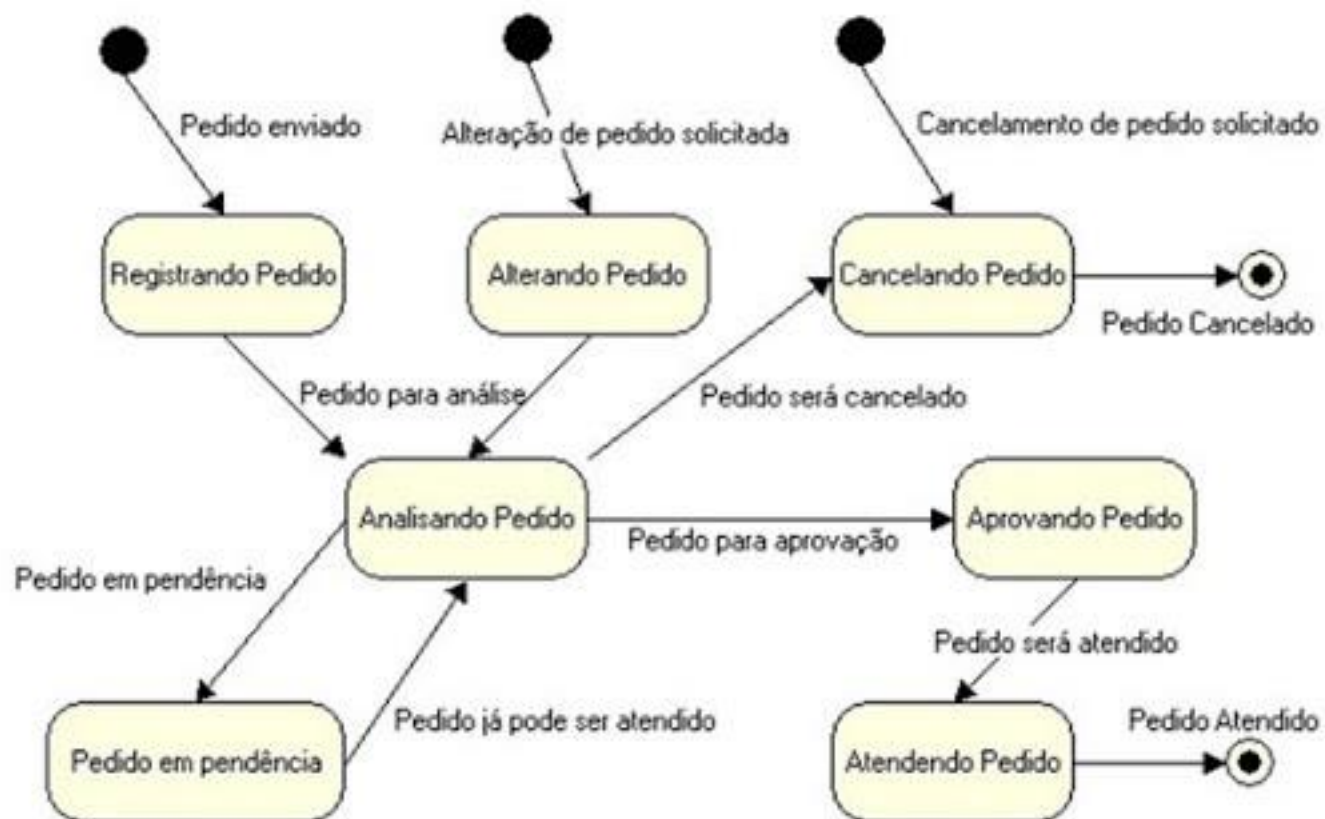
# Diagrama de Objetos

Os Diagramas de objetos também são usados como parte dos diagramas de colaboração, onde a colaboração dinâmica entre os objetos do sistema são mostrados.



# Diagrama de Estado

O diagrama de estado é tipicamente um complemento para a descrição das classes.






# Diagrama de Estado

Este diagrama mostra todos os estados possíveis que objetos de uma certa classe podem se encontrar e mostra também quais são os eventos do sistemas que provocam tais mudanças.

Os diagramas de estado não são escritos para todas as classes de um sistema, mas apenas para aquelas que possuem um número definido de estados conhecidos e onde o comportamento das classes é afetado e modificado pelos diferentes estados.





# Diagrama de Estado

Diagramas de estado capturam o ciclo de vida dos objetos, subsistemas e sistemas.

Eles mostram os estados que um objeto pode possuir e como os eventos (mensagens recebidas, timer, erros, e condições sendo satisfeitas) afetam estes estados ao passar do tempo.



# Diagrama de Estado

Diagramas de estado possuem um ponto de início e vários pontos de finalização.

Um ponto de início (estado inicial) é mostrado como um círculo todo preenchido, e um ponto de finalização (estado final) é mostrado como um círculo em volta de um outro círculo menor preenchido.



# Diagrama de Estado

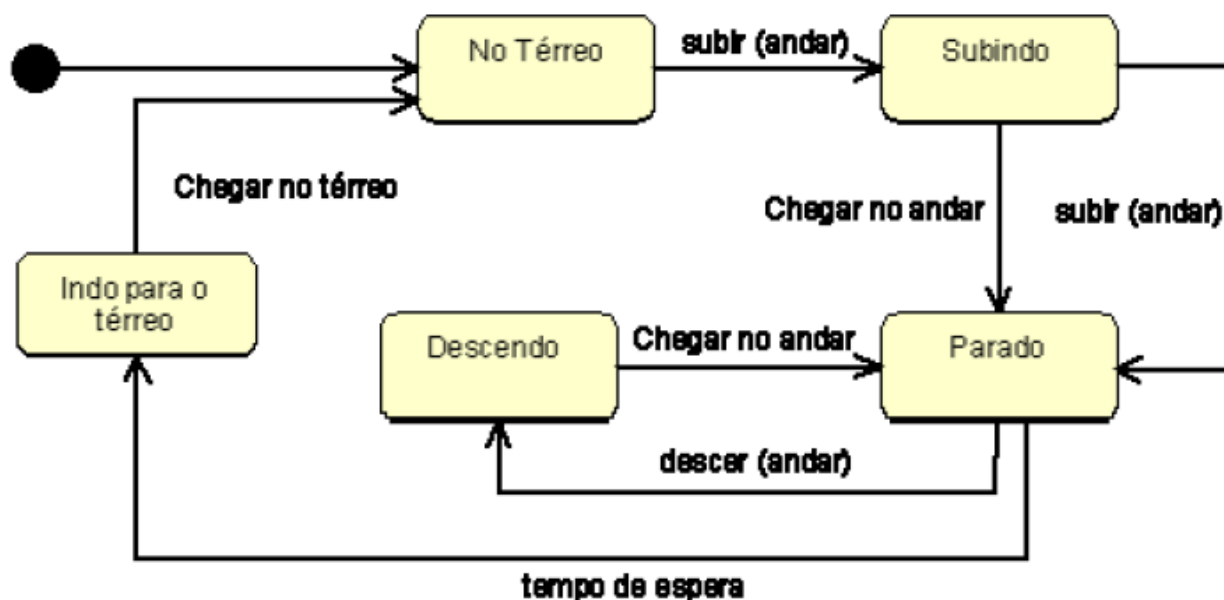
Um estado é mostrado como um retângulo com cantos arredondados.

Entre os estados estão as transições, mostrados como uma linha com uma seta no final de um dos estados.



# Diagrama de Estado

A transição pode ser nomeada com o seu evento causador.  
Quando o evento acontece, a transição de um estado para outro é executada ou disparada.





# Diagrama de Estado

Uma transição de estado normalmente possui um evento ligado a ela.

Se um evento é anexado a uma transição, esta será executada quando o evento ocorrer.



# Diagrama de Estado

Se uma transição não possuir um evento ligado a ela, a mesma ocorrerá quando a ação interna do código do estado for executada (se existir ações internas como entrar, sair, fazer ou outras ações definidas pelo desenvolvedor).

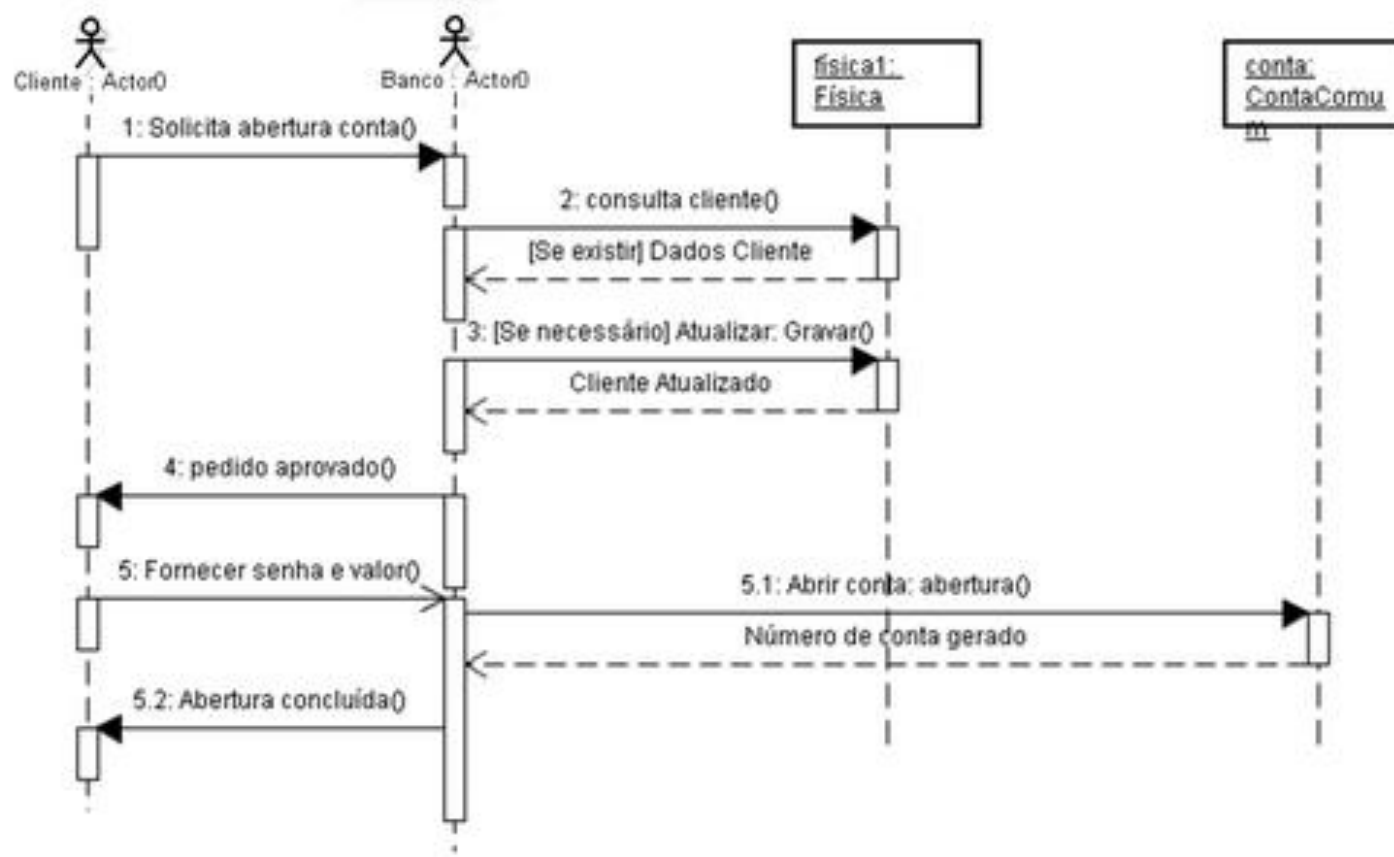


# Diagrama de Estado

Então quando todas as ações forem executadas pelo estado, a transição será disparada e serão iniciadas as atividades do próximo estado no diagrama de estados.

# Diagrama de Sequência

Um diagrama de sequência mostra a colaboração dinâmica entre os vários objetos de um sistema.





# Diagrama de Sequência

O mais importante aspecto deste diagrama é que a partir dele percebe-se a sequência de mensagens enviadas entre os objetos.

Ele mostra a interação entre os objetos, alguma coisa que acontecerá em um ponto específico da execução do sistema.



# Diagrama de Sequência

O diagrama de sequência consiste em um número de objetos mostrado em linhas verticais. O decorrer do tempo é visualizado observando-se o diagrama no sentido vertical de cima para baixo.

As mensagens enviadas por cada objeto são simbolizadas por setas entre os objetos que se relacionam.



# Diagrama de Sequência

Diagramas de sequência possuem dois eixos: o eixo vertical, que mostra o tempo e o eixo horizontal, que mostra os objetos envolvidos na sequência de uma certa atividade.

Eles também mostram as interações para um cenário específico de uma certa atividade do sistema.





# Diagrama de Sequência

No eixo horizontal estão os objetos envolvidos na sequência.

Cada um é representado por um retângulo de objeto (similar ao diagrama de objetos) e uma linha vertical pontilhada chamada de linha de vida do objeto, indicando a execução do objeto durante a sequência, como exemplo citamos: mensagens recebidas ou enviadas e ativação de objetos.



# Diagrama de Sequência

A comunicação entre os objetos é representada como linha com setas horizontais simbolizando as mensagens entre as linhas de vida dos objetos.

A seta especifica se a mensagem é síncrona, assíncrona ou simples. As mensagens podem possuir também números sequenciais, eles são utilizados para tornar mais explícito as sequência no diagrama.



# Diagrama de Sequência

Em alguns sistemas, objetos rodam concorrentemente, cada um com sua linha de execução (thread).

Se o sistema usa linhas concorrentes de controle, isto é mostrado como ativação, mensagens assíncronas, ou objetos assíncronos.

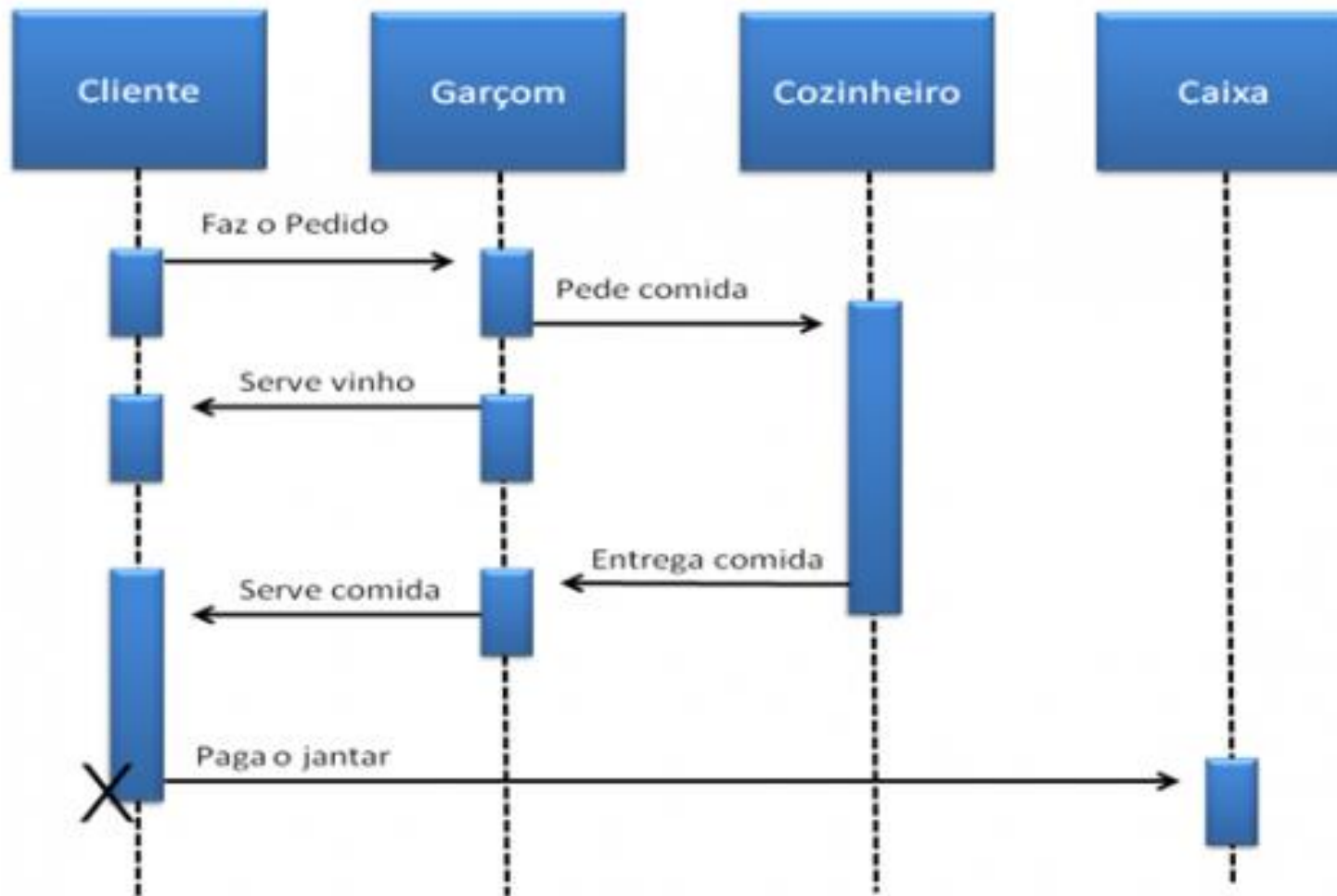


# Diagrama de Sequência

Os diagramas de sequência podem mostrar objetos que são criados ou destruídos como parte do cenário documentado pelo diagrama.

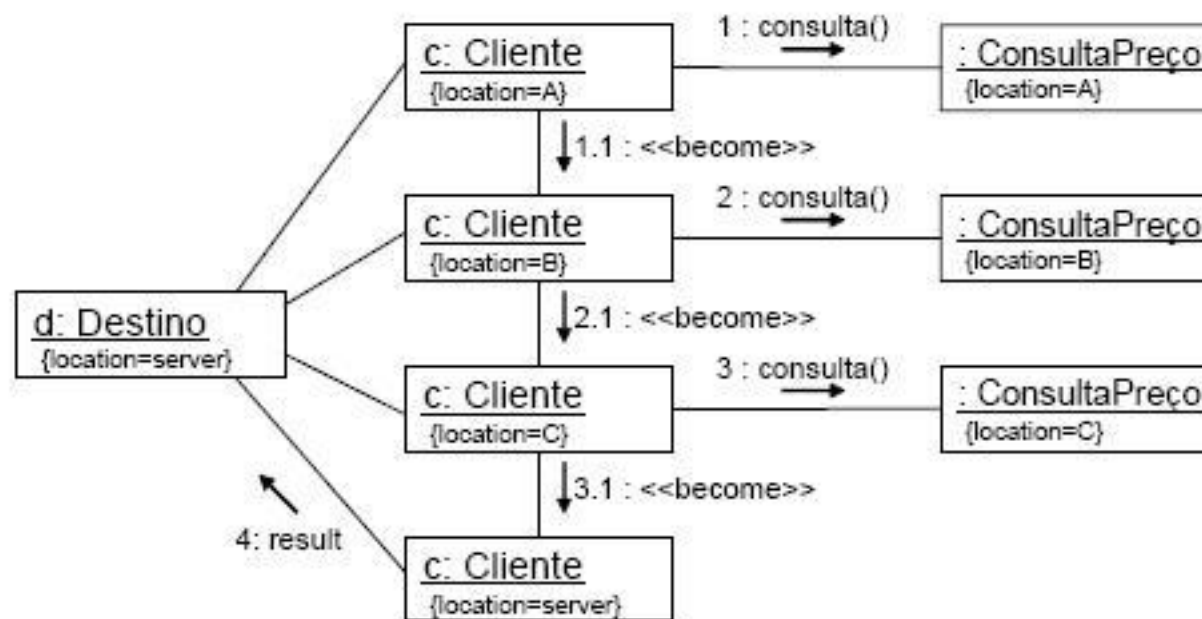
Um objeto pode criar outros objetos através de mensagens. A mensagem que cria ou destrói um objeto é geralmente síncrona, representada por uma seta sólida.

# Diagrama de Sequência



# Diagrama de Colaboração

Um diagrama de colaboração mostra de maneira semelhante ao diagrama de sequência, a colaboração dinâmica entre os objetos.





# Diagrama de Colaboração

Normalmente pode-se escolher entre utilizar o diagrama de colaboração ou o diagrama de sequência.

No diagrama de colaboração, além de mostrar a troca de mensagens entre os objetos, percebe-se também os objetos com os seus relacionamentos.



# Diagrama de Colaboração

A interação de mensagens é mostrada em ambos os diagramas.

Se a ênfase do diagrama for o decorrer do tempo, é melhor escolher o diagrama de sequência, mas se a ênfase for o contexto do sistema, é melhor dar prioridade ao diagrama de colaboração.





# Diagrama de Colaboração

O diagrama de colaboração é desenhado como um diagrama de objeto, onde os diversos objetos são mostrados juntamente com seus relacionamentos.

As setas de mensagens são desenhadas entre os objetos para mostrar o fluxo de mensagens entre eles.

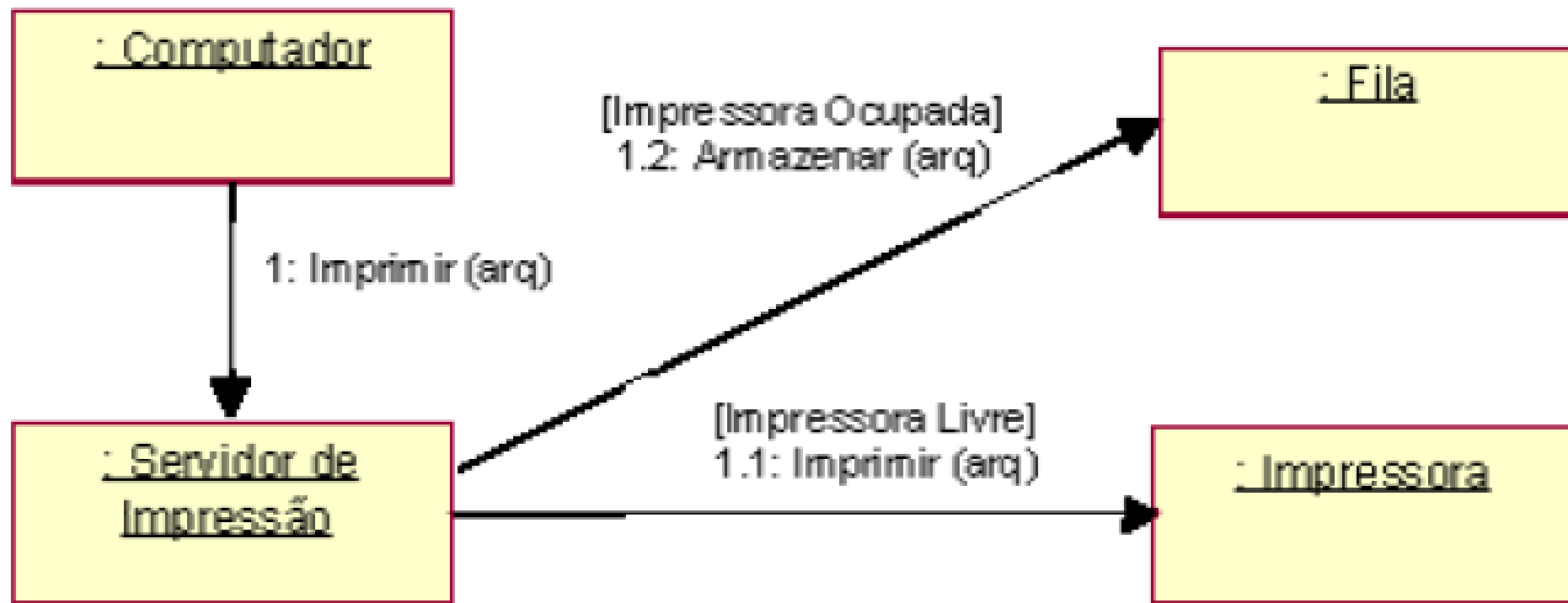


# Diagrama de Colaboração

As mensagens são nomeadas, que entre outras coisas mostram a ordem em que as mensagens são enviadas. Também podem mostrar condições, interações, valores de resposta, e etc.

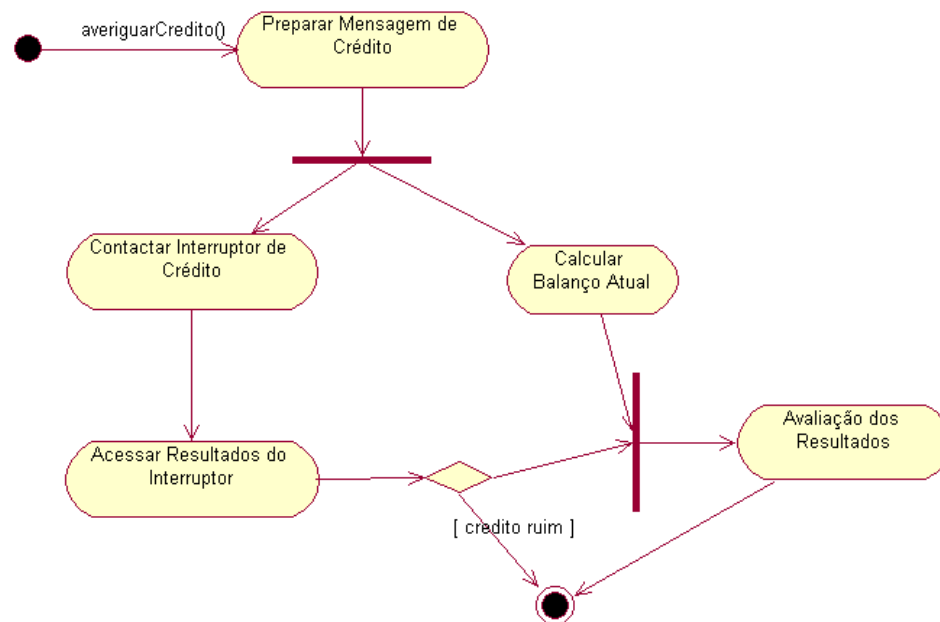
O diagrama de colaboração também pode conter objetos ativos, que executam paralelamente com outros.

# Diagrama de Colaboração



# Diagrama de Atividade

Diagramas de atividade capturam ações e seus resultados. Eles focam o trabalho executado na implementação de uma operação (método), e suas atividades numa instância de um objeto.



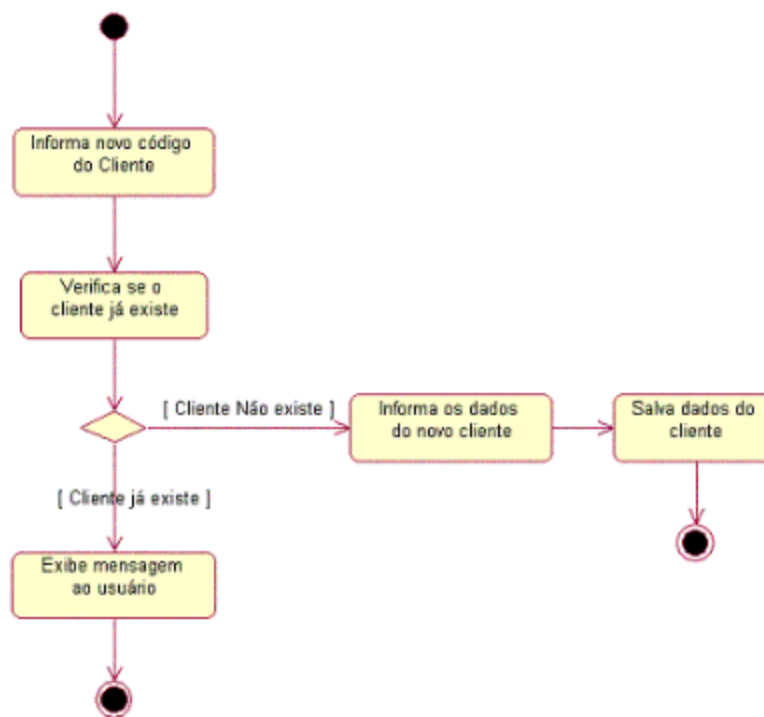


# Diagrama de Atividade

O diagrama de atividade é uma variação do diagrama de estado e possui um propósito um pouco diferente do diagrama de estado, que é o de capturar ações (trabalho e atividades que serão executados) e seus resultados em termos das mudanças de estados dos objetos.

# Diagrama de Atividade

Os estados no diagrama de atividade mudam para um próximo estágio quando uma ação é executada (sem ser necessário especificar nenhum evento como no diagrama de estado).





# Diagrama de Atividade

Outra diferença entre o diagrama de atividade e o de estado é que podem ser colocadas como "swimlanes".

Uma swimlane agrupa atividades, com respeito a quem é responsável e onde estas atividades residem na organização, e é representada por retângulos que englobam todos os objetos que estão ligados a ela (swimlane).

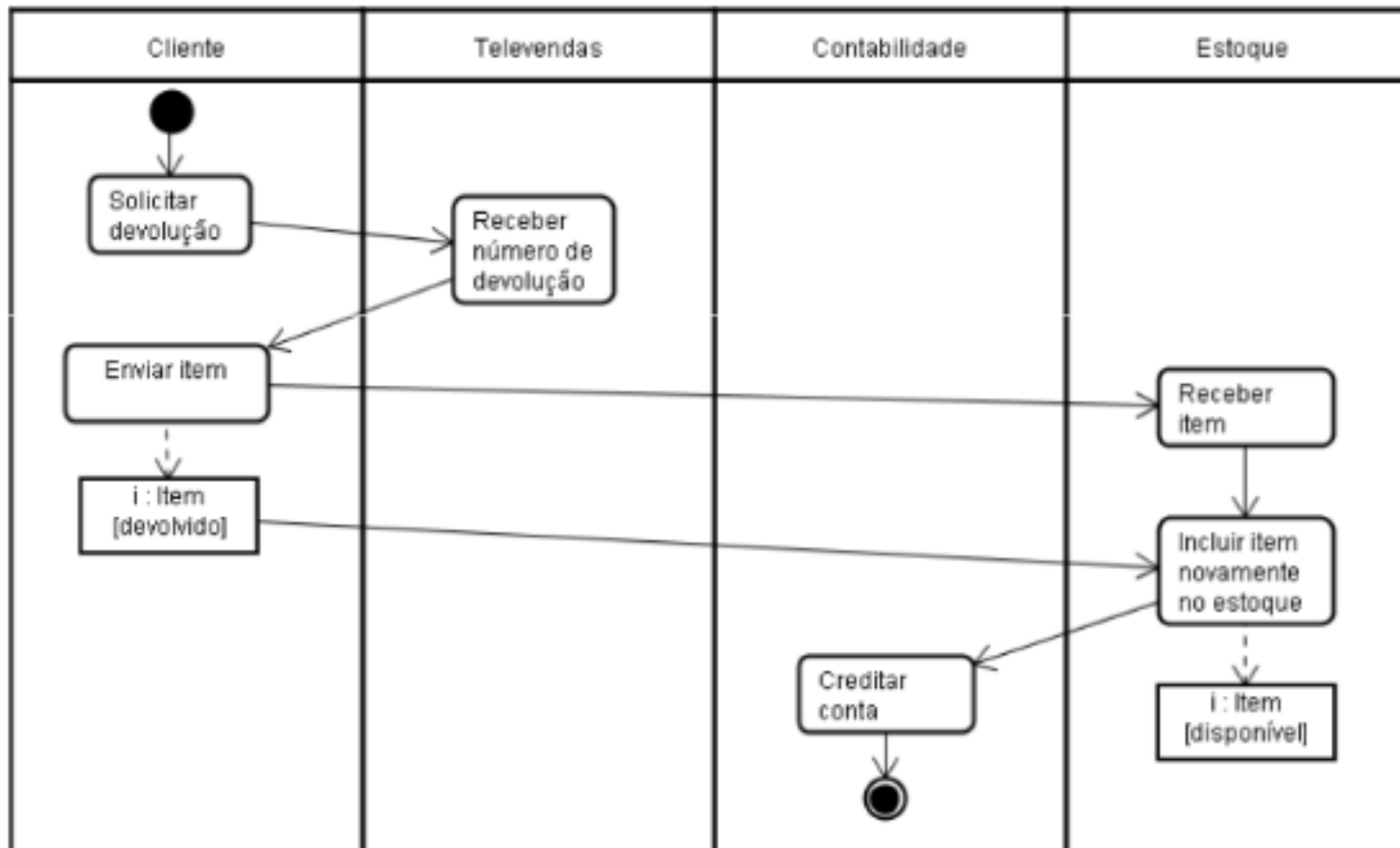


# Diagrama de Atividade

Um diagrama de atividade é uma maneira alternativa de se mostrar interações, com a possibilidade de expressar como as ações são executadas, o que elas fazem (mudanças dos estados dos objetos), quando elas são executadas (sequência das ações), e onde elas acontecem (swimlanes).



# Diagrama de Atividade





# Diagrama de Atividade

Um diagrama de atividade pode ser usado com diferentes propósitos inclusive:

1. Para capturar os trabalhos que serão executados quando uma operação é disparada (ações).

Este é o uso mais comum para o diagrama de atividade.

2. Para capturar o trabalho interno em um objeto.



# Diagrama de Atividade

3. Para mostrar como um grupo de ações relacionadas podem ser executadas, e como elas vão afetar os objetos em torno delas.
4. Para mostrar como uma instância pode ser executada em termos de ações e objetos.



# Diagrama de Atividade

5. Para mostrar como um negócio funciona em termos de trabalhadores (atores), fluxos de trabalho, organização, e objetos (fatores físicos e intelectuais usados no negócio).



# Diagrama de Atividade

O diagrama de atividade mostra o fluxo sequencial das atividades, é normalmente utilizado para demonstrar as atividades executadas por uma operação específica do sistema.

Consistem em estados de ação, que contém a especificação de uma atividade a ser desempenhada por uma operação do sistema.

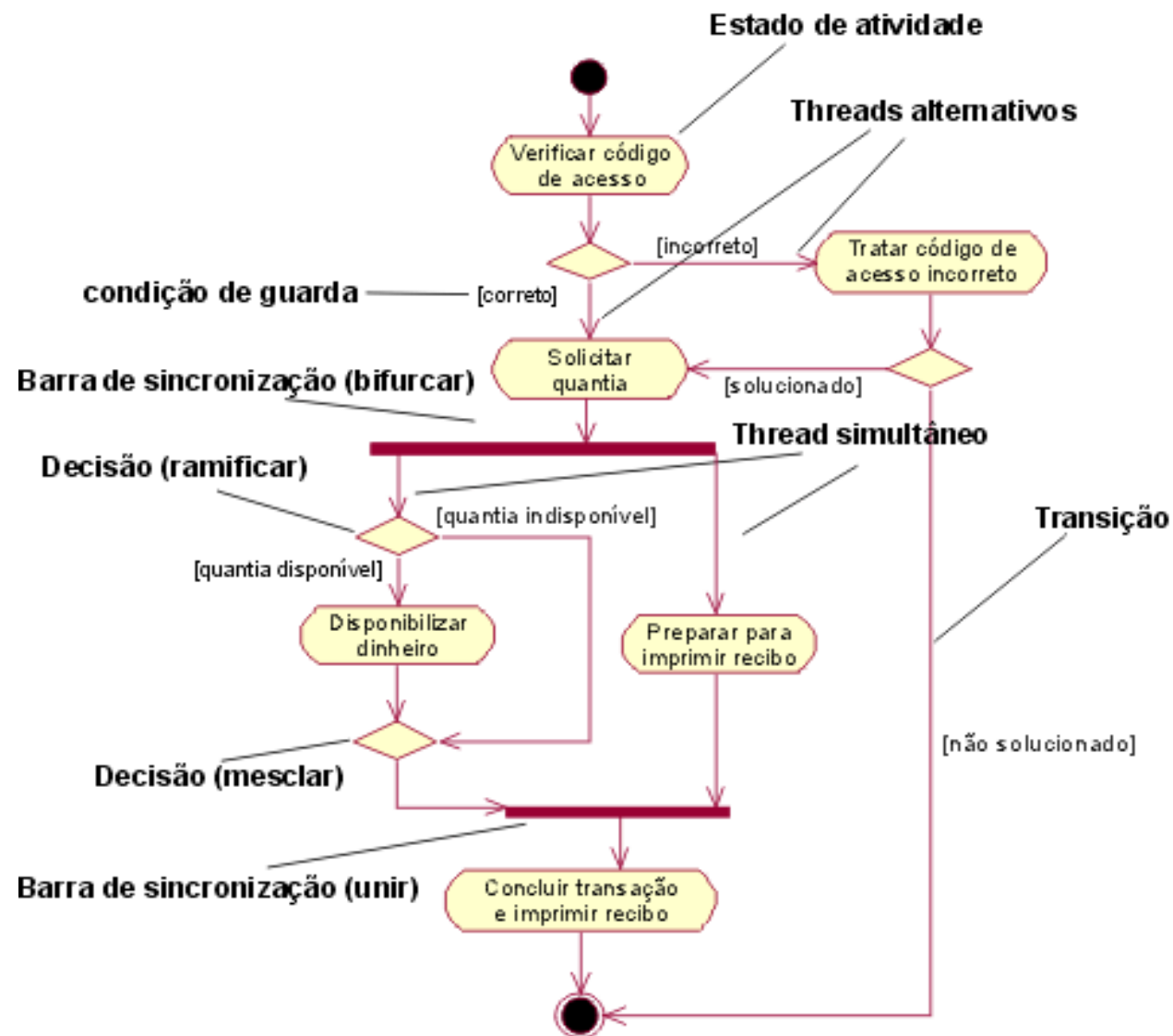


# Diagrama de Atividade

Decisões e condições, como execução paralela, também podem ser mostrados no diagrama de atividade.

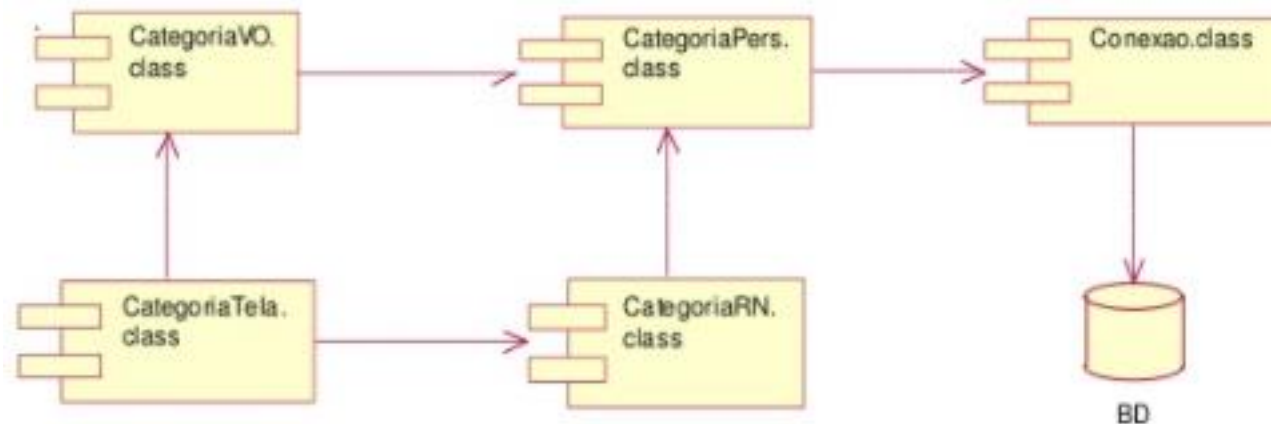
O diagrama também pode conter especificações de mensagens enviadas e recebidas como partes de ações executadas.

# Diagrama de Atividade



# Diagrama de Componente

O diagrama de componente e o de execução são diagramas que mostram o sistema por um lado funcional, expondo as relações entre seus componentes e a organização de seus módulos durante sua execução.







# Diagrama de Componente

O diagrama de componente descreve os componentes de software e suas dependências entre si, representando a estrutura do código gerado.

Os componentes são a implementação na arquitetura física dos conceitos e da funcionalidade definidos na arquitetura lógica (classes, objetos e seus relacionamentos).



# Diagrama de Componente

Eles são tipicamente os arquivos implementados no ambiente de desenvolvimento.

Um componente é mostrado em UML como um retângulo com uma elipse e dois retângulos menores do seu lado esquerdo. O nome do componente é escrito abaixo ou dentro de seu símbolo.




# Diagrama de Componente

Componentes são tipos, mas apenas componentes executáveis podem ter instâncias.

Um diagrama de componente mostra apenas componentes como tipos.

Para mostrar instâncias de componentes, deve ser usado um diagrama de execução, onde as instâncias executáveis são alocadas em nodes.





# Diagrama de Componente

A dependência entre componentes pode ser mostrada como uma linha tracejada com uma seta, simbolizando que um componente precisa do outro para possuir uma definição completa.

Com o diagrama de componentes é facilmente visível detectar que arquivos **.dll** são necessários para executar a aplicação.



# Diagrama de Componente

Componentes podem definir interfaces que são visíveis para outros componentes.

As interfaces podem ser tanto definidas ao nível de codificação (como em Java) quanto em interfaces binárias usadas em run-time (como em OLE).



# Diagrama de Componente

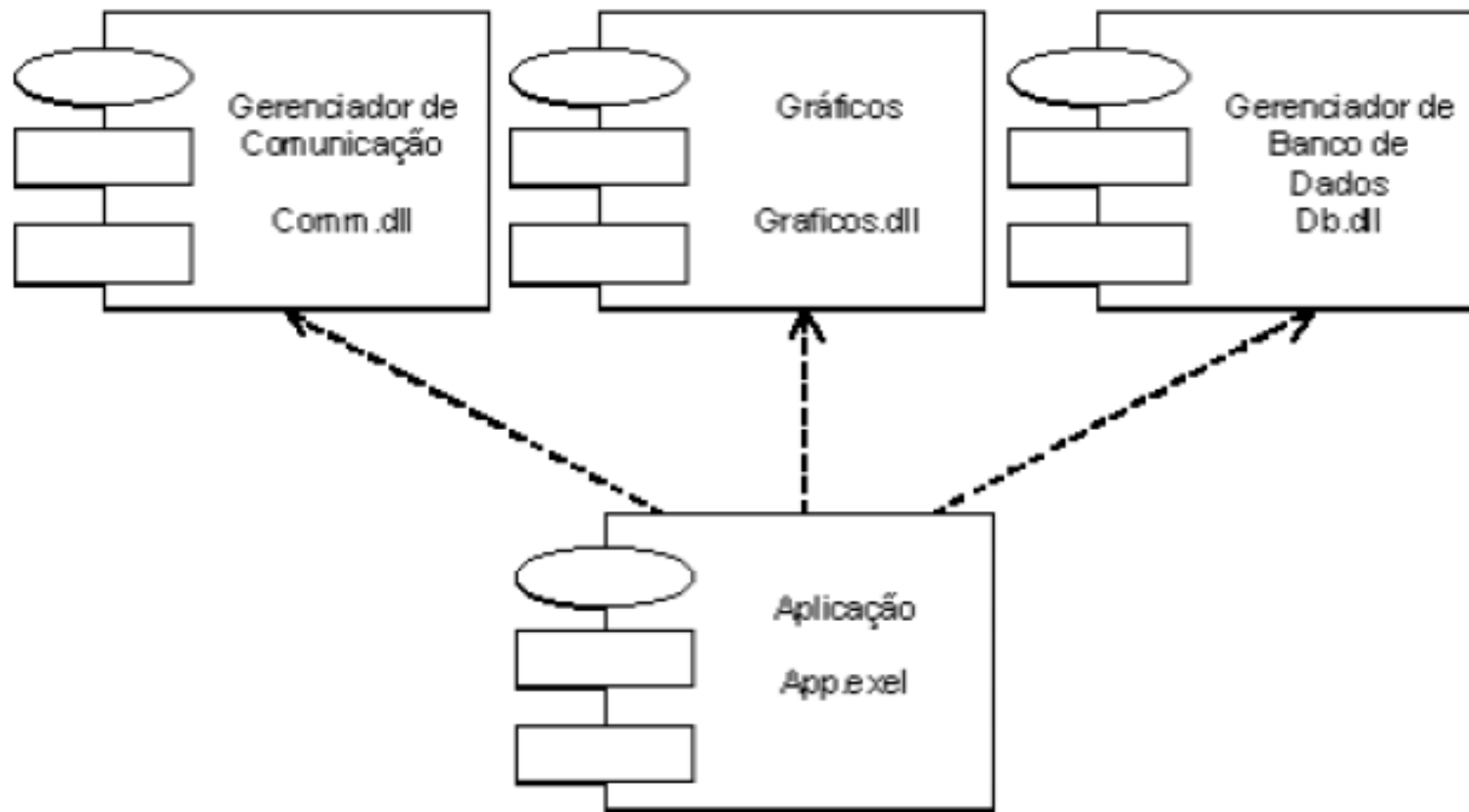
Uma interface é mostrada como uma linha partindo do componente e com um círculo na outra extremidade.

O nome é colocado junto do círculo no final da linha.

Dependências entre componentes podem então apontar para a interface do componente que está sendo usada.



# Diagrama de Componente



# Conclusão

Definir as técnicas de análise e os padrões é uma tarefa árdua, e a maioria das organizações não compreende a análise de negócios como suficiente para apreciar com flexibilidade as habilidades que um profissional de análise de negócios precisa ter.







# Conclusão

Os projetos variam muito e as análises diferem em perspectiva, quantidade e nível de detalhe.

Exigir um padrão de notação particular ou uma modelagem pode parecer uma boa maneira de introduzir consistência na organização.



# Conclusão

Em vez de organizações que empregam centenas de pessoas para realizar análises de negócios, as organizações devem empregar dezenas de excelentes profissionais de análise de negócios que tenham domínio de notações de modelagem diferentes, que tenham domínio da arte de trabalhar com as partes interessadas, que tenham domínio da arte da comunicação em todos os níveis da organização, que tenham a capacidade e a habilidade de usar diferentes estruturas e que usem ferramentas para organizar e representar os componentes da organização com precisão.

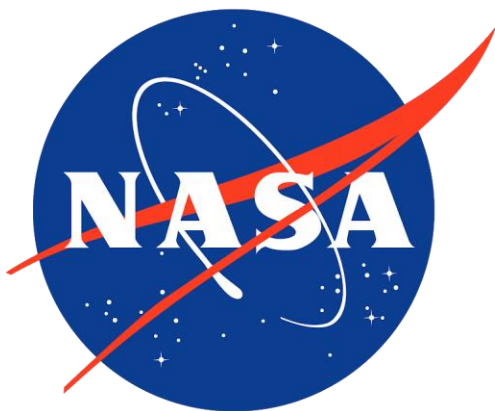


# Conclusão

Com base nessas ações, as organizações tendem a receber muito bem as ações de modelagem de processos de negócio, e seus impactos na organização passam a ser os mais benéficos possíveis, pois dá a visibilidade necessária para as etapas seguintes à modelagem, onde serão almejadas as mudanças consistentes de curto e longo prazo que farão a organização emergir no mercado com mais força, com processos corporativos padronizados, com ganho expressivo de produtividade e eficiência, além de garantir medições, análises e aperfeiçoamento da gestão estratégica.

# Pensamento ...

**A NASA:** Antes dos anos 60 a NASA iniciou o envio de astronautas para o espaço e os mesmos advertiram que as suas “**esferográficas**” não funcionariam sem a gravidade zero, dado que a tinta não desceria à superfície onde se desejava escrever.



# Pensamento ...

Ao fim de 6 anos de testes e investigações que exigiu, um gasto de 12 milhões de dólares, conseguiram desenvolver uma esferográfica que funcionava em gravidade zero, debaixo de água, sobre qualquer superfície incluindo vidro e num leque de temperaturas que iam desde abaixo de zero até 300 graus centígrados.



# Pensamento ...

O que os Russos fizeram??????



# Pensamento ...

Os Russos descartaram as esferográficas e simplesmente utilizaram **lápiz** para que suas tripulações pudessem escrever sem problemas.





“Nada na vida deve ser  
temido, somente  
compreendido.  
Agora é hora de  
compreender mais  
para temer menos”



**Marié Curié**



# Obrigado!

**Se precisar ...**

Prof. Claudio Benossi

**[Claudio.benossi@fatec.sp.gov.br](mailto:Claudio.benossi@fatec.sp.gov.br)**

