

# Análise e Projetos de Sistemas

**Aula: Introdução a Requisitos de Software , Engenharia de Requisitos, Requisitos de Usuários e Sistema.**



Prof. Anderson Augusto Bosing

# **Como entender e atender as necessidades de um cliente?**



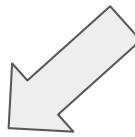
# O que são Requisitos?



# Caso você fosse comprar um PC. Quais seriam os Requisitos?



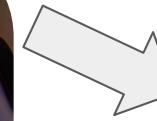
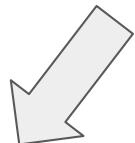
Placa de Vídeo RTX



Placa mãe



256 GB de SSD



16 GB de memória RAM



# O que são Requisitos?

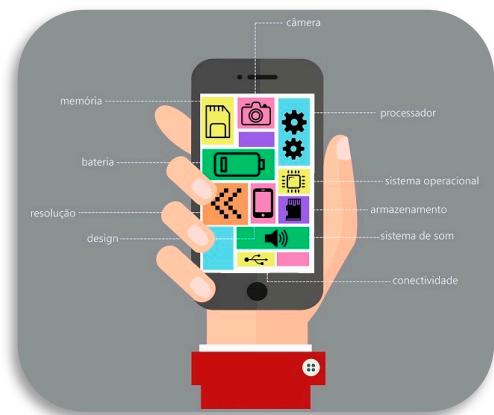
**Um requisito consiste em uma propriedade, característica ou comportamento que um produto deve atender.**



# E o que são Requisitos de Software ?

**Um requisito de software é simplesmente uma declaração do que o sistema deve fazer ou de quais características ele precisa possuir.**

**Quais serviços oferece.  
Seus Atributos.  
Restrições de Funcionamento.**



# **Gerenciamento de Requisitos**

**Etapa inicial no processo de desenvolvimento de softwares, fundamental para o sucesso do projeto.**

**É nessa etapa que são determinados:**

- ✓ As funcionalidades e os perfis desejados por clientes e usuários.**
- ✓ O projeto dos desenvolvedores para atingir esses requisitos.**



# O que é engenharia de requisitos?



# O que é engenharia de requisitos?

**O processo de descobrir, analisar, documentar e verificar esses serviços e restrições.(Sommerville, Engenharia de Software, 2011)**

**Engenharia de Requisitos** é o nome que se dá ao conjunto de atividades relacionadas com a descoberta, análise, especificação e manutenção dos requisitos de um sistema.(Marco Túlio Valente. Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade, 2020.)



# **E quais problemas a engenharia de requisitos resolve?**

**A falta de entendimento total ou parcial dos problemas que eles estavam resolvendo.**

**Os detalhes da solução abordada e a forma correta de resolver.**

**Aumento da qualidade de entrega do software e redução de custos de manutenções.**

**Estabelecendo e mantendo um acordo entre cliente/usuário e a equipe do projeto nas mudanças dos requisitos.**



# **Classificações de Requisitos**

## **Requisitos de Usuário**

**Segundo Sommerville:**

**Requisitos de Usuário – Descrevem os requisitos... de modo que eles sejam compreensíveis pelos usuários do sistema que não possuem conhecimento técnico detalhado.**

**Suponha, por exemplo, um sistema bancário. Um requisito de usuário — especificado pelos funcionários do banco — pode ser o seguinte: o sistema deve permitir transferências de valores para uma conta corrente de outro banco, por meio de TEDs.**



# **Tipos de Requisitos**

## **Requisitos de Sistema**

**Segundo Sommerville:**

**Requisitos de sistema** são descrições mais detalhadas das funções, serviços e restrições operacionais do sistema de software. O documento de requisitos do sistema (às vezes, chamado especificação funcional) deve definir exatamente o que deve ser implementado. Pode ser parte do contrato entre o comprador do sistema e os desenvolvedores de software.

**Utilizando o mesmo exemplo do Sistema Bancário Anterior.**

Esse requisito dá origem a um conjunto de requisitos de sistema, os quais vão detalhar e especificar o protocolo a ser usado para realização de tais transferências entre bancos. Portanto, requisitos de usuário estão mais próximos do problema, enquanto que requisitos de sistema estão mais próximos da solução.



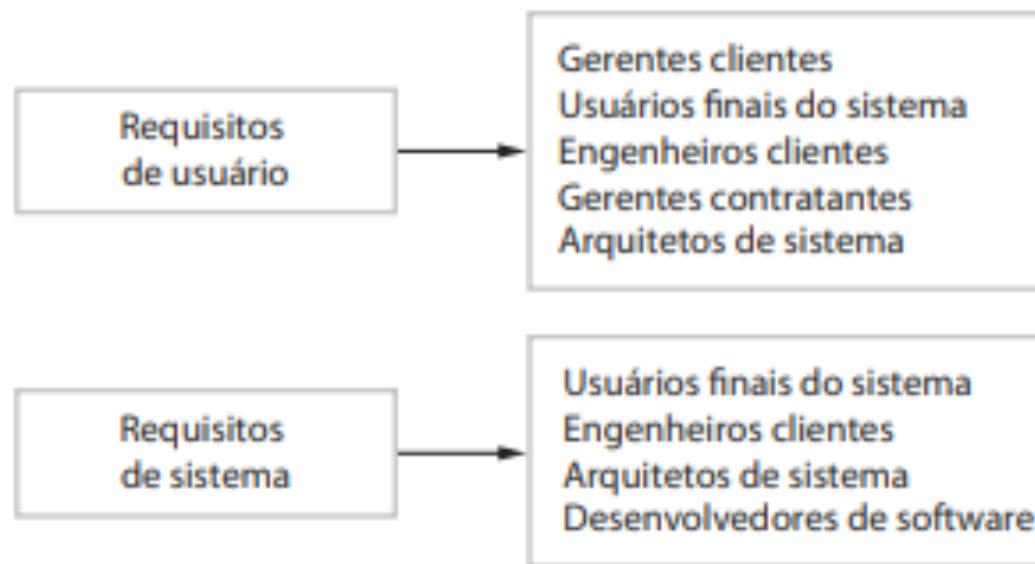
**Os requisitos precisam ser escritos em diferentes níveis de detalhamento para que diferentes leitores possam usá-los de diversas maneiras.**

**Requisitos de usuário não costumam se preocupar com a forma como o sistema será implementado.**

**Requisitos de sistema precisam saber mais detalhadamente o que o sistema fará, porque estão interessados em como ele apoiará os processos dos negócios ou estão envolvidos na implementação do sistema.**



# Leitores de diferentes tipos de especificação de requisitos



# Vamos Praticar ?

**Exercício Requisitos de Usuário X Requisitos de Sistemas.**

**Exercício disponível em:**

[https://docs.google.com/document/d  
1zBKK9RrO3fBt44XGEbak3p5v8NG7yGVR/edit?usp=shar  
ing&ouid=112890355844915434016&rtpof=true&sd=true.](https://docs.google.com/document/d/1zBKK9RrO3fBt44XGEbak3p5v8NG7yGVR/edit?usp=sharing&ouid=112890355844915434016&rtpof=true&sd=true)



## **Requisitos de usuário e de sistema**

### **Esse exemplo, de um Sistema de Gerenciamento da Saúde Mental de Pacientes (MHC-PMS, do inglês Mental Health Care Patient Management System)**

1. O MHC-PMS deve gerar relatórios gerenciais mensais que mostrem o custo dos medicamentos prescritos por cada clínica durante aquele mês.

- 1.1 No último dia útil de cada mês deve ser gerado um resumo dos medicamentos prescritos, seus custos e as prescrições de cada clínica.
- 1.2 Após 17:30h do último dia útil do mês, o sistema deve gerar automaticamente o relatório para impressão.
- 1.3 Um relatório será criado para cada clínica, listando os nomes dos medicamentos, o número total de prescrições, o número de doses prescritas e o custo total dos medicamentos prescritos.
- 1.4 Se os medicamentos estão disponíveis em diferentes unidades de dosagem (por exemplo, 10 mg, 20 mg), devem ser criados relatórios separados para cada unidade.
- 1.5 O acesso aos relatórios de custos deve ser restrito a usuários autorizados por uma lista de controle de gerenciamento de acesso



# Vamos Praticar ?

**Exercício Requisitos de Usuário X Requisitos de Sistemas.**

**Exercício disponível em:**

**<https://docs.google.com/document/d/1xzk4-B6Ftpls80j0iWZYtSaSi1a5jeN1/edit?usp=sharing&ouid=12890355844915434016&rtpof=true&sd=true>**



## **Requisitos de usuário e de sistema**

**Esse exemplo, O sistema LIBSYS um sistema de biblioteca que fornece uma interface única para uma série de banco de dados de artigos em bibliotecas diferentes .**

1.LIBSYS deve manter o acompanhamento de todos os dados exigidos pelas agências de licenciamento de direitos autorais no Reino Unido e em outros lugares .

1.1 Ao solicitar um documento ao LIBSYS , deve ser apresentado ao solicitante um formulário que registra os detalhes do usuário e da solicitação feita .

1.2 Os formulários de solicitação do LIBSYS devem ser armazenados no sistema durante cinco anos , a partir da data da solicitação .

1.3 Todos os formulários do LIBSYS devem ser indexados por usuário , nome do material solicitado e fornecedor da solicitação .

1.4 O LIBSYS deve manter um registro de todas as solicitações feitas ao sistema .

1.5 Para materiais aos quais se aplicam os direitos de empréstimo dos autores , os detalhes do empréstimo devem ser enviados mensalmente às agências de licenciamento de direitos autorais que se registraram no LIBSYS .

# **Leitores de diferentes tipos de especificação de requisitos**

**Os diferentes tipos de leitores de documento exibidos são exemplos de stakeholders do sistema. Incluem qualquer um que seja afetado de alguma maneira pelo sistema.**

**Exemplos:**

- ✓ Médicos responsáveis por avaliar e tratar os pacientes.
- ✓ Recepçãonistas que marcam as consultas dos pacientes;
- ✓ Equipe de TI responsável pela instalação e manutenção do sistema;



# REVISANDO

- Engenharia de Requisitos.
- Problemas que a Engenharia de Requisitos Resolve.
- Requisitos de Sistema.
- Requisitos de Usuário.



# Engenharia de Software I

**Aula: Requisitos Funcionais e  
Não Funcionais.**



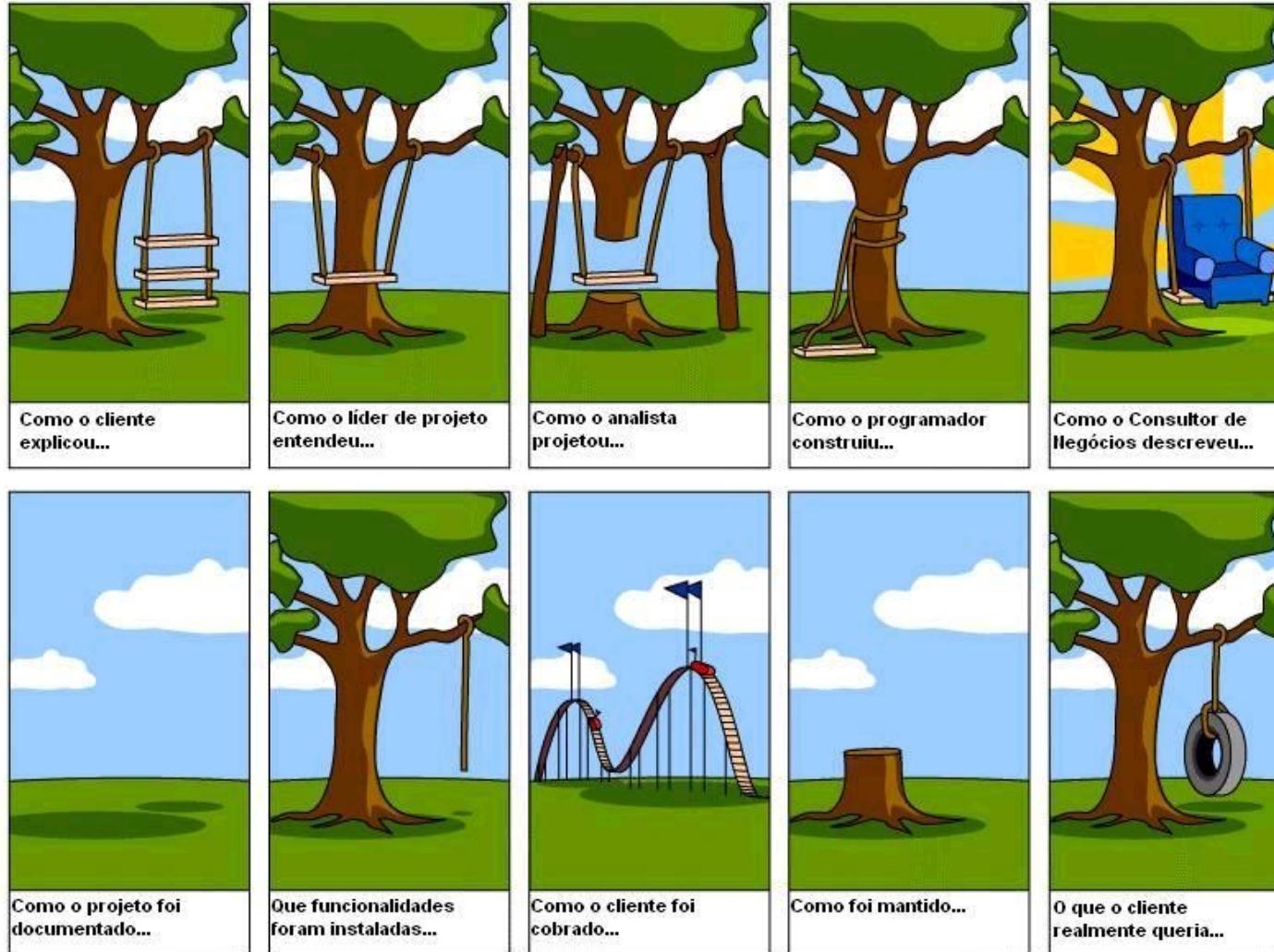
Prof. Anderson Augusto Bosing

# **Revisando a Ultima Aula**

- Como entender e atender as necessidades do cliente ?
- O que são Requisitos ?
- O que são Requisitos de Software ?
- Engenharia de Requisitos.
- Problemas que a Engenharia de Requisitos resolve.
- Requisitos de usuário x Requisitos de Sistema
- Leitores diferentes para cada um dos requisitos.



# Qual a motivação para gerenciar os requisitos ?



# Requisitos funcionais e não funcionais

**Os requisitos de software são frequentemente classificados como requisitos funcionais e requisitos não funcionais:**

**Requisitos funcionais.** São declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem explicitar o que o sistema não deve fazer.



# **Requisitos funcionais**

**Os requisitos funcionais de um sistema descrevem o que ele deve fazer.(Sommerville, Engenharia de Software, 2011).**

**Ex:**

**Incluir/Excluir/Alterar os dados de clientes.**  
**Efetuar cobranças de compras efetuadas pelos clientes.**  
**Consultar saldos de estoque de um produto.**



# **Requisitos funcionais**

**Por exemplo, requisitos funcionais para o sistema MHC-PMS, usados para manter informações sobre os pacientes em tratamento por problemas de saúde mental:**

- 1. Um usuário deve ser capaz de pesquisar as listas de agendamentos para todas as clínicas.**
- 2. O sistema deve gerar a cada dia, para cada clínica, a lista dos pacientes para as consultas daquele dia.**
- 3. Cada membro da equipe que usa o sistema deve ser identificado apenas por seu número de oito dígitos.**



# Requisitos funcionais e não funcionais

**Requisitos não funcionais.** São restrições aos serviços ou funções oferecidos pelo sistema. Incluem restrições de timing, restrições no processo de desenvolvimento e restrições impostas pelas normas.

Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo.



# **Requisitos Não Funcionais**

**Os requisitos não funcionais como o nome sugere, são requisitos que não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários. Eles podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, tempo de resposta, desempenho, plataformas suportadas.(Sommerville, Engenharia de Software, 2011)**



# Requisitos Não Funcionais

## Tipos:

- Aparência
- Usabilidade
- Desempenho
- Operacional
- Manutenção e suporte
- Segurança
- Cultura organizacional
- Legal



# Requisitos Não Funcionais

## Aparência

- O produto deve ter um estilo igual ao dos outros produtos da empresa.
- O produto deve ser atrativo para usuários adolescentes.
- O produto deve ser identificável com a empresa onde será usado.



# Requisitos Não Funcionais

## Usabilidade

- O produto deve limpar o campo e exibir mensagem de erro quando o usuário entrar com dados incorretos. (facilidade de uso)
- O produto deve ser especialmente intuitivo de usar para crianças com 4 anos de idade. (facilidade de aprendizagem)
- O produto deve ser apresentado ao usuário na língua portuguesa e inglesa. (personalização)



# Requisitos Não Funcionais

## Desempenho

- O produto deve identificar o funcionário com base na foto em menos de 3 segundos. (tempo de resposta)
- O produto deve trabalhar em modo local, se perder a conexão com o servidor (disponibilidade).
- O produto deve calcular os juros até os décimos de centavos. (precisão dos resultados)



# Requisitos Não Funcionais

## Operacional

- O produto deve operar debaixo de água até a profundidade de 30 metros.
- O produto deve carregar os dados em lote por meio de arquivos texto.
- O produto deve exportar o *curriculum vitae* no formato PDF.



# **Requisitos Não Funcionais**

## **Manutenção e suporte**

- O programa do produto deve conter comentários.**
- O produto deve estar preparado para ser utilizado em qualquer língua.**
- O produto deve disponibilizar um tutorial que explique como operá-lo.**



# Requisitos Não Funcionais

## Segurança

- Os dados da avaliação de desempenho de um funcionário devem ser fornecidos apenas ao próprio funcionário e aos seus superiores.
- O produto deve garantir que só usuários registrados tenham acesso aos dados clínicos dos pacientes.
- O produto deve solicitar senha de acesso com no mínimo 8 dígitos, que contenha letras, números e caracteres especiais.



# Requisitos Não Funcionais

## Cultural e organizacional

- O produto deve usar português do Brasil.
- O produto deve mostrar os feriados locais no calendário.
- O produto deve usar componentes fabricados no Mercosul.



# Requisitos Não Funcionais

## Legal

- O produto deve estar alinhado ao referencial de gestão do PMBOK.
- O produto deve ser certificado pela Autoridade Tributária e Aduaneira.
- O produto deve atender às normas estabelecidas na Lei nº 8112/90.



# **Requisitos Não Funcionais**

**Ex:**

**Uso de Design responsivo nas interfaces gráficas(Possibilita o uso em diferentes dispositivos como tablets, smartphones e computadores).**

**Compatibilidade com sistemas operacionais Windows e Linux.**

**Deve possuir conformidade com as normativas da LGPD.**



# Requisitos Não Funcionais

Por exemplo, requisitos não funcionais para o sistema MHC-PMS, usados para manter informações sobre os pacientes em tratamento por problemas de saúde mental: O MHC-PMS deve estar disponível para todas as clínicas durante as horas normais de trabalho (segunda a sexta-feira, 8h30 às 17h30). Períodos de não operação dentro do horário normal de trabalho não podem exceder cinco segundos em um dia.



# **Requisitos Não Funcionais**

**Usuários do sistema MHC-PMS devem se autenticar com seus cartões de identificação da autoridade da saúde.**

**O sistema deve implementar as disposições de privacidade dos pacientes, tal como estabelecido no HStan-03-2006-priv.**



# **Requisitos Não Funcionais**

**A equipe médica deve ser capaz de usar todas as funções do sistema após quatro horas de treinamento.**

**Após esse treinamento, o número médio de erros cometidos por usuários experientes não deve exceder dois por hora de uso do sistema.**



# **Requisito de software - Classificação**

**Podem ser classificados ainda, seja funcional ou não funcional como:**

- **Requisitos primários** –provém diretamente de alguma parte interessada, ou seja, foi solicitado por uma pessoa ou entidade.
- **Requisitos secundários** –são obtidos por refinamento de um requisito primário .



# Requisito de software - Classificação

Requisitos não funcionais podem ser divididos em:

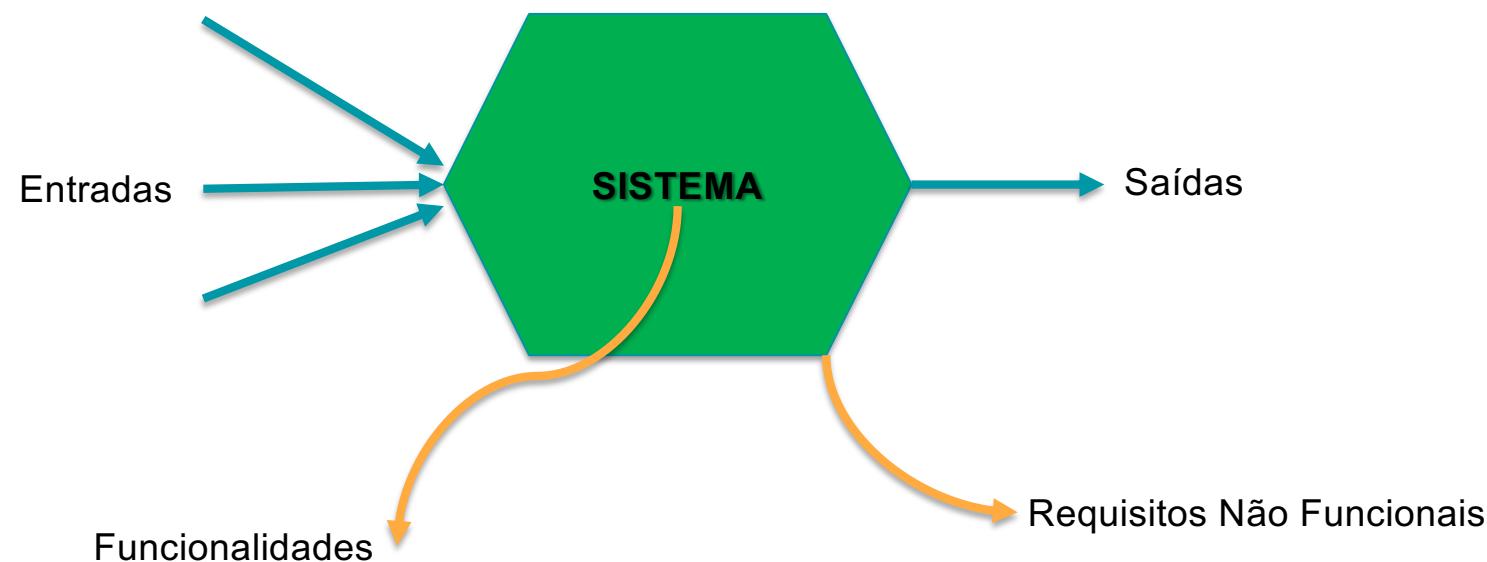
- Requisitos de produto – caracterizam aspectos do funcionamento do produto, ex: confiabilidade, desempenho, eficiência, portabilidade, usabilidade, testabilidade, manutenibilidade.
- Requisitos organizacionais – provém de estratégias e procedimento estabelecidos no contexto do fabricante do produto ou da organização do cliente, ex: normas a serem seguidas, requisitos de implementação, como a linguagem de programação a usar.
- Requisitos externos – tem origem em fatores externos ao produto ou ao desenvolvimento, ex: requisitos de interoperabilidade que definem como os produtos interagem com outros sistemas, requisitos legais, requisitos éticos.



# Métricas para especificar requisitos não funcionais.

Propriedade	Medida
Velocidade	Transações processadas/segundo Tempo de resposta de usuário/evento Tempo de atualização de tela
Tamanho	Megabytes Número de chips de memória ROM
Facilidade de uso	Tempo de treinamento Número de frames de ajuda
Confiabilidade	Tempo médio para falha Probabilidade de indisponibilidade Taxa de ocorrência de falhas Disponibilidade
Robustez	Tempo de reinício após falha Percentual de eventos que causam falhas Probabilidade de corrupção de dados em caso de falha
Portabilidade	Percentual de declarações dependentes do sistema-alvo Número de sistemas-alvo

# O que os Requisitos de Software Especificam?



# **Gerenciamento de Requisitos Não é Fácil, Porque...?**

- Nem sempre são óbvios
- Podem vir de muitas fontes
- Nem sempre é fácil de expressar claramente com Palavras.
- Relatado por um e entregue por outro no processo de engenharia de software.
- Tem propriedade única.
- Mudam.
- Em grande quantidade são difíceis de controlar.



# **Qualidades do Requisito de Software**

- Verificável.
- Priorizável por importância.
- Modificável.
- Rastreável.
- Compreensível.
- Correto.
- Completo.
- Consistente.
- Não Ambíguo.
- Testável.



# Como especificar Requisitos ?



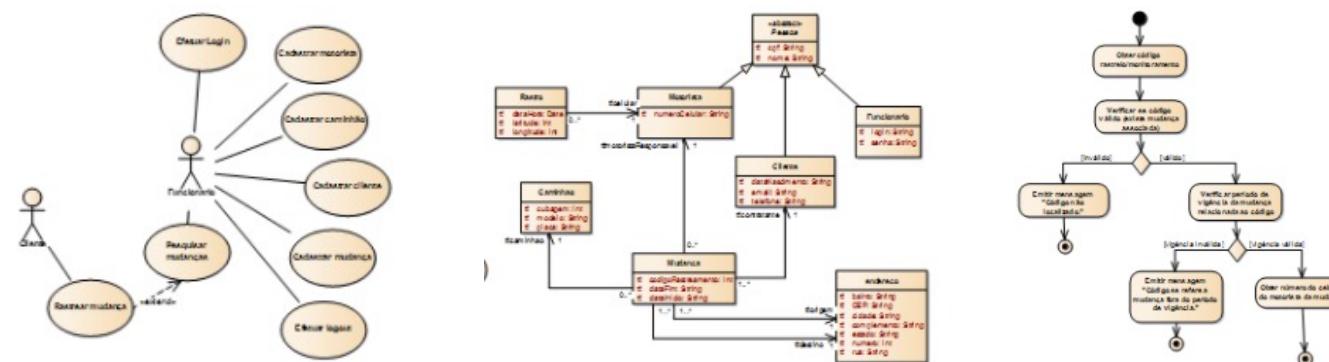
# Como especificar Requisitos ?

- Como funcionário gostaria de consultar os motoristas da empresa disponíveis para realizar uma mudança.
- Como funcionário gostaria de cadastrar os caminhões da empresa para controlar a frota.
- Como funcionário gostaria de registrar o contrato de mudança para controlar os serviços da empresa.
- Como cliente gostaria de rastrear a minha mudança para saber quando chegará ao destino.

- O sistema deve permitir ao funcionário cadastrar um motorista da empresa.
- O sistema deve permitir ao funcionário cadastrar um caminhão da empresa.
- O sistema deve permitir ao funcionário cadastrar um cliente.
- O sistema deve permitir ao funcionário cadastrar um serviço de mudança contra
- O sistema deve permitir ao funcionário e o cliente rastrear uma mudança.

## Linguagem Natural

## Modelos Conceituais



# Como especificar Requisitos ? Linguagem Natural.

A linguagem natural é a forma de documentação de requisitos mais aplicada na prática.

**Vantagens:**

Nenhum stakeholder precisa aprender uma nova notação.

Qualquer requisito pode ser expresso em linguagem natural.

**Desvantagens:**

Pode resultar em requisitos ambíguos.

Requisitos de diferentes tipos pode ser misturados.

Em ambas as situações acima podem ocorrer mesmo que não intencionalmente.



# **Como especificar Requisitos ?**

## **Linguagem Conceitual.**

Ao documentar requisitos por meio de modelos, linguagens de notação especiais para modelagem devem ser utilizadas(UML, BPMN).

### **Vantagens:**

**Modelos retratam os requisitos de forma mais compacta, sendo de compreensão mais fácil para um leitor treinado.**

**Oferecem menor grau de ambiguidade devido ao seu maior grau de formalidade.**

### **Desvantagens:**

**Exigem conhecimentos específicos de modelagem.**

**Retratam apenas uma perspectiva específica.**



# VAMOS PRATICAR ?

Levante alguns requisitos com base na descrição abaixo:

O Professor Anderson precisa de um sistema para que sejam armazenadas as notas e os dados dos alunos. Assim que fechada a média final os alunos devem receber um e-mail automaticamente com a sua média. O Professor Anderson deseja utilizar este software tanto no seu notebook quanto em seu smartphone android.



# **Correção do Exercício Anterior:**

**“O Professor Anderson precisa de um sistema para que sejam armazenadas as notas e os dados dos alunos.**

**Assim que fechada a média final os alunos deve receber um e-mail automaticamente com a sua média. Ele deseja utilizar este software tanto no seu notebook quanto em seu smartphone android.”**

- 1. Cadastrar Alunos:** O sistema deve permitir o cadastro de alunos, contendo informações como nome, e-mail e outras informações pessoais. **Cadastro de Alunos:** O sistema deve permitir o cadastro de alunos, contendo informações como nome, e-mail e outras informações pessoais.
- 2. Gerenciar Notas:** O sistema deve permitir que o professor gerencie as notas de cada aluno por disciplina.
- 3. Cálculo da Média Final:** O sistema deve calcular a média final dos alunos automaticamente com base nas notas inseridas.



## **Correção do Exercício Anterior:**

**“O Professor Anderson precisa de um sistema para que sejam armazenadas as notas e os dados dos alunos.**

**Assim que fechada a média final os alunos deve receber um e-mail automaticamente com a sua média. Ele deseja utilizar este software tanto no seu notebook quanto em seu smartphone android.”**

- 4. Envio de E-mails:** O sistema deve enviar automaticamente um e-mail para cada aluno com a sua média final assim que ela for fechada.
- 5. Sistema deve ser híbrido(WEB)** e possível de ser utilizado em diversas plataformas como desktop e mobile.



# Atividade de Levantamento de Requisitos de software



# Engenharia de Software I

**Aula: Exercício e O  
documento de requisitos de  
software.**



Prof. Anderson Augusto Bosing

# **Revisando a Ultima Aula**

- **Requisitos funcionais e Não funcionais**
- **Qualidades de um bom Requisito.**



# O Documento de Requisitos de software

O documento de requisitos de software, às vezes chamado Especificação de Requisitos de Software (SRS — do inglês Software Requirements Specification), é uma declaração oficial de o que os desenvolvedores do sistema devem implementar. Deve incluir tanto os requisitos de usuário para um sistema quanto uma especificação detalhada dos requisitos de sistema

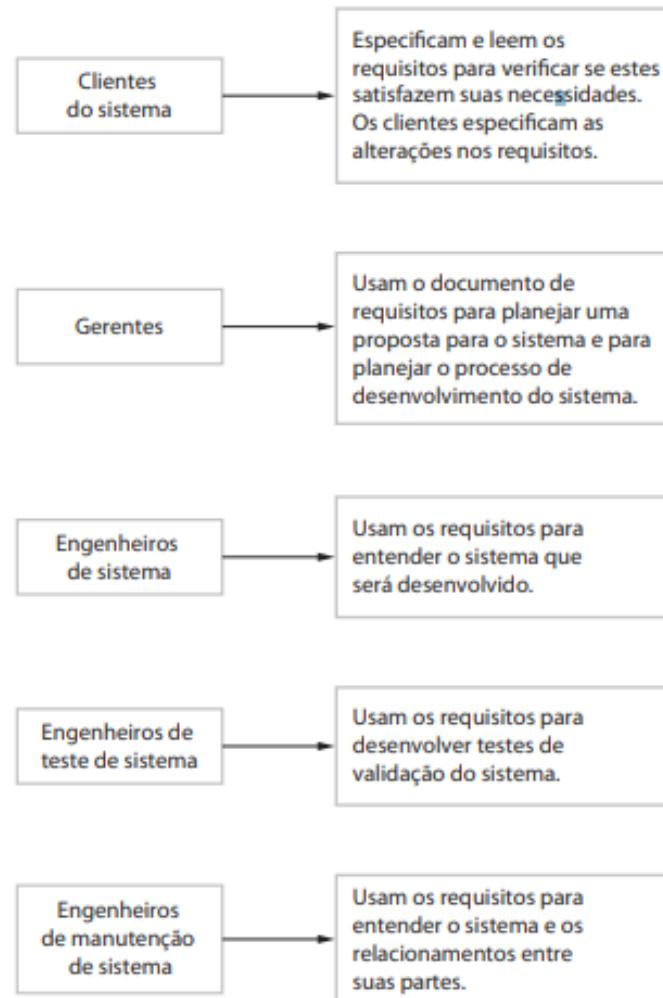


# O Documento de Requisitos de software

O documento de requisitos tem um conjunto diversificado de usuários, que vão desde a alta administração da organização que está pagando pelo sistema até os engenheiros responsáveis pelo desenvolvimento do software. A Figura a seguir, mostra os possíveis usuários do documento e como eles o utilizam.



# O Documento de Requisitos de software



# A estrutura de um documento de requisitos.

Prefácio	Deve definir os possíveis leitores do documento e descrever seu histórico de versões, incluindo uma justificativa para a criação de uma nova versão e um resumo das mudanças feitas em cada versão.
Introdução	Deve descrever a necessidade para o sistema. Deve descrever brevemente as funções do sistema e explicar como ele vai funcionar com outros sistemas. Também deve descrever como o sistema atende aos objetivos globais de negócio ou estratégicos da organização que encomendou o software.
Glossário	Deve definir os termos técnicos usados no documento. Você não deve fazer suposições sobre a experiência ou o conhecimento do leitor.
Definição de requisitos de usuário	Deve descrever os serviços fornecidos ao usuário. Os requisitos não funcionais de sistema também devem ser descritos nessa seção. Essa descrição pode usar a linguagem natural, diagramas ou outras notações comprehensíveis para os clientes. Normas de produto e processos que devem ser seguidos devem ser especificados.
Arquitetura do sistema	Deve apresentar uma visão geral em alto nível da arquitetura do sistema previsto, mostrando a distribuição de funções entre os módulos do sistema. Componentes de arquitetura que são reusados devem ser destacados.



# A estrutura de um documento de requisitos.

Especificação de requisitos do sistema	Deve descrever em detalhes os requisitos funcionais e não funcionais. Se necessário, também podem ser adicionados mais detalhes aos requisitos não funcionais. Interfaces com outros sistemas podem ser definidas.
Modelos do sistema	Pode incluir modelos gráficos do sistema que mostram os relacionamentos entre os componentes do sistema, o sistema e seu ambiente. Exemplos de possíveis modelos são modelos de objetos, modelos de fluxo de dados ou modelos semânticos de dados.
Evolução do sistema	Deve descrever os pressupostos fundamentais em que o sistema se baseia, bem como quaisquer mudanças previstas, em decorrência da evolução de hardware, de mudanças nas necessidades do usuário etc. Essa seção é útil para projetistas de sistema, pois pode ajudá-los a evitar decisões capazes de restringir possíveis mudanças futuras no sistema.
Apêndices	Deve fornecer informações detalhadas e específicas relacionadas à aplicação em desenvolvimento, além de descrições de hardware e banco de dados, por exemplo. Os requisitos de hardware definem as configurações mínimas ideais para o sistema. Requisitos de banco de dados definem a organização lógica dos dados usados pelo sistema e os relacionamentos entre esses dados.
Índice	Vários índices podem ser incluídos no documento. Pode haver, além de um índice alfabético normal, um índice de diagramas, de funções, entre outros pertinentes.



# VAMOS PRATICAR ?

Com base no exercício feito na aula anterior  
mapeie no documento os requisitos funcionais  
e não funcionais.



# Atividade de Levantamento de Requisitos de software



# Engenharia de Software I

**Aula: User Story.**



Prof. Anderson Augusto Bosing

# Histórias de Usuário (User Stories)

Maneira utilizada pelos Métodos Ágeis para documentar requisitos/casos de uso do sistema;

- Uma história de usuário é uma descrição em linguagem natural e informal dos recursos de um sistema de software. Elas são escritas da perspectiva de um usuário final ou usuário de um sistema e podem ser registradas em cartões, notas post-its ou digitalmente no software de gerenciamento do projeto.
- Dependendo do projeto, as histórias de usuário podem ser escritas por diferentes partes interessadas, como cliente, usuário, gerente ou equipe de desenvolvimento



# Histórias de Usuário (User Stories)

As histórias de usuário são um tipo de objeto de fronteira. Eles facilitam a criação de sentido e a comunicação; e pode ajudar as equipes de software a documentar sua compreensão do sistema e seu contexto.

- As histórias de usuário são escritas por ou para usuários ou clientes para descrever a funcionalidade do sistema que está sendo desenvolvido. Em algumas equipes, o gerente de produto (ou proprietário do produto no Scrum) é o principal responsável por formular histórias de usuário e organizá-las em um backlog do produto.
- Em outras equipes, qualquer pessoa pode escrever uma história de usuário. As histórias de usuários podem ser desenvolvidas por meio de discussão com as partes interessadas, baseadas em personas ou simplesmente inventadas.

# Histórias de Usuário (User Stories)

## Writing a user story

- 1 Define your **end user**
- 2 Specify what **they want**
- 3 Describe **the benefit**
- 4 Add **acceptance criteria**

# Histórias de Usuário (User Stories)

Como um <PAPEL>  
posso/devo/gostaria/quero  
<FUNÇÃO>  
Para/de  
<RESULTADO para o NEGÓCIO>



# Histórias de Usuário (User Stories)

Como um vendedor, gostaria de consultar o estoque de um determinado produto para oferecer a um cliente

Como um vendedor, devo registrar a venda realizada para um cliente para manter o histórico de vendas

Como um diretor, gostaria de obter o volume de vendas do mês para acompanhar o atingimento das metas



# Histórias de Usuário (User Stories)

Para definir boas Stories elas devem ser:

I	independent	Uma User Story representa basicamente a visão do usuário no projeto, sendo assim, deve conter informação suficiente sem depender da finalização de outras. Podendo ser assimilada por qualquer pessoa e não necessitando de nenhuma fonte externa ou da leitura de outras User Stories para ser compreendida.
N	negotiable	Uma boa Story deve ser flexível. Seus detalhes devem ser discutidos entre todos os envolvidos no projeto. Esta característica compreende às “conversas” dos 3 C’s ( cartão, conversa e confirmação) das User Stories.
V	valuable	A User Story deve fornecer valor para o cliente, para a equipe do projeto ou para o produto. O valor agregado a ela contribui para priorizar o backlog, se alguma delas não possuírem um valor associado, devem ser colocadas no final da lista.
E	estimable	Deve reunir um número de detalhes suficiente através das conversas, para que seja estimado o seu tempo total de realização, o custo de desenvolvimento, e outras métricas que sejam necessárias. Caso a estória não possa ser estimada, é conveniente reavaliá-la e se necessário dividi-la.
S	small	Boas estórias de Usuários são pequenas, normalmente menores que o trabalho de uma pessoa por um mês. Quanto menor é a estória, mais provável é a precisão das estimativas e da estória ser entendida.
T	testable	Uma estória deve ser testável para que seja aceita. O time deve ser capaz de testá-la, apenas com o critério de aceitação que foi descrito na estória.

# Histórias de Usuário (User Stories)

Para definir boas Stories elas devem conter critérios de aceitação.

- Critérios de aceitação são definidos como "notas sobre o que a história deve fazer para que o proprietário do produto a aceite como completa."
- Eles definem os limites de uma história de usuário e são usados para confirmar quando uma história está concluída e funcionando como pretendido.
- A quantidade apropriada de informações a serem incluídas nos critérios de aceitação varia de acordo com a equipe, programa e projeto.
- Para que uma história seja considerada concluída ou concluída, todos os critérios de aceitação devem ser atendidos



# Histórias de Usuário (User Stories) vs Casos de Uso

- Apesar de uma user story deixar claro que uma funcionalidade é requisitada no sistema por seus usuários, ela não dá ideia de como a funcionalidade vai ser (i.e., como o sistema vai se comportar), enquanto que os casos de uso dão essa ideia.
- No geral, uma user story é só uma forma de registrar a existência de um requisito, sem muitos detalhes, enquanto que um caso de uso é a documentação do resultado de se pensar em como aquele requisito vai ser tratado pelo sistema na prática.
- User stories são centradas no resultado e no benefício da coisa que se está descrevendo, enquanto casos de uso são mais granulares e descrevem como o sistema irá agir.
- User stories são apenas o início do processo de entendimento dos requisitos do sistema. Para os desenvolvedores que implementarão o sistema, casos de uso parecem um jeito muito melhor de descrever requisitos do que user stories. Mas, descrever casos de uso demanda esforço.



# Engenharia de Software I

**Aula: Técnicas de Elicitação  
de Requisitos**



Prof. Anderson Augusto Bosing

# Técnicas de Elicitação

As “primeiras perguntas” darão somente um entendimento básico do problema;

Diversas técnicas podem ser utilizadas no levantamento de requisitos, as quais podem possuir diferentes objetos de investigação ou podem ter foco em tipos diferentes de requisitos.

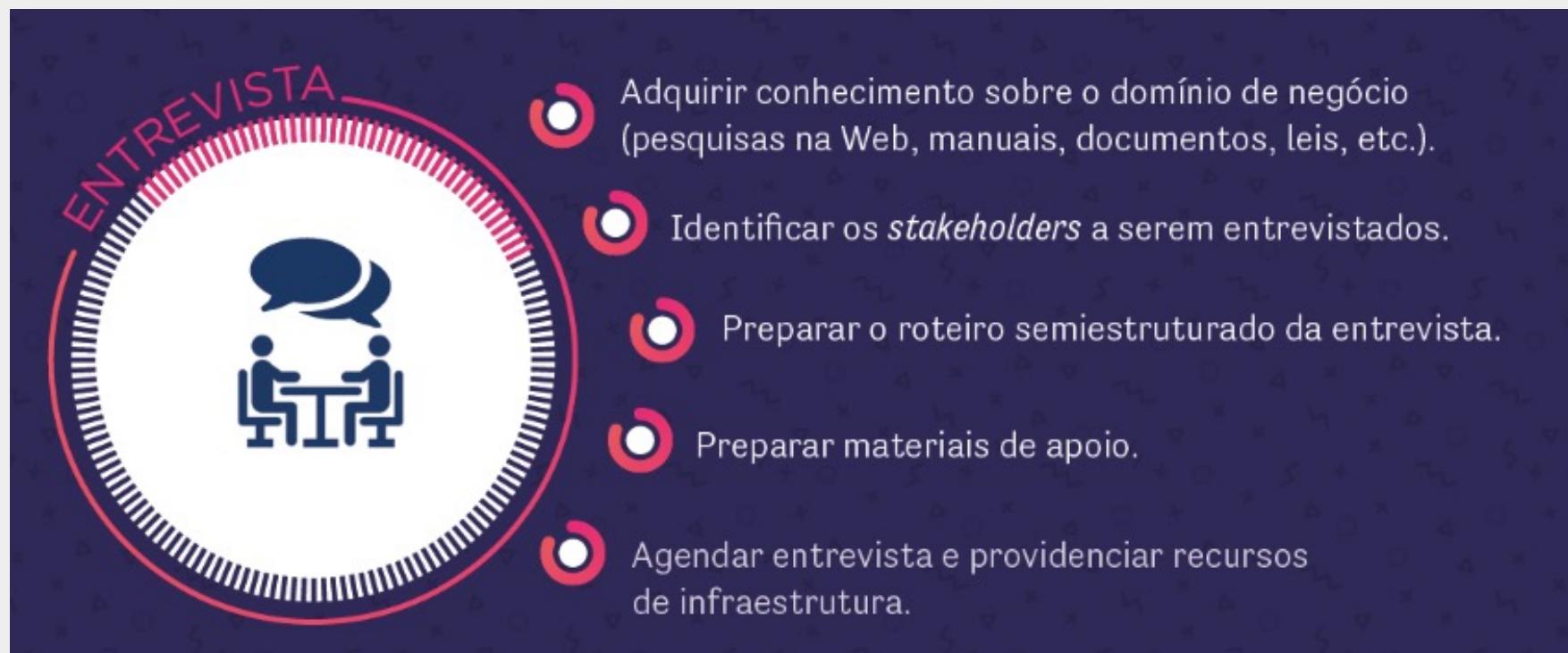
Assim, é útil empregar várias dessas técnicas concomitantemente, de modo a se ter um levantamento de requisitos mais eficaz. Dentre as várias técnicas, podem ser citadas:

Entrevista;  
Questionário;  
Cenários.  
Prototipação;  
Análise de Documentos;

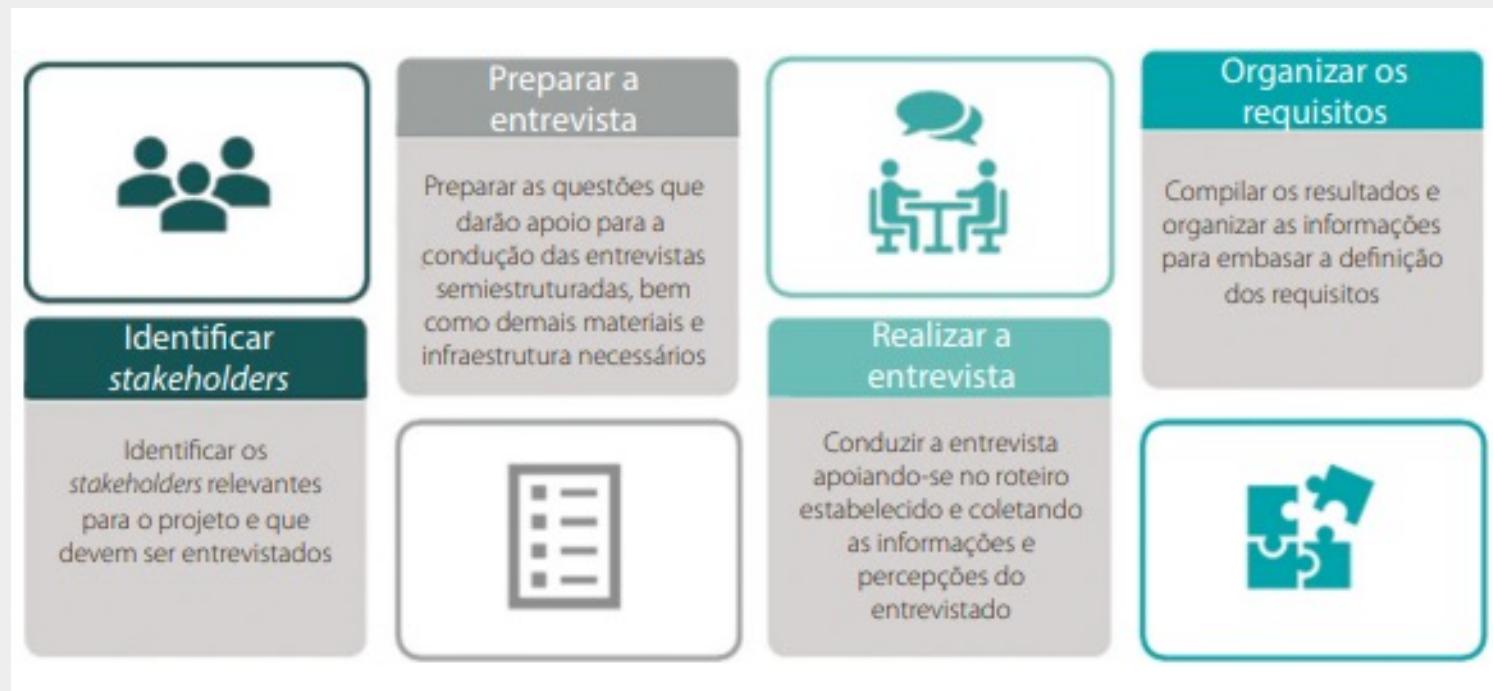


# Entrevistas

Técnica amplamente utilizada, que consiste em conversas direcionadas com um propósito específico e com formato “pergunta-resposta”. Seu objetivo é descobrir problemas a serem tratados, levantar procedimentos importantes e saber a opinião e as expectativas do entrevistado sobre o sistema.

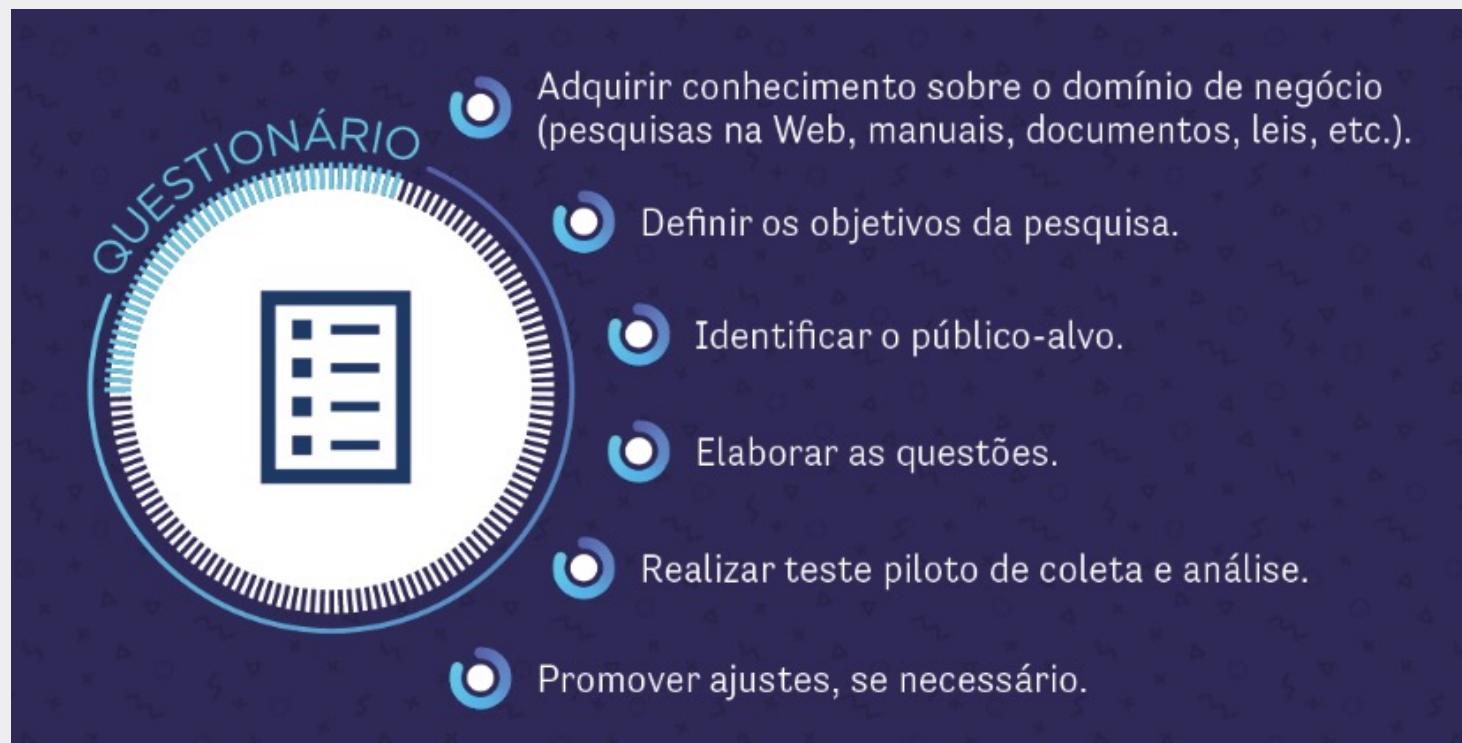


# Entrevistas

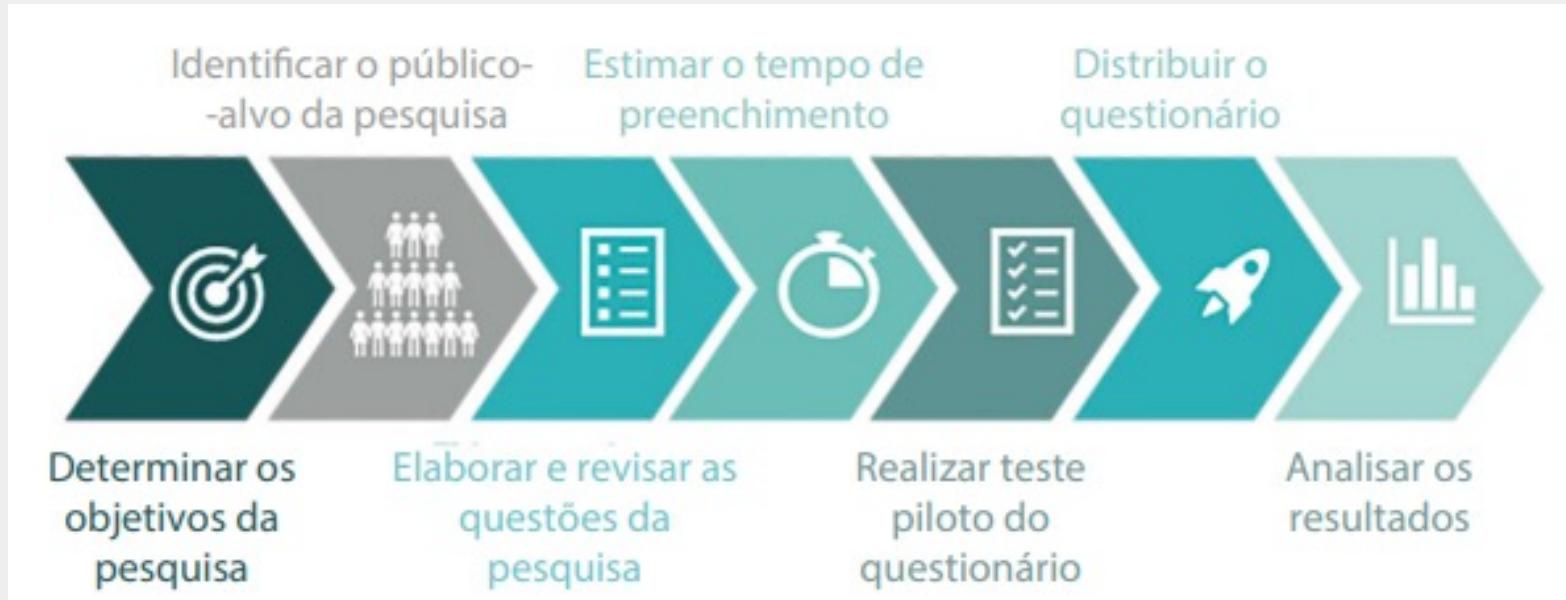


# Questionários

O uso de questionários possibilita ao analista obter informações como postura, crenças, comportamentos e características de várias pessoas que serão afetas pelo sistema.



# Questionários



# Análise de documentos

Pela análise de documentos existentes na organização, analistas capturam informações e detalhes difíceis de conseguir por entrevista e observação. Documentos revelam um histórico da organização e sua direção.



# Cenários

Com o uso desta técnica, um cenário de interação entre o usuário final e o sistema é montado e o usuário simula sua interação com o sistema nesse cenário, explicando ao analista o que ele está fazendo e de que informações ele precisa para realizar a tarefa descrita no cenário. O uso de cenários ajuda a entender requisitos, a expor o leque de possíveis interações e a revelar facilidades requeridas.



# Prototipagem

Um protótipo é uma versão preliminar do sistema, muitas vezes não operacional e descartável, que é apresentada ao usuário para capturar informações específicas sobre seus requisitos de informação, observar reações iniciais e obter sugestões, inovações e informações para estabelecer prioridades e redirecionar planos.

Principal / Livros Cadastrados / Novo Livro

## Cadastro de Livro

Ricardo Pereira

Voltar

Nome do Livro ISBN Editora

Novatec

Categoria Autor Ano de Publicação

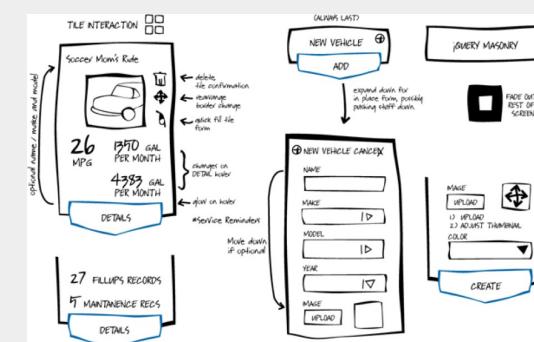
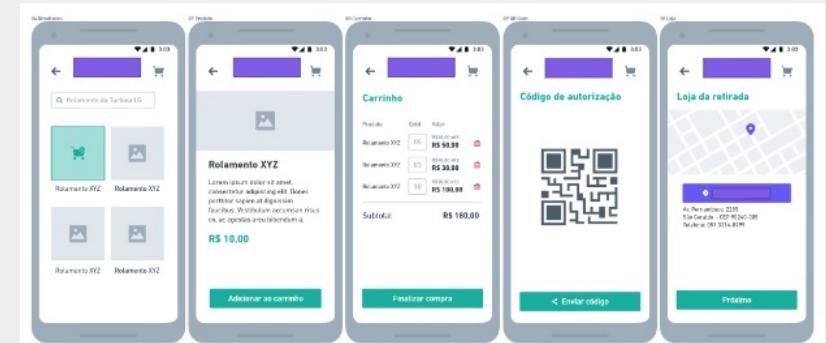
Novatec

Imagens Novo Imagem Tamanho mínimo: 800x600px

Quantidade

Salvar Limpar

Principal Livros Categorias Editoras Relatórios Usuários Vendas Estoque



# Técnicas de Elicitação

Realizar uma pesquisa sobre as técnicas abaixo e responder ao questionário:

Joint Application Design(JAD)

Etnografia

Brainstorming

Workshops



# Engenharia de Software

**Aula:**  
**A Linguagem UML.**  
**Visões de um Sistema.**  
**Diagramas UML.**



Prof. Anderson Augusto Bosing

# **Revisando a Ultima Aula**

- Requisitos funcionais e Não funcionais**
- Requisitos em Grande Quantidade podem ser difíceis de gerenciar.**



# A Linguagem UML

A Unified Modeling Language (UML, Linguagem de Modelagem Unificada) é “uma linguagem-padrão para descrever/documentar projeto de software.

A UML pode ser usada para visualizar, especificar, construir e documentar os artefatos de um sistema de software.



# A Linguagem UML

## Motivação

Em outras palavras, assim como os arquitetos criam plantas e projetos para serem usados por uma empresa de construção, os arquitetos de software criam diagramas UML para ajudar os desenvolvedores de software a construir o software. Se você entender o vocabulário da UML (os elementos visuais do diagrama e seus significados), poderá facilmente entender e especificar um sistema e explicar o projeto desse sistema para outros interessados.



# A Linguagem UML

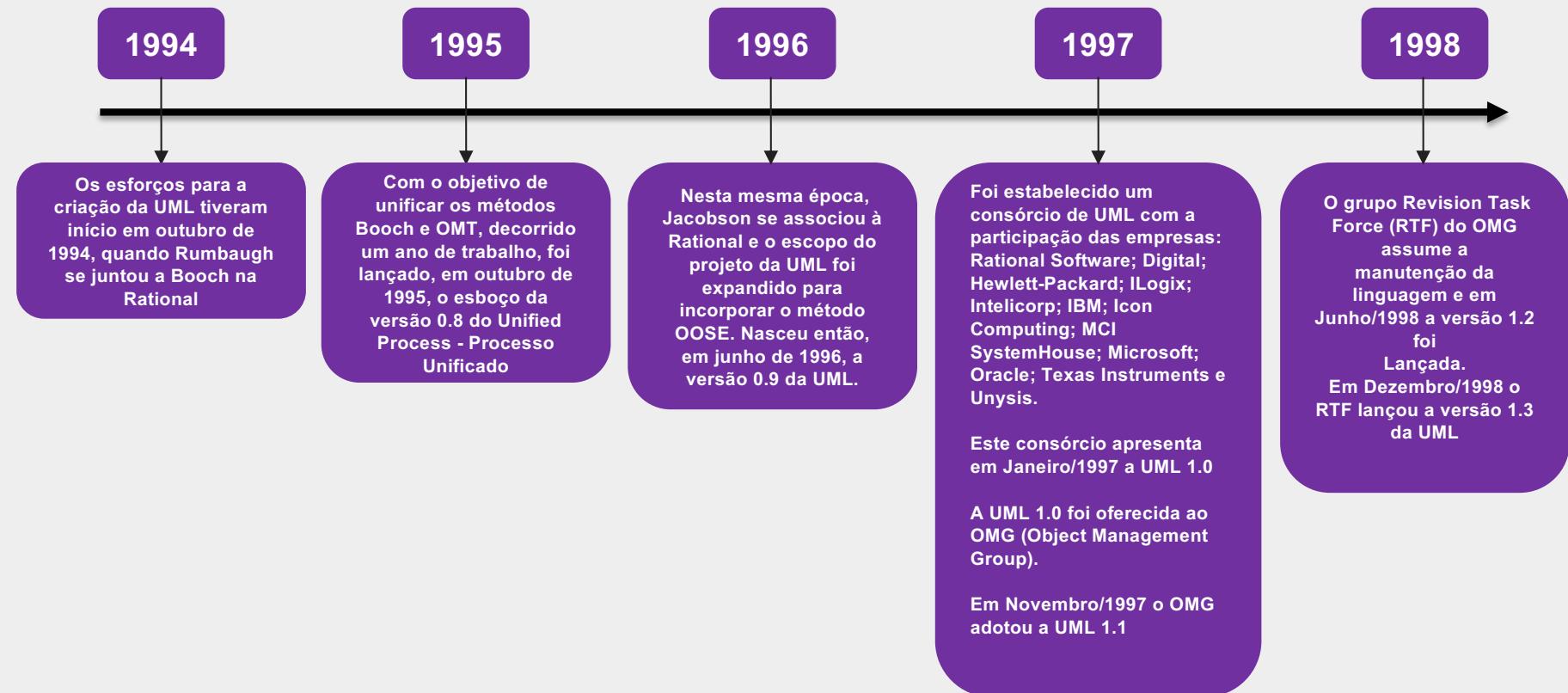
**A construção da UML teve muitos contribuintes, mas os principais atores no processo foram Grady Booch, James Rumbaugh e Ivar Jacobson.**

**Em 1997 a UML foi aprovada como padrão pelo OMG(Object Management Group).**

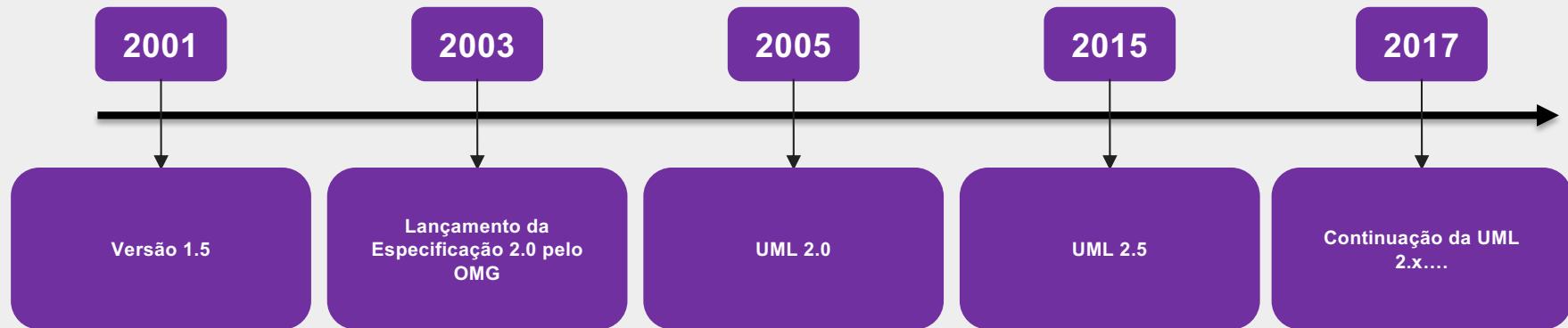
**Desde então, a UML tem tido grande aceitação pela comunidade de desenvolvedores de sistemas.**



# A Linguagem UML Timeline



# A Linguagem UML Timeline

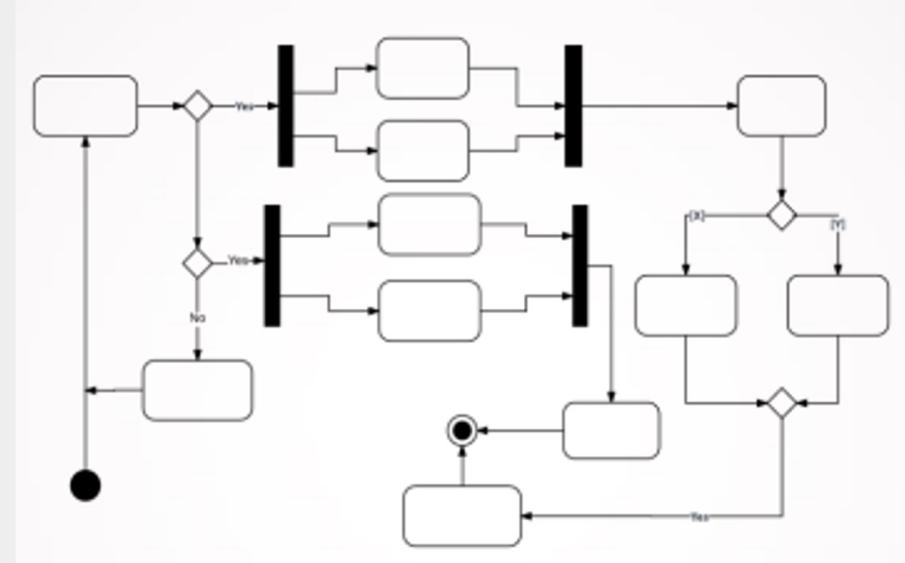


# Visão Geral da UML



A UML é uma linguagem visual para modelar sistemas orientados a objetos.

Por meio dos elementos gráficos definidos nesta linguagem pode-se construir diagramas que representam diversas perspectivas de um sistema.



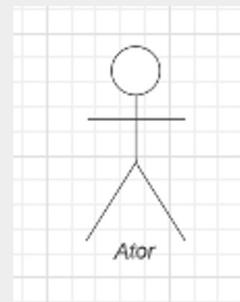
# Visão Geral da UML



Cada **elemento gráfico** da UML possui uma sintaxe e uma semântica.

A **sintaxe** de um elemento corresponde à forma predeterminada de desenhar o elemento.

A **semântica** define o que significa o elemento e com que objetivo ele deve ser utilizado.



# Visão Geral da UML

A UML é independente tanto de linguagens de programação quanto de processos de desenvolvimento.

Esse é um fator importante para a utilização da UML, pois diferentes sistemas de software requerem abordagens diversas de desenvolvimento.



# Visão Geral da UML

O desenvolvimento de um software complexo demanda que seus desenvolvedores tenham a possibilidade de examinar e estudar esse sistema a partir de perspectivas diversas.

Os autores da UML sugerem que um sistema pode ser descrito por cinco visões:

- Visão de Casos de Uso.
- Visão de Projeto.
- Visão de implementação.
- Visão de Implantação.
- Visão de Processo.



# **Visão Geral da UML**

## **Visão de Caso de Uso**

**Descreve o sistema de um ponto de vista externo como um conjunto de interações entre o sistema e os agentes externos ao sistema.**

**Esta visão é criada em um estágio inicial e direciona o desenvolvimento das outras visões do sistema.**



# **Visão Geral da UML**

## **Visão de Projeto(Lógica)**

**Enfatiza as características do sistema que dão suporte, tanto estrutural quanto comportamental, às funcionalidades externamente visíveis do sistema.**

**Ligada ao problema do negócio.**



# **Visão Geral da UML**

## **Visão de Implementação**

**Abrange o gerenciamento de versões do sistema, construídas pelo agrupamento de módulos (componentes) e subsistemas.**



# **Visão Geral da UML**

## **Visão de Implantação**

**Corresponde à distribuição física do sistema em seus subsistemas e à conexão entre essas partes.**



# Visão Geral da UML

## Visão de Processo

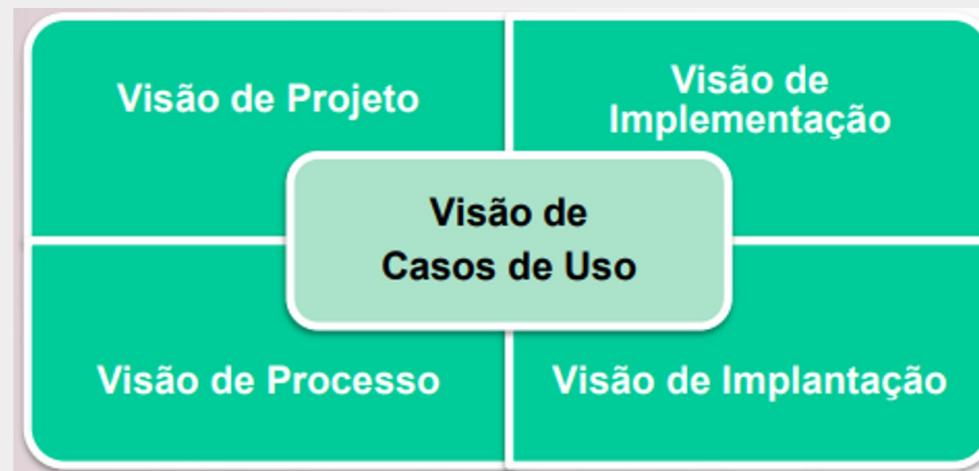
Esta visão enfatiza as características de concorrência (parallelismo), sincronização e desempenho do sistema.



# Visão Geral da UML

## Visão da UML

Dependendo das características do sistema, nem todas as visões precisam ser construídas.



# **Visão Geral da UML**

## **Visão da UML**

**Um processo de desenvolvimento que utilize a UML envolve a criação de diversos documentos.**

**Esses documentos podem ser textuais ou gráficos.**

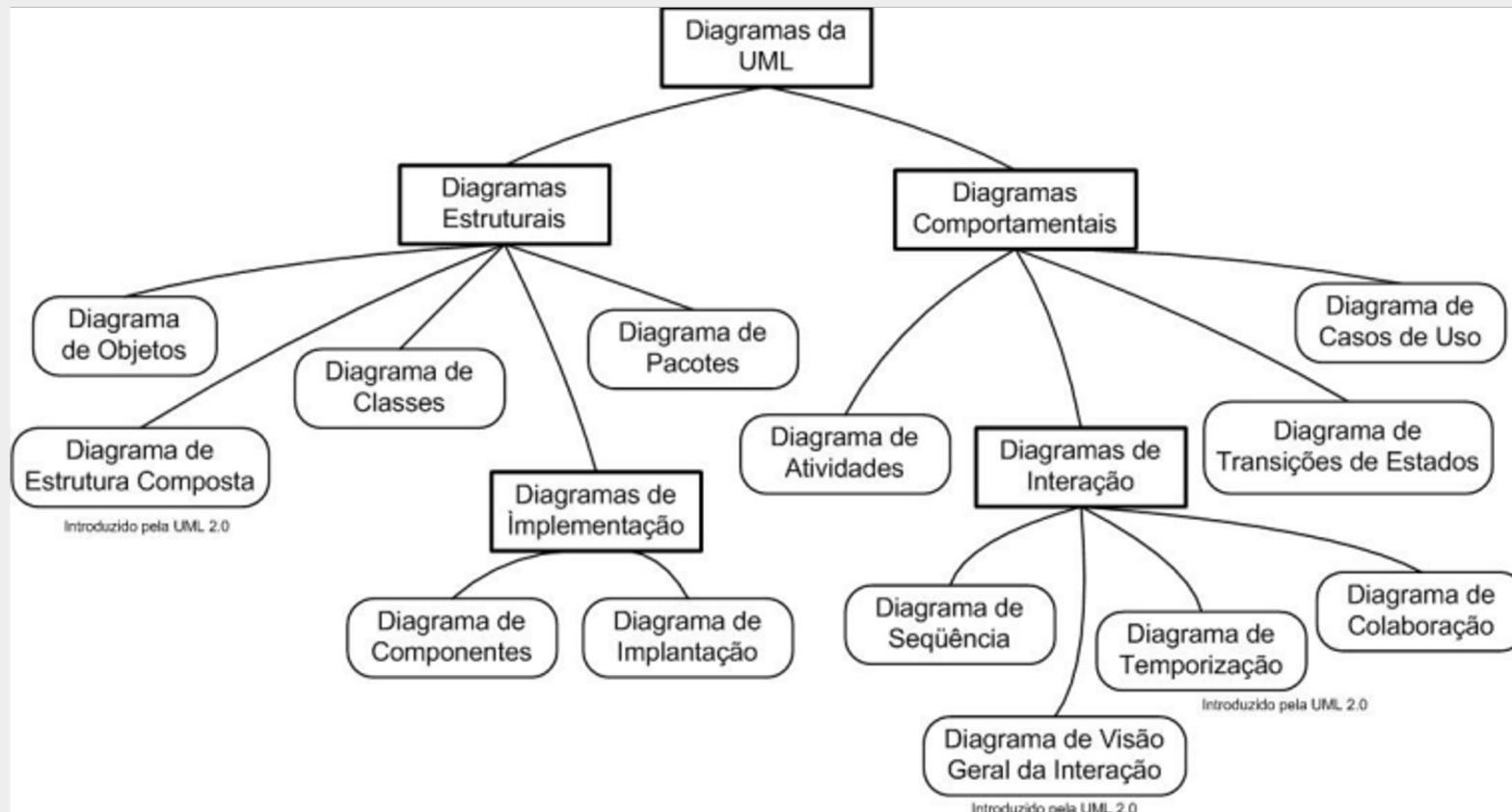
**Eles são denominados artefatos de software, ou simplesmente artefatos.**

**São os artefatos que compõem as visões do sistema.**



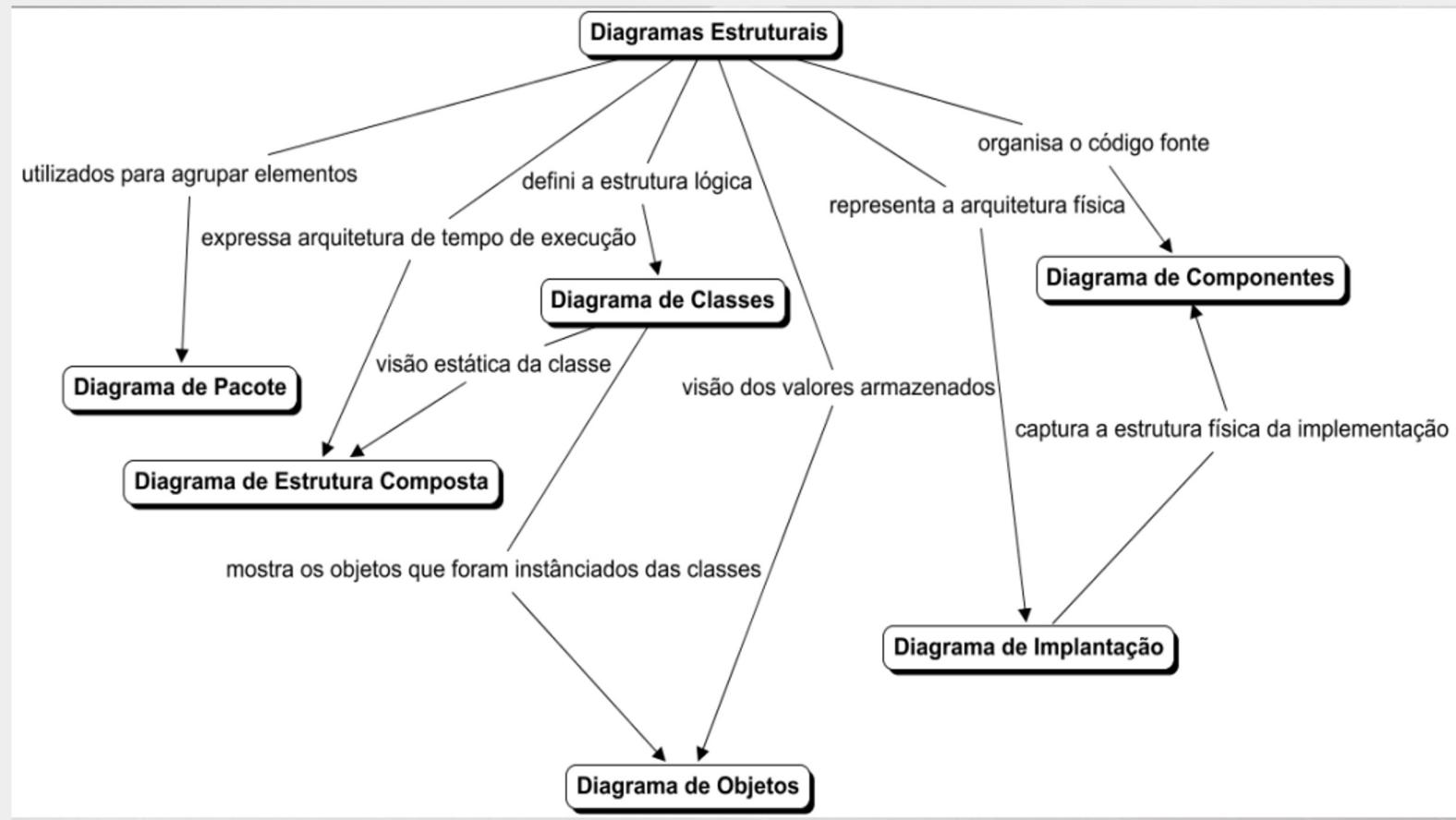
# Visão Geral da UML

## Artefatos da UML



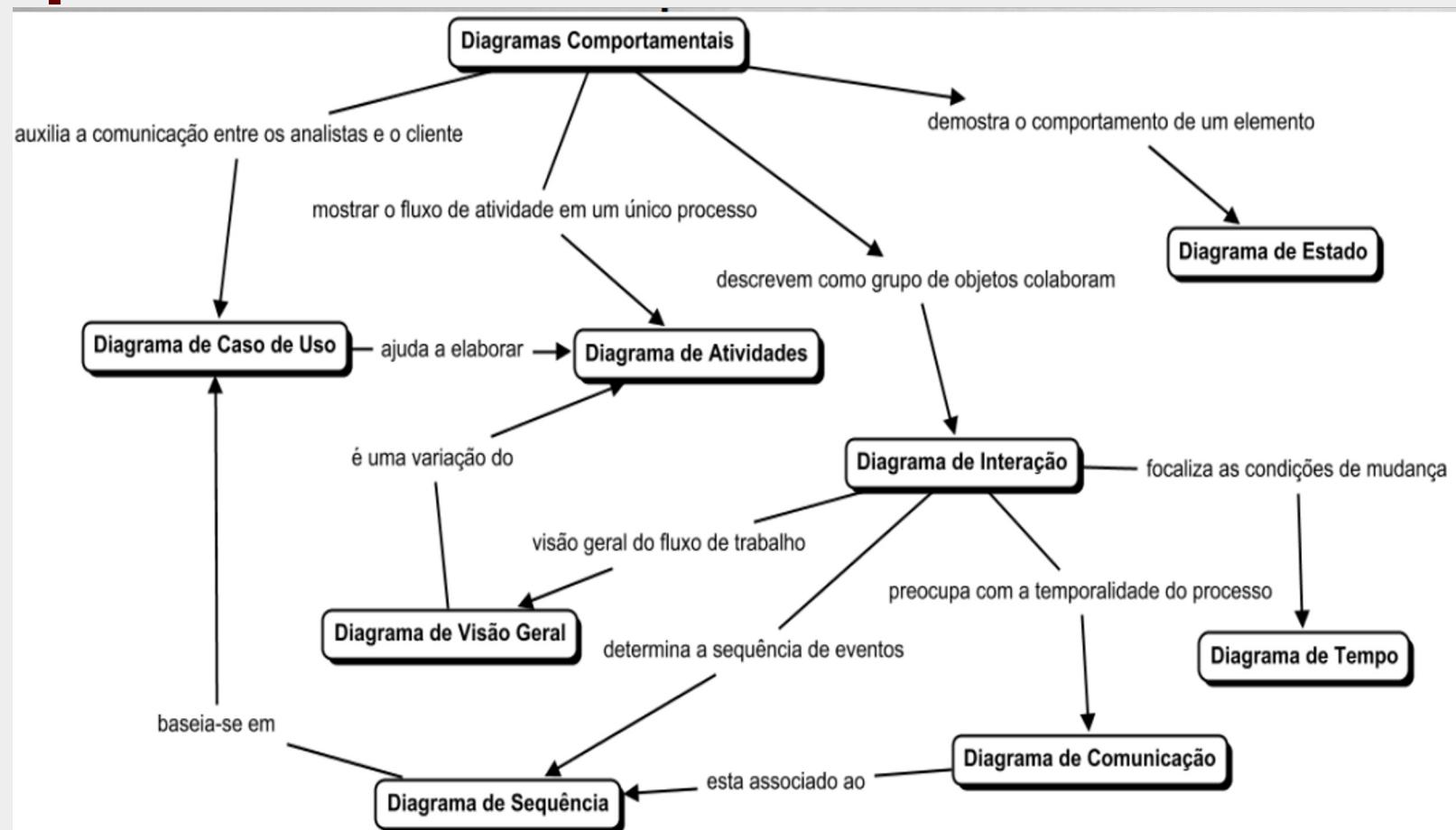
# Visão Geral da UML

## Mapa Conceitual - Artefatos da UML



# Visão Geral da UML

## Mapa Conceitual - Artefatos da UML



# A Linguagem UML Site, Certificações e o futuro

<https://www.uml.org/>



# Análise e Projetos de Sistemas

**Aula:  
Modelo de Caso de Uso**



Prof. Anderson Augusto Bosing

**Sistemas não existem de forma  
isolada. Existem interações com  
humanos ou máquinas.**



# **Modelagem de Caso de Uso - Histórico**

- Foi originalmente desenvolvida por Jacobson et al. (1993).
- Foi incorporada ao primeiro release da UML (RUMBAUGH et al., 1999).



# **Modelo de Casos de Uso(MCU)**

**O MCU representa os possíveis usos de um sistema da maneira como são percebidos por um observador externo a este sistema.**

**Cada um desses usos está associado a um ou mais requisitos funcionais identificados para o sistema.**

**É um modelo de análise que representa um refinamento dos requisitos funcionais.**

**Possui diversos componentes: casos de uso, atores e relacionamentos entre eles.**



# O que é Modelagem de Caso de Uso?

- Faz a ligação entre as necessidades dos envolvidos para os requisitos de software.
- Define limites claro para um sistema.
- Captura e comunica o comportamento desejado do sistema.
- Identifica quem ou o que interage com o sistema.
- Valida/Verifica os requisitos.
- Um instrumento de planejamento.



# O que é um Caso de Uso ?

**Um caso de uso define uma sequência de ações a serem executadas pelo sistema para produzir um resultado de valor observável para um ator.(RUP)**

**Usamos casos de uso para captar os comportamentos pretendidos de um sistema, sem especificar como esse comportamento é implementado.**



# **Casos de Uso são comumente identificados por nomes ou identificadores.**

**Todo caso de uso possui um nome que o identifica e diferencia dos demais casos de uso do sistema.**

**O nome é uma sequencia de caracteres de texto e deve ser único no pacote que o contém.**

**No geral, os nomes são expressões verbais ativas, que nomeiam um comportamento específico do sistema.**

**Exemplos: Registrar Venda, Fazer Pedido, Manter Usuários, Manter Produtos.**



# **Casos de Uso são comumente identificados por nomes ou identificadores.**

**Os identificadores representam através de um código o caso de uso a que se referenciam.**

**Exemplo:**

**UC.001 - Registrar Venda**

**UC.002 - Fazer Pedido**

**UC.003 - Manter Usuários**

**UC.004 - Manter Produtos**



# O que é um Caso de Uso ?

Um caso de uso é a especificação de uma sequência completa de interações entre um sistema e um ou mais agentes externos a esse sistema.

Representa um relato de uso de certa funcionalidade do sistema em questão, sem revelar a estrutura e o comportamento internos desse sistema.

# O que é um Caso de Uso ?

Cada caso de uso de um sistema se define pela descrição narrativa das interações que ocorrem entre o elemento externo e o sistema.

A UML não define uma estrutura textual a ser utilizada na descrição de um caso de uso.

Há vários estilos de descrição propostos para definir casos de uso.



# Casos de Uso contém Requisitos de Software

**Cada caso de uso:**

- Descreve ações que o sistema faz para entregar algo de valor para um agente.
- Mostra a funcionalidade do sistema que o agente usa.
- Modela um diálogo entre o sistema e os agentes.
- É um completo e significativo fluxo de eventos da perspectiva de um agente em particular.



# Benefícios dos Casos de Uso

- Dá um contexto para os requisitos

Coloca os requisitos do sistema em sequências lógicas.

Ilustra o porque o sistema é necessário.

Ajuda a verificar se todos os requisitos foram capturados.

- São fáceis de entender.

Usam terminologia que clientes e usuários entendem.

Descreve a história concreta de uso do sistema.

Verifica o entendimento dos envolvidos.

- Facilita o acordo com os clientes.

- Facilita o reuso: teste, documentação e design.



# Elementos do Diagrama de Caso de Uso



**“Um caso de uso é a especificação de uma sequência completa de interações entre um sistema e um ou mais agentes externos a esse sistema.”**

**O que são os agentes externos? Onde vivem? Do que se alimentam ?**



# Atores

**Qualquer elemento externo ao sistema que interage com o mesmo é denominado ator.**

**“Externo”** nessa definição indica que atores não fazem parte do sistema.

**“Interage”** significa que um ator troca informações com o sistema (envia ou recebe informações).



# Atores

Podem ser Categorizados em:

Um **ator humano** é uma pessoa física, que no diagrama deve possuir como nome o papel que a pessoa executa no contexto empresarial onde o sistema será utilizado. Por exemplo: Cliente, Fornecedor, Atendente.

Um **ator sistêmico** é um sistema, ou módulo de um sistema, ou componente de um sistema, que realizará a execução da funcionalidade que está especificada pelo caso de uso. No diagrama deve possuir seu nome de fato (se o ator é o sistema “Disparador de rotinas batch” por exemplo, este deve ser o nome do ator sistêmico).

# Atores

**Exemplos de Atores:**

- **Cargos (Ex. Empregado, Cliente, Gerente, Vendedor).**
- **Organizações ou divisões de uma organização (Ex. Fornecedor, Administradora de Cartões, Almoxarifado ).**
- **Outros sistemas de software (Ex. Sistema de Cobrança, Sistema de Estoque de Produtos).**
- **Equipamentos com os quais o sistema deve se comunicar (Ex. Leitora de Código de Barras, Sensor).**



# Atores

**Um ator corresponde a um papel representado em relação ao sistema.**

**Exemplos:**

O mesmo indivíduo pode ser o Cliente que compra mercadorias e o Vendedor que processa vendas.

Uma pessoa pode representar o papel de Funcionário de um banco que realiza a manutenção de um caixa eletrônico, mas também pode ser o Cliente do banco que realiza o saque.



# Atores

É uma boa prática de modelagem fazer com que o nome dado a esse ator lembre o seu papel, em vez de lembrar quem o representa.

**Exemplos de bons nomes para atores:**

✓ Cliente, Estudante, Fornecedor etc.

**Exemplos de maus nomes para atores:**

✓ João, Fornecedor, ANVISA etc.



# Atores

**Um ator pode participar de muitos casos de uso (situação muito comum na prática).**

**Do mesmo modo, um caso de uso pode envolver a participação de vários atores, nesses casos, os atores podem ter duas classificações:**

**Primários ou Secundários.**



# Atores

**Um ator primário é aquele que inicia uma sequência de interações de um caso de uso.**

**São eles os agentes externos para os quais o caso de uso traz benefício direto.**

**As funcionalidades principais do sistema são definidas tendo em mente os objetivos dos atores primários.**



# Atores

**Já um ator secundário supervisiona, opera, mantém ou auxilia na utilização do sistema pelo atores primários.**

**Atores secundários existem apenas para que os atores primários possam utilizar o sistema.**



# Atores

**Exemplo:**

**Em um navegador de internet.**

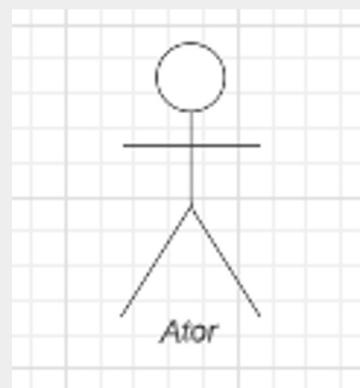
**Para que o Usuário (ator primário) requisite uma página ao programa, outro ator (secundário) está envolvido: o Servidor Web.**

**O ator primário Usuário é auxiliado pelo secundário, Servidor Web, uma vez que é através deste último que o primeiro consegue alcançar seu objetivo.**



# Atores

**Notação de um Ator:**



# Caso de Uso

**Um caso de uso define uma sequência de ações a serem executadas pelo sistema para produzir um resultado de valor observável para um ator.(RUP)**

**Usamos casos de uso para captar os comportamentos pretendidos de um sistema, sem especificar como esse comportamento é implementado.**



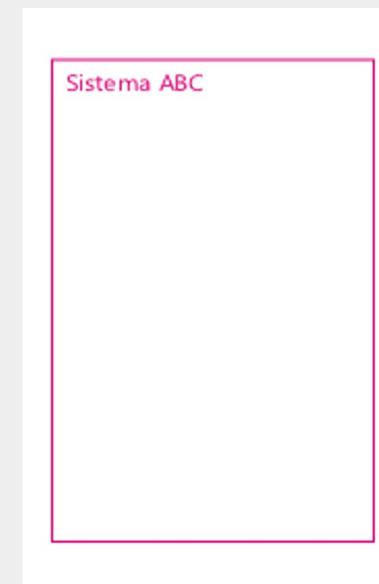
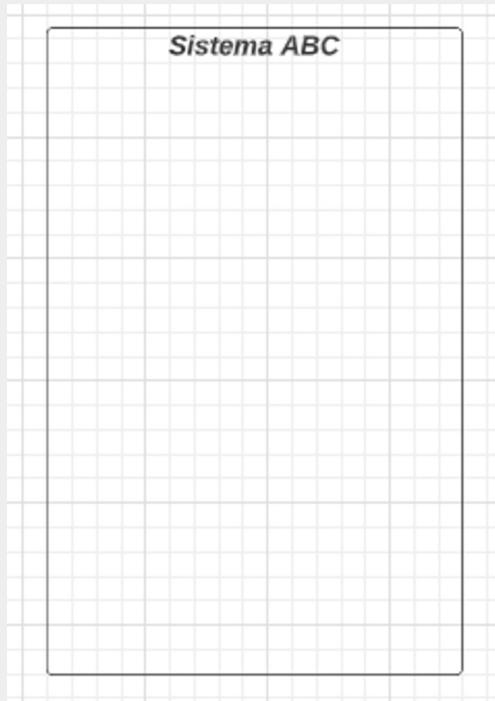
# Caso de Uso

Notação de um Caso de Uso:



# Sistema

**Representa a fronteira do sistema ou componentes de sistemas e subsistemas.**



# **Como os Atores e Casos de Uso se relacionam ?**

**Um ator deve estar relacionado a um ou mais casos de uso do sistema.**

**Além disso, pode haver relacionamentos entre os casos de uso ou entre os atores de um sistema.**

**A UML define os seguintes relacionamentos para o modelo de casos de uso: comunicação, inclusão, extensão e generalização.**



# **Associação de Comunicação**

**Um relacionamento de comunicação informa a que caso e uso o ator está associado.**

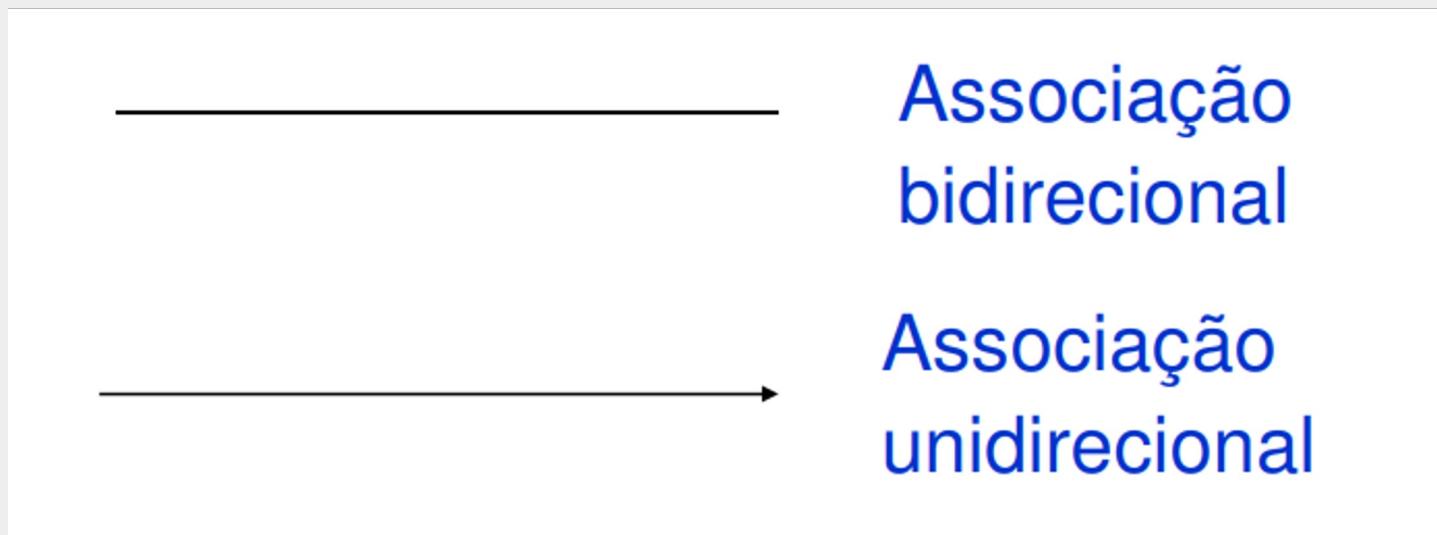
**Isso significa que esse ator interage (troca informações) com o sistema com ajuda daquele caso de uso.**

**O relacionamento de comunicação é, de longe, o mais comumente utilizado de todos.**



# Associações

Notação de um Caso de Uso:



# **Relacionamento de inclusão.**

**Na modelagem UML, um relacionamento de inclusão é aquele no qual um caso de uso (o caso de uso base) inclui a funcionalidade de outro caso de uso (o caso de uso de inclusão). O relacionamento de inclusão suporta a reutilização da funcionalidade em um modelo de caso de uso.**

**O relacionamento de inclusão existe somente entre casos de uso. Quando dois ou mais casos de uso incluem uma sequência comum de interações, essa sequência comum pode ser descrita em outro caso de uso.**

**A partir daí, vários casos de uso do sistema podem incluir o comportamento desse caso de uso comum.**



# **Relacionamento de inclusão.**

**Exemplo: Sistema bancário**

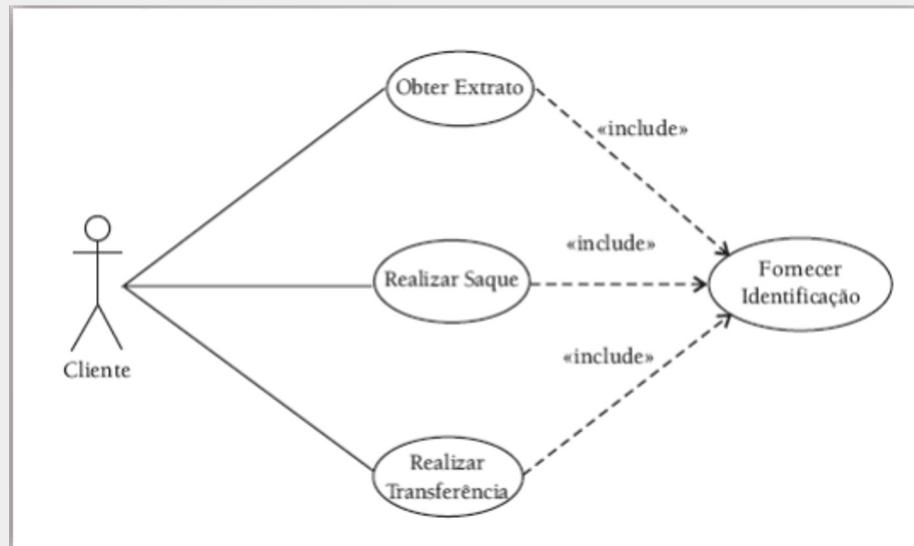
**Esses casos de uso têm uma sequência de interações em comum: a que valida a senha do cliente do banco.**

**Essa sequência de interações em comum pode ser descrita em um caso de uso Fornecer Identificação.**

**Alguns casos de uso desse sistema são Obter Extrato, Realizar Saque e Realizar Transferência.**



# Relacionamento de inclusão.



# **Relacionamento de extensão.**

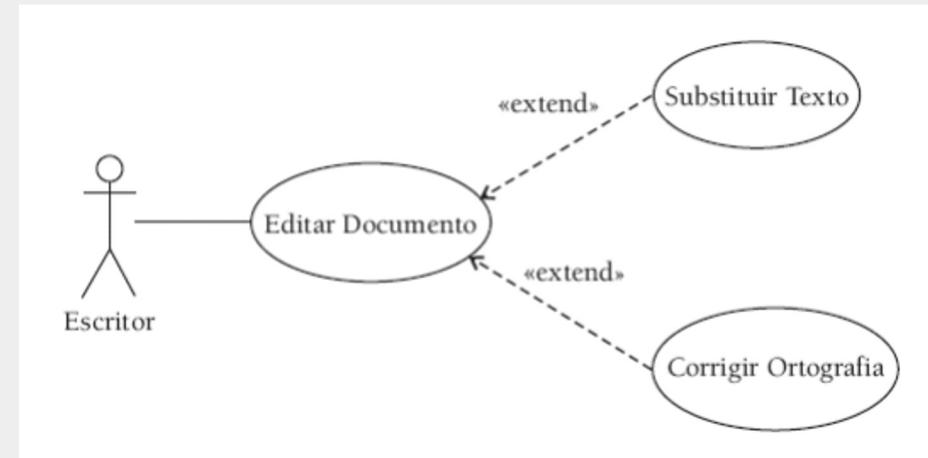
**Na modelagem UML, é possível utilizar um relacionamento de extensão para especificar que um caso de uso (extensão) estende o comportamento de outro caso de uso (base). Esse tipo de relacionamento revela detalhes sobre um sistema ou aplicativo que normalmente estão ocultos em um caso de uso.**

**Cada uma dessas diferentes sequências representa um comportamento eventual, ou seja, um comportamento que só ocorre sob certas condições, ou cuja realização depende da escolha do ator.**



# Relacionamento de extensão.

Mostra que os casos de uso Corrigir Ortografia e Substituir Texto têm sequências de interações que são eventualmente utilizadas quando o ator Escritor estiver usando o caso de uso Editar Documento

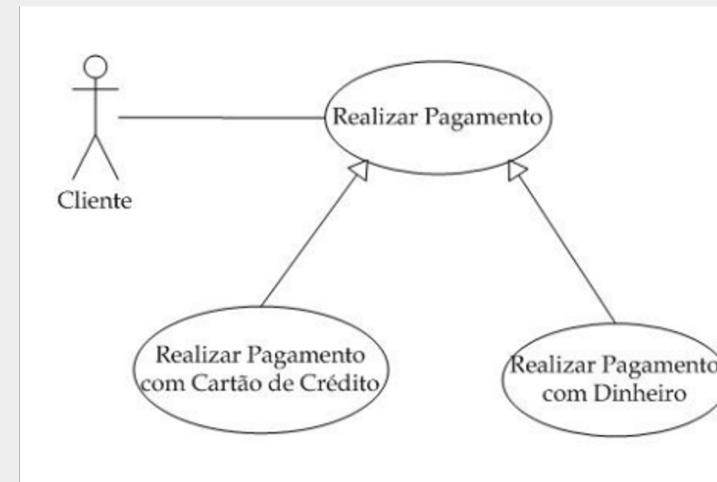


Fonte: Bezerra (2007)

# Relacionamento de generalização.

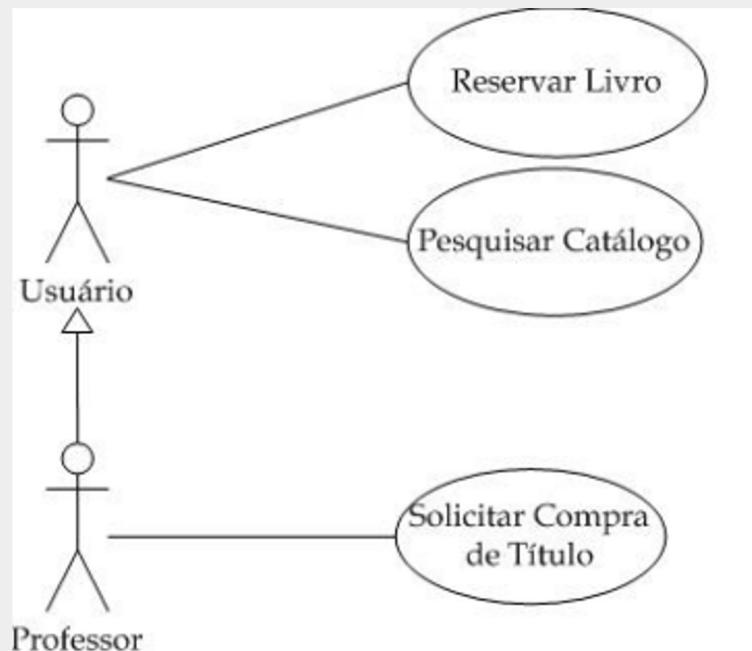
**Na modelagem UML, um relacionamento de generalização é aquele no qual um elemento de modelo (o filho) tem como base outro elemento de modelo (o pai).** Os relacionamentos de generalização são utilizados em diagramas de classe, componente, implementação e caso de uso para indicar que o filho recebe todos os atributos, operações e relacionamentos definidos no pai.

**Realizar Pagamento com Cartão de Crédito e Realizar Pagamento com Dinheiro herdam características em relação ao caso de uso Realizar Pagamento, especializando o seu comportamento.**

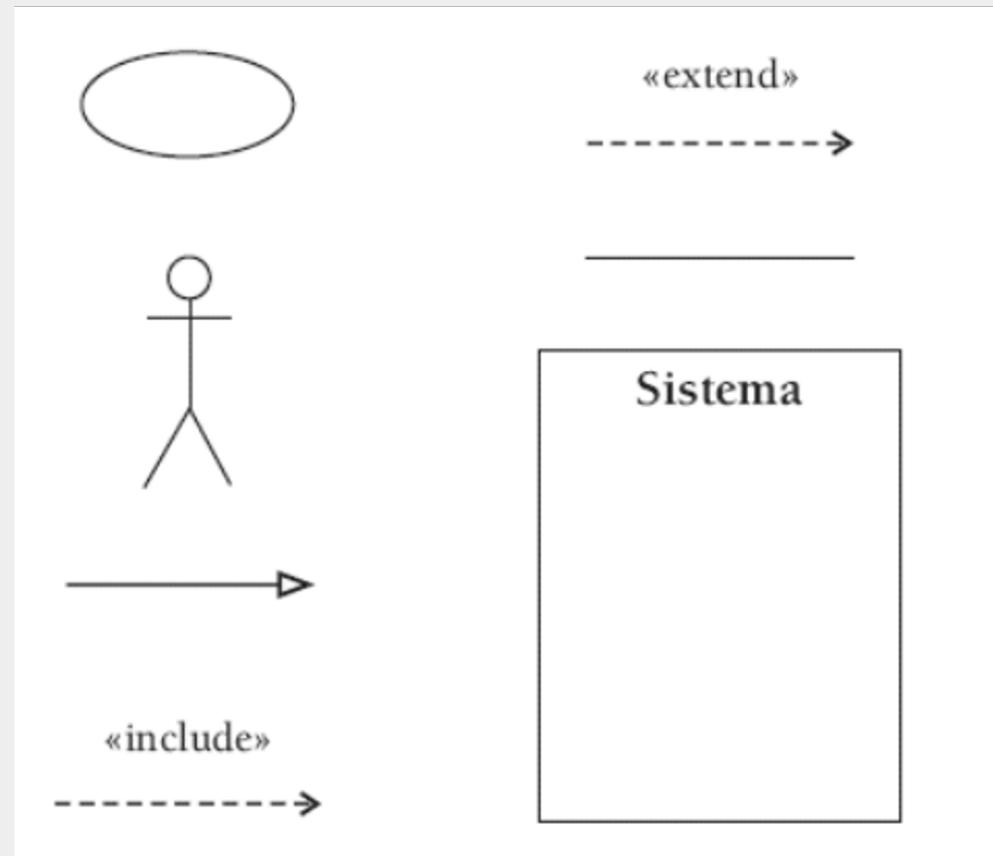


# Relacionamento de generalização.

A generalização entre os atores Usuário e Professor indica que este último pode interagir com qualquer caso de uso que um usuário comum interage.



# Resumo das Notações



# Vamos Praticar?

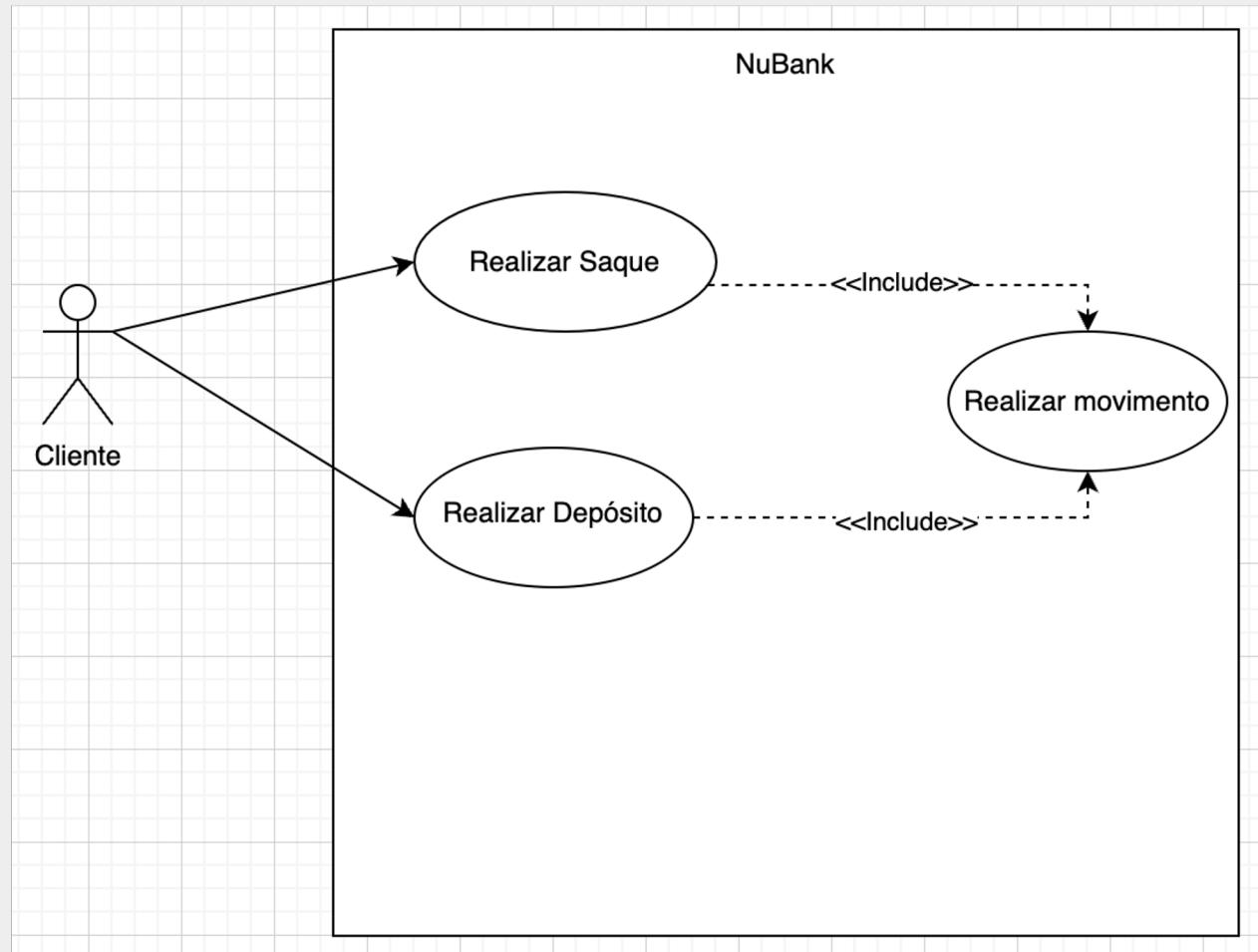
Considerando o seguinte cenário:

Um cliente vai até o banco realizar um saque ou um depósito e ambas as operações devem registrar movimentos.

<https://online.visual-paradigm.com/>



# Vamos Praticar?



**Além dos conceitos e dos elementos do MCU, tem mais alguma coisa que precisamos conhecer?**



**Tão importante quanto dominarmos a notação do MCU é termos conhecimento de técnicas e boas práticas de modelagem.**

**Se seguidas e utilizadas, nos levam à construção de modelos coerentes com as reais necessidades dos futuros usuários.**



**Atores e os casos de uso são identificados a partir de informações coletadas no levantamento de requisitos (analistas devem identificar as atividades do negócio relevantes ao sistema a ser construído).**

**Não há uma regra geral que indique quantos casos de uso e atores são necessários para descrever um sistema.**

**A quantidade de casos de uso e atores depende da complexidade do sistema.**



**“Um ator é todo elemento externo que interage com o sistema.”**

**Atores em potencial: Fontes e os destinos das informações a serem processadas.**

**O analista deve identificar:**

- ✓ As áreas da empresa que serão afetadas ou utilizarão o sistema.**
  
- ✓ Fontes de informações a serem processadas e os destinos das informações geradas pelo sistema.**



## **Perguntas úteis:**

- » Que órgãos, empresas ou pessoas (cargos) irão utilizar o sistema?
- » Que outros sistemas irão se comunicar com o sistema?
- » Alguém deve ser informado de alguma ocorrência no sistema?
- » Quem está interessado em um certo requisito funcional do sistema?

**Devemos continuar pensando sobre atores quando passar para a identificação dos casos de uso.**



**A partir da lista (inicial) de atores, deve-se passar à identificação dos casos de uso.**

**Nessa identificação, pode-se distinguir entre dois tipos de casos de uso:**

- ✓ Primário:** representa os objetivos dos atores.
- ✓ Secundário:** aquele que não traz benefício direto para os atores, mas que é necessário para que sistema funcione adequadamente.



## **Perguntas úteis:**

- » Quais são as necessidades e objetivos de cada ator em relação ao sistema?**
- » Que informações o sistema deve produzir?**
- » O sistema deve realizar alguma ação que ocorre regularmente no tempo?**
- » Para cada requisito funcional, existe um (ou mais) caso(s) de uso para atendê-lo?**



- » Caso de uso “oposto”: Deve-se perguntar para cada caso de uso: “As ações realizadas pelo sistema quando da realização deste caso de uso podem ser desfeitas?”.
- » Caso de uso que precede outro caso de uso: Por exemplo, para que um cliente realize uma compra, é necessário que ele esteja cadastrado no sistema. A pergunta a ser feita para cada caso de uso, “o que pode ocorrer antes da realização deste caso de uso?”.

- » Caso de uso que sucede a outro caso de uso: A pergunta geral que se deve é “o que pode ocorrer após a realização deste caso de uso?”.
- » Caso de uso temporal: Por exemplo: “O sistema deve gerar um relatório de vendas toda sexta-feira”. A pergunta geral nessa situação é: “Há alguma tarefa que o sistema deva realizar automaticamente?”.



» Caso de uso relacionado a alguma condição interna:

**Seguem dois exemplos disso: “O sistema deve notificar o usuário de que há novas mensagens de correio”;**

**“O sistema deve avisar o almoxarife de que um determinado produto chegou no nível de estoque mínimo”.**



## Categorias:

- ✓ Manutenção de cadastros;
- ✓ Manutenção de usuários;
- ✓ Gerenciamento de acesso;
- ✓ Manutenção de informações provenientes de outros sistemas.

O sistema de software não existe para cadastrar informações, nem tampouco para gerenciar os usuários.

O objetivo principal de um sistema é agregar valor ao ambiente no qual ele está implantado.



# **Inclusões, Alterações, Exclusões e Consultas são casos de uso?**

**Para cada objeto do sistema, seria necessário  
considerar três casos de uso?**

**Exemplo:**

**Incluir Livro, Alterar dados do livro e Excluir Livro**  
**E quanto às consultas, por exemplo, Consultar Livro  
por Autor, Consultar Livro por Título, etc.,**

**Seriam casos de uso?**



# **Inclusões, Alterações, Exclusões e Consultas são casos de uso?**

**Não há consenso sobre isso.**

**Como são processos em geral muito simples, com lógica conhecida, não vamos detalhá-los como faremos com os casos de uso mais complexos.**

**No entanto, incluiremos no diagrama de casos de uso, para dar uma noção geral do escopo e tamanho do sistema.**

**Ex:**

**Manter Usuários, Manter Clientes.**

