

Informatica 7

Escola Mascarenhas de Moraes

Introdução

- Sistema de informação
 - Produtos de software cada vez mais requisitados
 - São compostos por muitos elementos de processamento, armazenamento, distribuição, organização, etc.
- Por conta da complexidade de projetos de software é necessário utilizar métodos, técnicas e ferramentas para aumentar a efetividade em obter altos níveis de qualidade com custos baixos

Introdução

- Engenharia de Software: área de computação que aborda a construção de sistemas de informação como um produto de engenharia.

O Início da Engenharia de Software

- Antes de 1961
 - Altos custos de hardware escondiam o custo do software
 - Sistemas eram simples e construídos por pequenas equipes
 - Sistemas construídos para resolver problemas específicos

O Início da Engenharia de Software

- Depois de 1961
 - Surgimento de computadores mais modernos, com mais poder computacional
 - Hardware baixou de preço
 - Software ganha notoriedade
 - O “amadorismo” que era utilizado na construção de software ficou evidente.

A Crise do Software

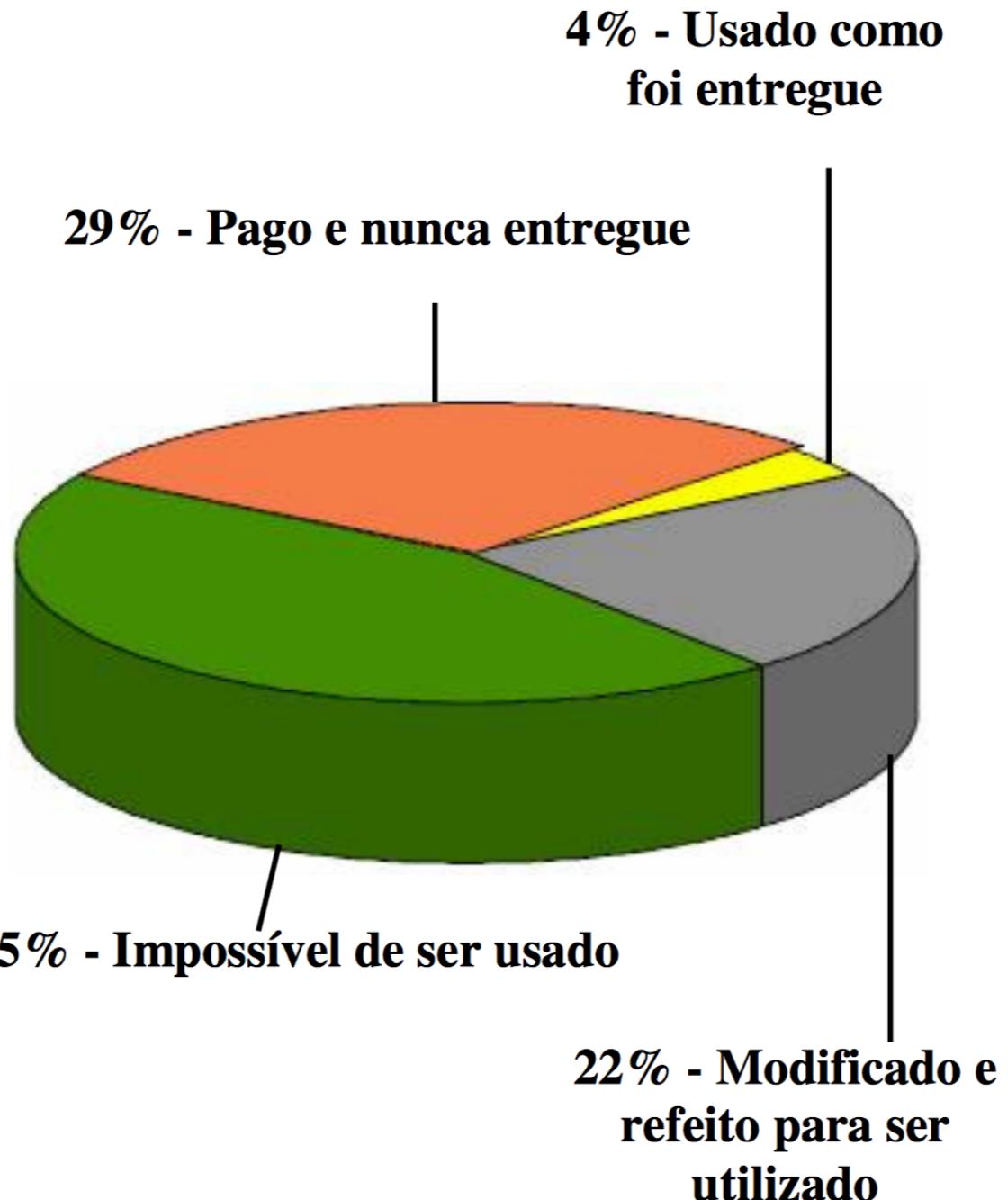
- Em 1968
- O termo “A Crise do Software” expressava as dificuldades do desenvolvimento de software frente ao rápido crescimento da demanda existente, da complexidade dos problemas a serem resolvidos e da inexistência de técnicas estabelecidas para o desenvolvimento de sistemas que funcionassem adequadamente ou pudessem ser validados.

A Crise do Software

- Para resolver estes problemas aconteceu uma conferência chamada: “NATO Software Engineering Conference”
- Marcou o início de uma nova era da computação

A Crise do Software

- O que realmente foi (ou é) a Crise de Software?
 - Em resumo: a imaturidade no desenvolvimento de software, que causou problemas como:
 - Projetos estourando o orçamento
 - Projetos estourando o prazo
 - Software de baixa qualidade
 - Software muitas vezes não atendendo os requisitos
 - Projetos não gerenciáveis e códigos difíceis de manter

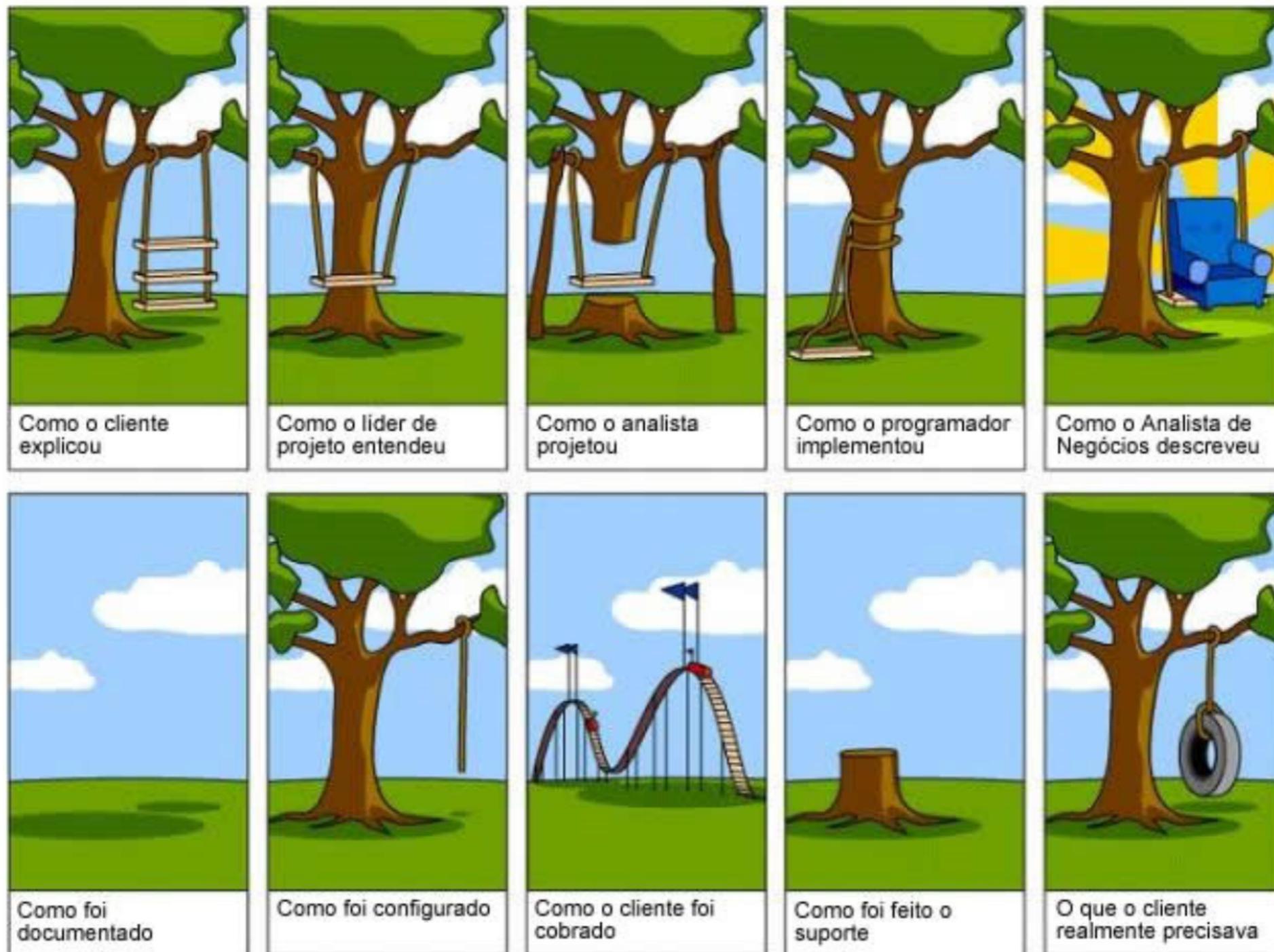


Situação na época da crise de software

- Mesmo com a evolução das técnicas os números não mudaram tanto
 - Dados do STANDISH GROUP de 2004
 - Base de 8000 projetos
 - 29% dos projetos com sucesso
 - 18% dos projetos cancelados
 - 53% falha
 - Prazo (Média de atraso de 222%)
 - Custo (Média de custo de 189%)
 - Requisitos (61% das funcionalidades originais incluídas)

A Crise do Software

- Será que a crise acabou?



Um grande problema a ser superado é a falha de comunicação entre as partes envolvidas em um projeto de software

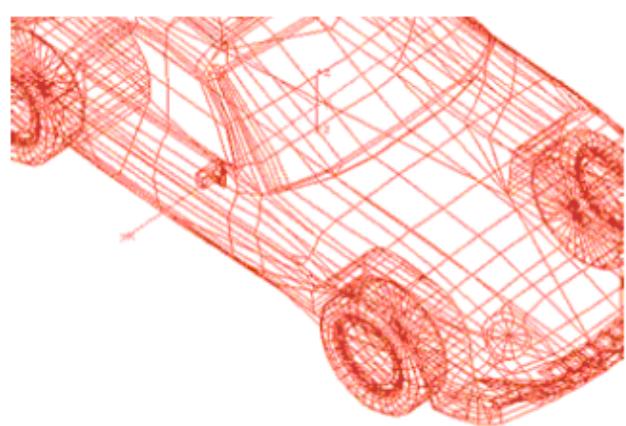
Engenharia de Software

- Problemas (e com muita frequência) acontecem nos projetos de software
- Processos, métodos e ferramentas auxiliam muito no desenvolvimento de um projeto.
- Devem ser aplicados por pessoas com o devido conhecimento
- Utilizando CORRETAMENTE a Engenharia de Software as chances de sucesso de um projeto são grandes.
- A Crise acabou? Sim, mas somente para aqueles que utilizam a engenharia de software

Engenharia de Software

- O que é Engenharia de Software?

Criando uma analogia com o processo de fabricação de automóveis

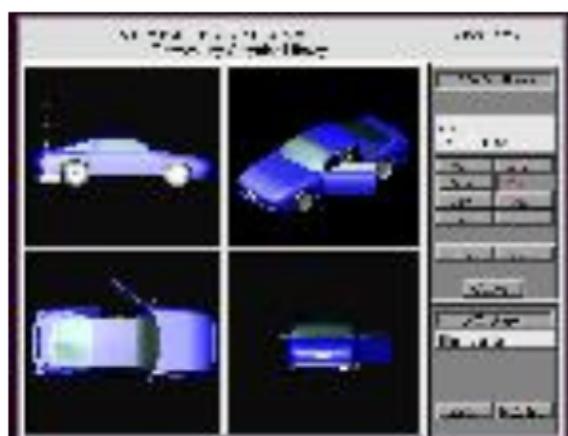


Requisitos

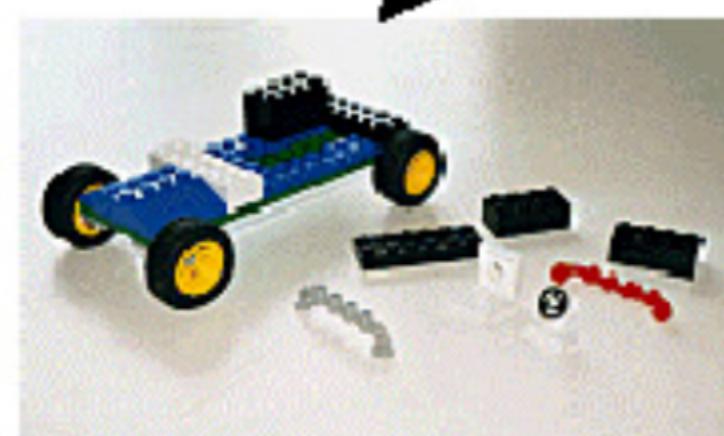
Unidades



Integração



**Projeto e
Simulações**



Integração



**Produto
Concluído**

Analogia

- Construção inicia-se a partir do surgimento da ideia de construir um carro
- A partir dai, levanta-se as características, necessidade e desejos relacionados ao produto
 - Isso compõe a especificação de requisitos do carro
 - Exemplos: Capacidade do porta malas, potência do motor, câmbio automático, etc.

Analogia

- Utilizando os requisitos iniciais, inicia-se uma prototipação
- O modelo criado é utilizado para responder diversas questões
- A partir dai, desenvolve-se as partes que vão compor o automóvel
 - Essas partes são chamadas de unidades
 - Exemplo: Direção, Portas, Motor, Rodas
 - Cada um desses componentes possui especificações que necessitam ser verificadas

Analogia

- Fase de integração: junta-se todos os componentes e realiza-se mais uma verificação
- Testes finais

Analogia

- Processo de construção de um carro tem muito em comum com a Engenharia de Software
 - Levantamento de Requisitos
 - Construção de modelos
 - Implementar unidades
 - Verificação
 - Integração
 - Testes

Engenharia de Software

- Engenharia de Software é a aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento de software.

Engenharia de Software

- Segundo a literatura:
 - *"O estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais.",* NAUR PETER, 1969.
 - *"A aplicação prática do conhecimento científico para o projeto e a construção de programas computacionais e a documentação necessária à sua operação e manutenção.",* BARRY BOEHM, 1976.
 - *"Conjunto de métodos, técnicas e ferramentas necessárias à produção de software de qualidade para todas as etapas do ciclo de vida do produto.",* SACHA KRAKOWIAK, 1985.

Engenharia de Software

- De modo mais objetivo, a engenharia de software busca prover tecnologia e conhecimento necessário para produzir software de alta qualidade e baixo custo.

Problemas

- Um grande problema facilmente percebido hoje é justamente o fato de boa parte das organizações não encararem o desenvolvimento de software como um projeto de verdade, aplicando as técnicas, métodos e ferramentas necessárias.

Problemas no Desenvolvimento de Software

- A crise continua em muitos lugares, pela falta de uso das ferramentas, métodos e técnicas existentes

Processos



Pessoas



Tecnologia



Tripé da Engenharia de Software

- Não adianta ter os melhores profissionais do mundo se não possuímos boas tecnologias ou se não possuímos um processo que guie o desenvolvimento de software
- Não adianta possuir boas tecnologias se não há profissionais capacitados para utilizá-las de forma devida.
- Não adianta possuir os melhores profissionais e melhores tecnologias sem o processo para guiar as atividades.

Mitos sobre o Software

- Muitas causas de problemas no desenvolvimento de software são provenientes de mitos que acabaram se espalhando com o tempo
- **Mitos de Gerenciamento**
- **Mitos do Cliente**
- **Mitos do Profissional**

Mitos de Gerenciamento

Mito 1. "Se a equipe dispõe de um manual repleto de padrões e procedimentos de desenvolvimento de software, então a equipe será capaz de conduzir bem o desenvolvimento."

Realidade 1. Isso não é o suficiente! É preciso que a equipe aplique efetivamente os conhecimentos apresentados no manual. É necessário que o manual reflita a moderna prática de desenvolvimento de software e que este seja exaustivo com relação a todos os problemas de desenvolvimento que poderão aparecer no percurso.

Mitos de Gerenciamento

Mito 2. "A equipe tem ferramentas de desenvolvimento de software de última geração, uma vez que eles dispõem de computadores modernos."

Realidade 2. Ter à sua disposição o último modelo de computador pode ser bastante confortável para o desenvolvedor do software, mas não oferece nenhuma garantia quanto à qualidade do produto desenvolvido. Mais importante do que ter um hardware de última geração é ter ferramentas para a automação do desenvolvimento de software e sabê-las utilizar adequadamente.

Mitos de Gerenciamento

Mito 3. "Se o desenvolvimento do software estiver atrasado, aumentando a equipe poderemos reduzir o tempo de desenvolvimento."

Realidade 3. Acrescentar pessoas em um projeto atrasado provavelmente vai atrasá-lo ainda mais. De fato, a introdução de novos profissionais numa equipe em fase de condução de um projeto vai requerer uma etapa de treinamento dos novos elementos da equipe; para isto, serão utilizados elementos que estão envolvidos diretamente no desenvolvimento, o que vai, consequentemente, implicar em maiores atrasos no cronograma.

Mitos do Cliente

Mito 4. "Uma descrição breve e geral dos requisitos do software é o suficiente para iniciar o seu projeto. Maiores detalhes podem ser definidos posteriormente."

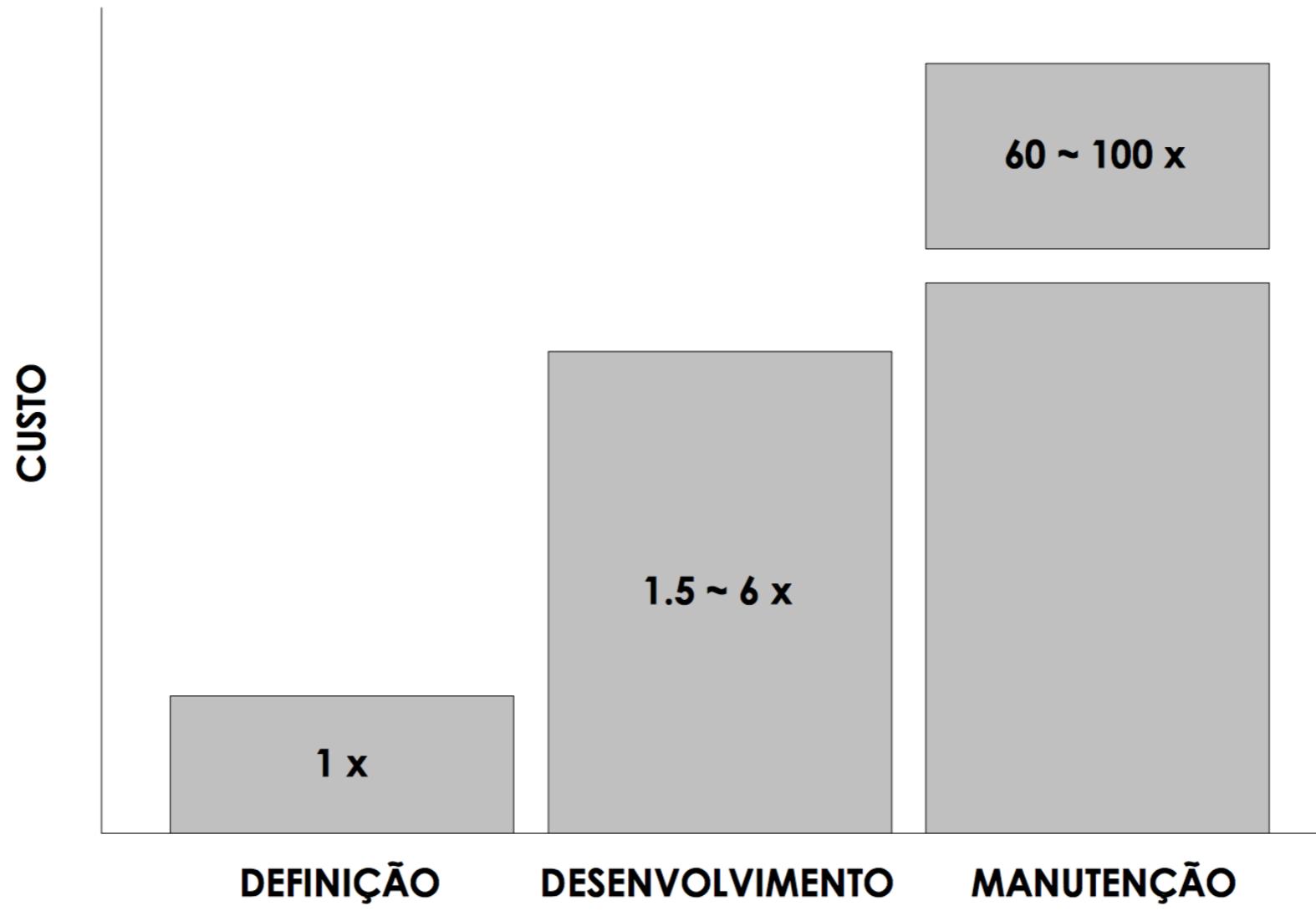
Realidade 4. Este é um dos problemas que podem conduzir um projeto ao fracasso, o cliente deve procurar definir o mais precisamente possível todos os requisitos importantes para o software: funções, desempenho, interfaces, restrições de projeto e critérios de validação são alguns dos pontos determinantes do sucesso de um projeto. O “deixar pra depois” pode simplesmente não acontecer, a não ser em casos previstos pelos processos ágeis em que os clientes estão sempre presente e dentro da organização desenvolvedora. No entanto, é sabido que essa prática é uma das mais difíceis de serem seguidas...

Mitos do Cliente

Mito 5. "Os requisitos de projeto mudam continuamente durante o seu desenvolvimento, mas isto não representa um problema, uma vez que o software é flexível e poderá suportar facilmente as alterações."

Realidade 5. É verdade que o software é flexível (pelo menos mais flexível do que a maioria dos produtos manufaturados). Entretanto, não existe software, por mais flexível que suporte alterações de requisitos significativas sem um adicional em relação ao custo de desenvolvimento. O fator de multiplicação nos custos de desenvolvimento do software devido a alterações nos requisitos cresce em função do estágio de evolução do projeto, como mostra a seguir.

Mitos do Cliente



Mitos do Professional

Mito 6. "Após a finalização do programa e a sua implantação, o trabalho está terminado."

Realidade 6. O que ocorre na realidade é completamente diferente disto. Segundo dados obtidos a partir de experiências anteriores, ilustrados no livro de Roger Pressman, 50 a 70% do esforço de desenvolvimento de um software é empregado após a sua entrega ao cliente (manutenção).

Mitos do Profissional

Mito 7. "Enquanto o programa não entrar em funcionamento, é impossível avaliar a sua qualidade."

Realidade 7. Na realidade, a preocupação com a garantia da qualidade do software deve fazer parte de todas as etapas do desenvolvimento. O teste, por exemplo, pode iniciar antes do produto atingir um estado funcional, a partir do planejamento dos casos de teste.

Mitos do Profissional

Mito 8. "O produto a ser entregue no final do projeto é o programa funcionando."

Realidade 8. O programa em funcionamento é um das componentes do software. Além do software em si, um bom projeto deve ser caracterizado pela produção de um conjunto importante de documentos. Um produto de software sem um manual de operação pode ser tão ruim quanto um software que não funciona!

Exercícios

1. Qual foi a principal causa do surgimento da Engenharia de software?

2. Quais eram os problemas associados à Crise do Software?

3. A crise do software realmente acabou? Comente sobre isso.

4. Faça uma comparação entre a Engenharia de Software e o desenvolvimento de um carro.

5. Defina Engenharia de Software.

6. Qual o tripé em que a Engenharia de Software está baseada?

7. O que é um sistema? Quais seus principais componentes?

8. É possível fazer a estimativa de custo e prazo para desenvolvimento de um software com apenas alguns poucos minutos de conversa?

Comente sobre isso relacionando sua resposta a outras áreas.

9. O que são os mitos do Software?

10. Cite alguns mitos relacionados ao gerenciamento, comentando a realidade relacionada aos mitos.

11. Cite alguns mitos relacionados aos clientes, comentando a realidade relacionada aos mitos.

12. Cite alguns mitos relacionados aos profissionais do desenvolvimento de software, comentando a realidade relacionada aos mitos.

Processos de Software

- Conforme comentado nos slides anteriores, a Engenharia de Software nada mais é que o tratamento do software como um produto, empregando dentro do processo de desenvolvimento todos os princípios da engenharia.
- Como todo produto, o software também possui um ciclo de vida, que pode ser definido como o conjunto de todas as etapas relacionadas à sua existência

Etapas do Processo de Software

- a concepção, onde o produto é idealizado, a partir da percepção de uma necessidade;
- o desenvolvimento, a partir da identificação dos requisitos e sua transformação em itens a serem entregues ao cliente;
- a operação, quando o produto é instalado para ser utilizado em algum processo de negócio, sujeito a manutenção, sempre que necessário;
- a retirada, quando o produto tem sua vida útil finalizada.

Etapas do Processo de Software

- A codificação do projeto, é apenas uma parte do ciclo de vida, embora muitos profissionais acreditem, erroneamente, que ela é a única tarefa relacionada a desenvolvimento de software

Processo X Projeto

- Processo é um guia para construir um produto
- Projeto é o uso de um processo para construir um produto específico
- Similar ao modelo de classes e objetos
 - Classe é um estrutura que abstrai um conjunto de objetos com características similares
 - Objeto é uma instancia dessa classe com valores específicas para cada característica
 - Sendo assim, um projeto é uma instanciação de um processo para a construção de um produto

Ciclos de Vida de Um Projeto

- O Processo de Software trata-se da formalização de como o produto de software deve ser construído, do início ao fim.
- Existem vários modelos de ciclo de vida de projetos

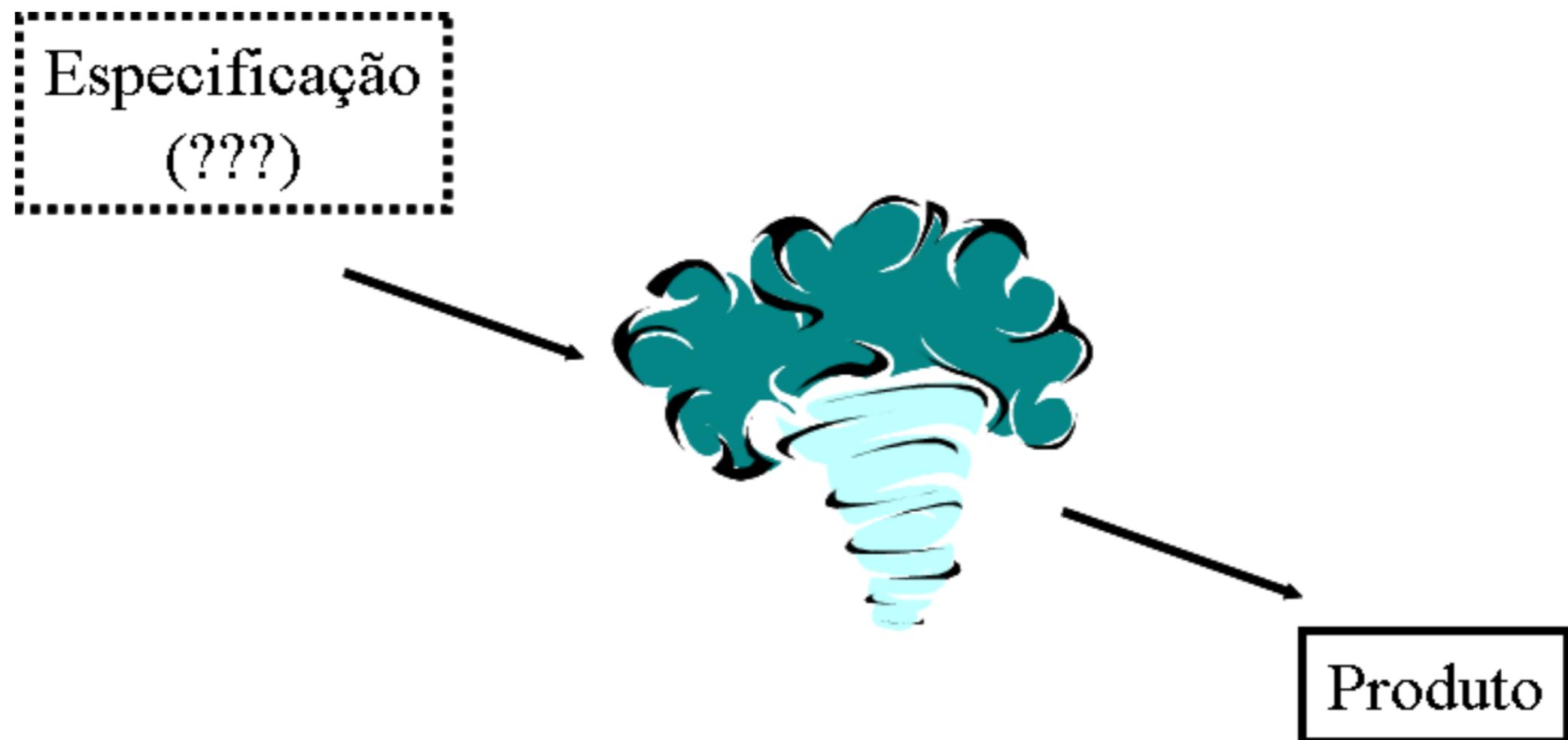
Ciclos de Vida de Um Projeto

- Etapas de um ciclo de projetos:
 - Requisitos: obtenção do enunciado, completo, claro e preciso dos desejos, necessidades, expectativas e restrições dos clientes em relação ao produto a ser desenvolvido.
 - Análise: modelagem dos conceitos relevantes do domínio do problema, com o intuito de verificar a qualidade dos requisitos obtidos e detalhar tais requisitos em um nível adequado aos desenvolvedores.
 - Desenho (ou projeto): definição de uma estrutura implementável para um produto, que atenda aos requisitos especificados.
 - Implementação: codificação das partes que compõe o software, definidas no desenho, utilizando as tecnologias selecionadas.
 - Teste: verificação dinâmica das partes que constituem o software, utilizando um conjunto finito de casos de teste, selecionados dentro de um domínio potencialmente infinito, contra seu comportamento esperado.

Codifica-Remenda

- Parte de uma especificação incompleta ou inexistente
- Na maior parte das vezes resulta em algo que não era o que o cliente desejava
- Sofre alterações e correções (remendos) até alcançar um nível aceitável
- Nenhum processo é seguido nessas iterações

Codifica-Remenda



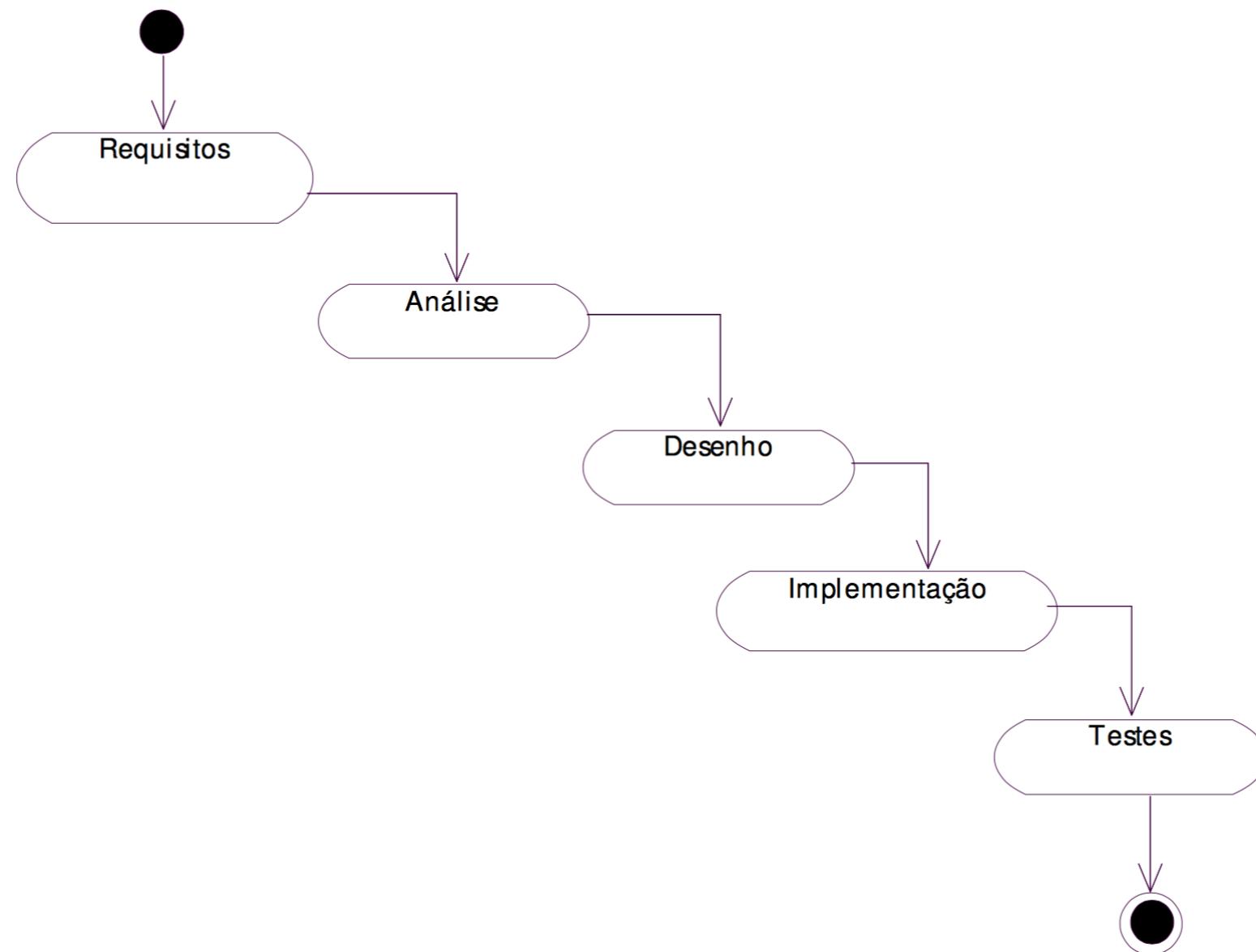
Codifica-Remenda

- É utilizado por pessoas que não tem conhecimento em Engenharia de Software
- Pode até dar certo para projetos muito pequenos com equipes também pequenas
- Aspecto positivo que não exige conhecimento técnico ou gerencial
- Modelo de alto risco

Cascata

- Também conhecido como modelo sequencial linear
- Muito utilizado no passado
- Não é um modelo ótimo quando requer mudanças no meio do projeto
- Muito difícil que o cliente identifique todas necessidades de uma vez só
- Primeira versão do produto só fica pronta ao final do projeto

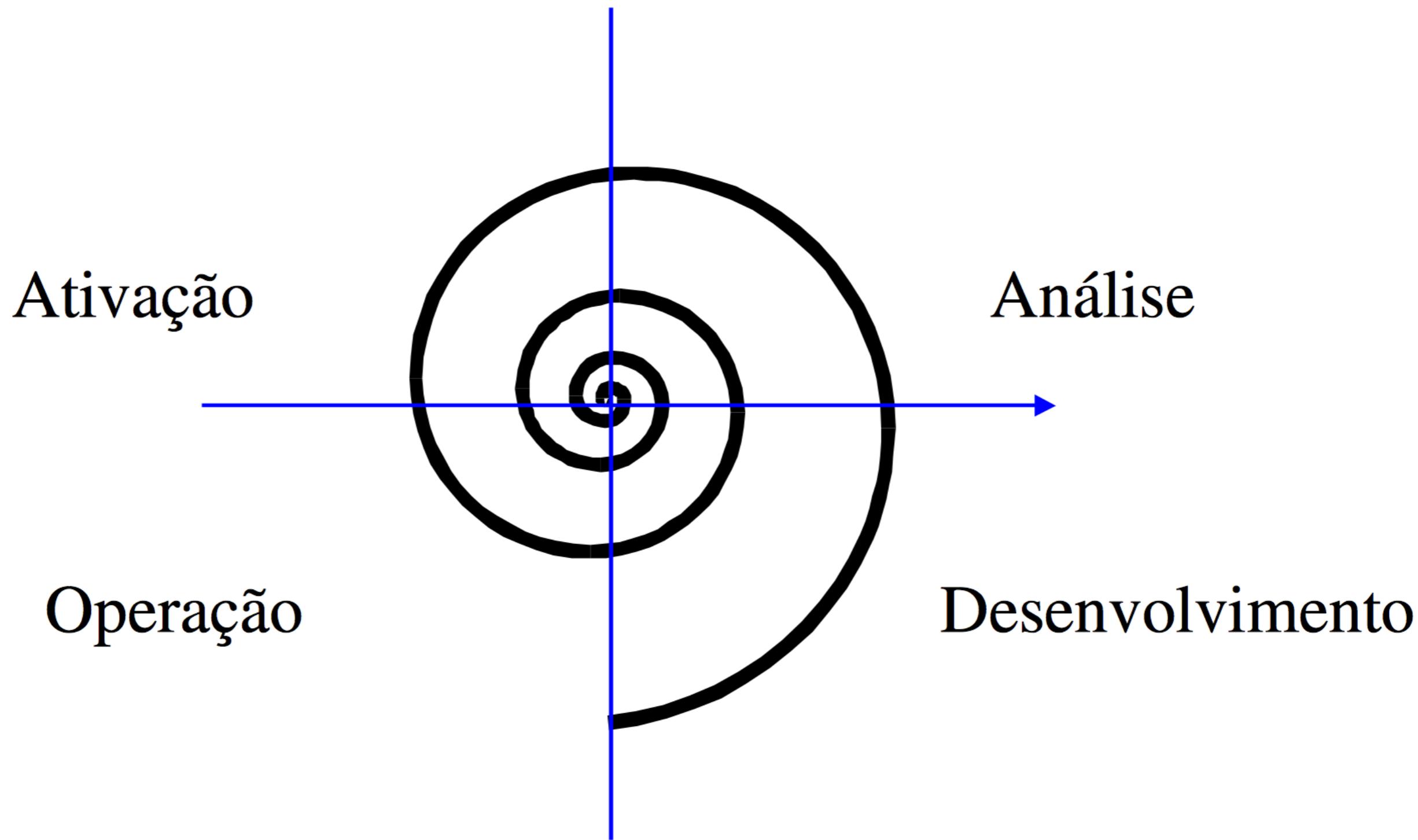
Cascata



Espiral

- A idéia é desenvolver o projeto a partir de pequenas versões incrementáveis
- Cada iteração (volta na espiral) equivale a execução do modelo cascata para uma pequena parte do software
- Para cada iteração realiza-se as 5 etapas: Levantamento de Requisitos, Análise, Desenho, Implementação e Testes.

Espiral



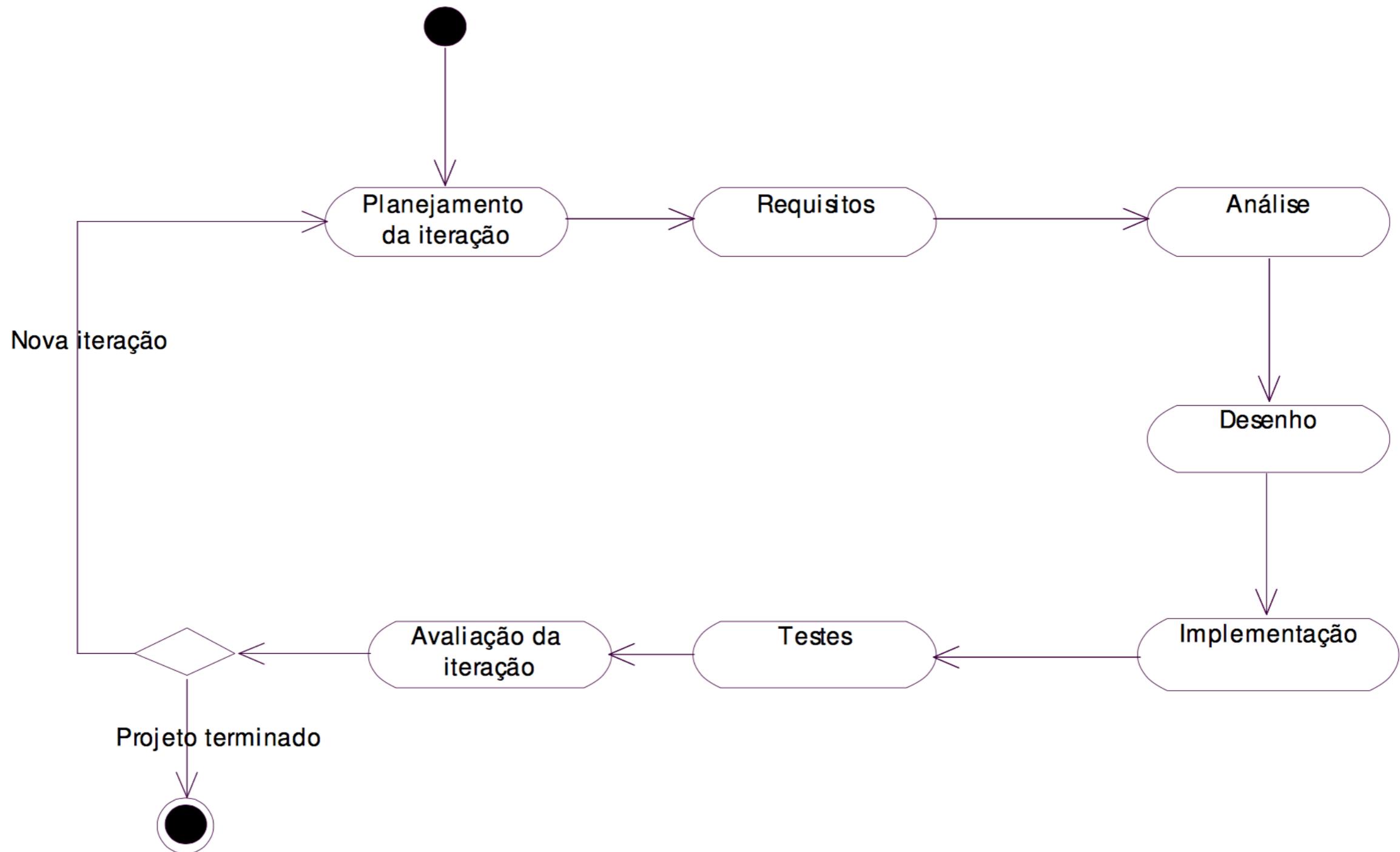
Espiral

- Esse modelo ajuda o cliente a identificar requisitos com mais facilidade
- Porém é possível que a tomada de decisões de uma etapa seja incompatível com as outras
- Apesar de ser modelo por partes, a entrega ao cliente não se dá a cada iteração ou com um conjunto delas

Incremental

- Baseado no modelo Espiral
- Diferença é que as versões parciais do produto são entregues ao cliente

Incremental



Incremental

- Sofre dos mesmos problemas do modelo Espiral
- Requer uma gerência sofisticada e arquitetura forte
- Processos famosos como XP e Scrum utilizam esse modelo de ciclo de vida
 - Conhecidos como métodos agéis

Exercícios

1. Quais são as principais fases do ciclo de vida de um produto de software?
2. Qual a definição para Processo de Software?
3. Quais são as principais etapas ligados ao desenvolvimento de software?
4. Descreva o modelo de ciclo de vida denominado Codifica-Remenda.
5. Quais são os problemas relacionados ao modelo de ciclo de vida em cascata?
6. Como podemos relacionar o modelo em cascata e o modelo em espiral?
7. Qual a grande diferença entre o modelo em espiral e o modelo incremental?

RUP

- Retomando:
 - Processo: é o conjunto de passos que tem como objetivo atingir uma meta
 - Processo de software: na ES, processo que visa a produzir o software - de modo eficiente e previsível um produto de qualidade.

RUP

- O RUP: Processo unificado de Desenvolvimento de Software
 - *Rational Unified Process*
- Definido como um *framework* (metamodelo) para gerar processos
 - Processos + Métodos + Linguagem (UML)

Principais Características do RUP

- Baseado em componentes
- Utiliza a linguagem UML para especificar, modelar e documentar artefatos
- Guiado por casos de uso
- Centrado na arquitetura
- Iterativo
- Incremental

Principais Características do RUP

- RUP é um processo configurável
 - Pode ser adaptado para diferentes tipos de softwares a serem desenvolvidos e a diferentes características do ambiente de desenvolvimento
 - Ex.: Número de pessoas na equipe, técnicas usadas, etc.
- Não existe uma maneira exata de aplicar o RUP, pois ele pode ser aplicado de várias formas e será diferente em cada projeto e organização.

Principais Características do RUP

- É bem definido e estruturado
 - define claramente quem é responsável pelo que, como as coisas devem ser feitas e quando fazê-las.
 - provê uma estrutura bem definida para o ciclo de vida de um projeto, articulando claramente os marcos essenciais e pontos de decisão

Princípios do RUP

- Atacar os riscos cedo e continuamente;
- Certificar-se de entregar algo de valor ao cliente;
- Focar no software executável;
- Acomodar mudanças cedo;
- Liberar um executável da arquitetura cedo;
- Construir o sistema com componentes;
- Trabalhar junto como um time;
- Fazer da qualidade um estilo de vida, não algo para depois.

Ciclos/Fases

- RUP segue as 4 fases já conhecidas
 - Concepção (define o escopo do projeto)
 - Elaboração (define os requisitos e a arquitetura)
 - Construção (desenvolve o sistema)
 - Transição (implanta o sistema)
- Vários ciclos se repetem até a aposentadoria do sistema.
 - Cada ciclo gera um produto liberado para uso.

Ciclos/Fases

fases/tempo

dimensão/componente

Análise de Requisitos

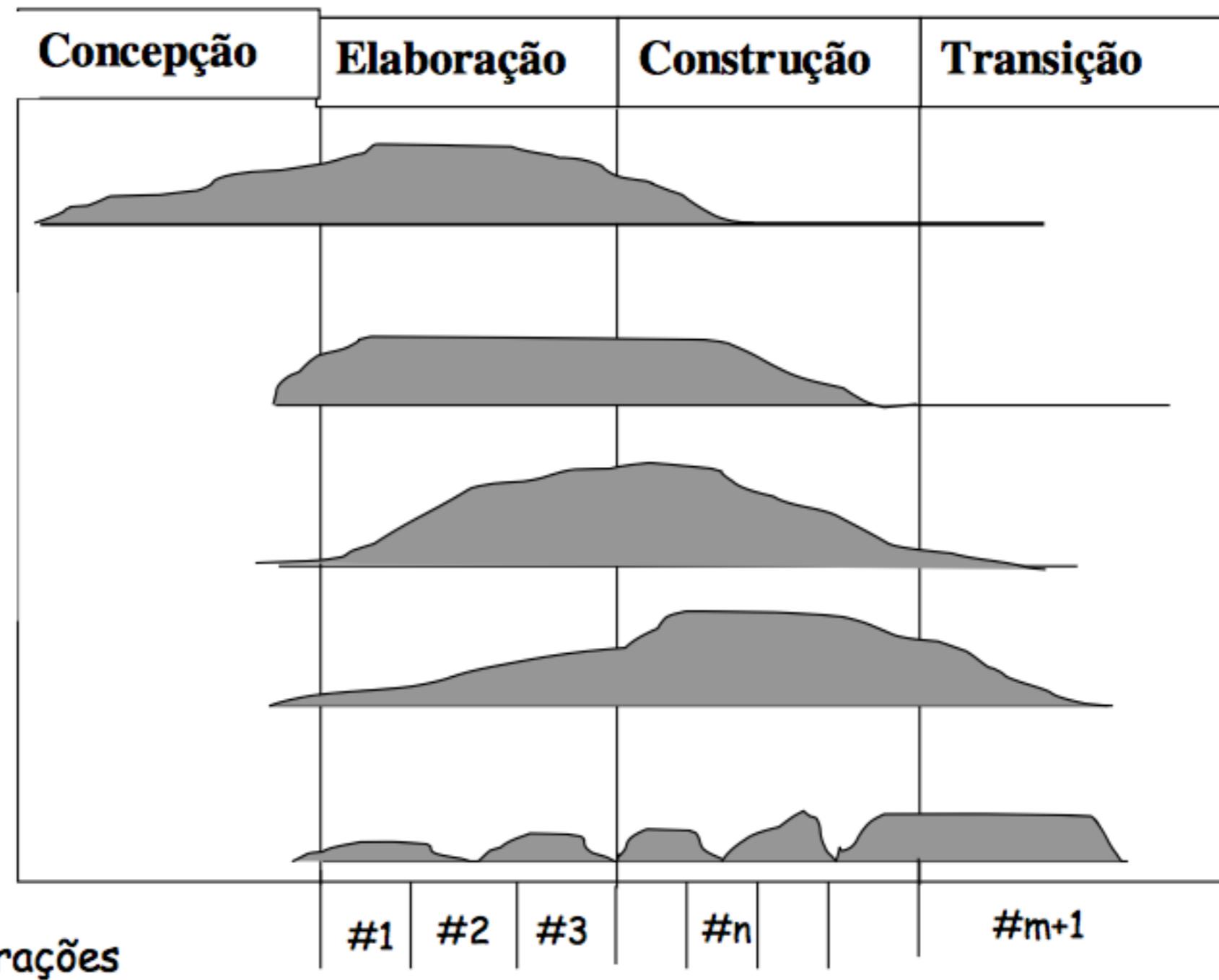
Nível de arquitetura

Design

Nível de classe

Implementação

Teste



Detalhando as principais disciplinas de ES

1. Requisitos
2. Análise
3. Desenho
4. Implementação
5. Testes
6. Gestão de Projetos

Requisitos

- O fluxo de requisitos possui atividades que visam obter um enunciado completo, claro e preciso de um produto de software
- Os requisitos correspondem a qualquer desejo, necessidade, restrição ou expectativa do cliente em relação ao software
- O conjunto de técnicas usadas para levantar, detalhar, documentar e validar esses requisitos forma a Eng. de Requisitos

Requisitos

- Resultado principal dessa fase é o documento de Especificação de Requisitos de Software
- Esse documento possui toda a caracterização do problema do cliente, e deve ser usado para a criação do produto
- A engenharia de requisitos é mais complicada para produtos novos
 - Também é mais complicada em projetos maiores

Requisitos

- A boa engenharia de requisitos é um passo essencial para o desenvolvimento de um bom produto
- Requisitos devem ser:
 - Claros
 - Completos
 - Sem ambiguidade
 - Implementáveis
 - Consistentes
 - Testáveis

Requisitos

- Requisitos que não possuem essas características devem ser revisados e renegociados com clientes e usuários

Requisitos

- As características contidas numa Especificação de Requisitos de Software incluem:
 - Funcionalidade: O que o software deverá fazer?
 - Interfaces externas: Como o software interage com as pessoas, com o hardware do sistema, com outros sistemas e com outros produtos?
 - Desempenho: qual o tempo de resposta máximo permitido dado um contexto de funcionamento, como por exemplo, 100 usuários realizando empréstimo de um livro.

Requisitos

- As características contidas numa Especificação de Requisitos de Software incluem:
 - Outros atributos: Quais as considerações sobre portabilidade, manutenibilidade e confiabilidade que devem ser observadas?
 - Restrições impostas pela aplicação: existem padrões e outros limites a serem obedecidos, como linguagem de implementação, ambientes de operação, limites de recursos etc.?

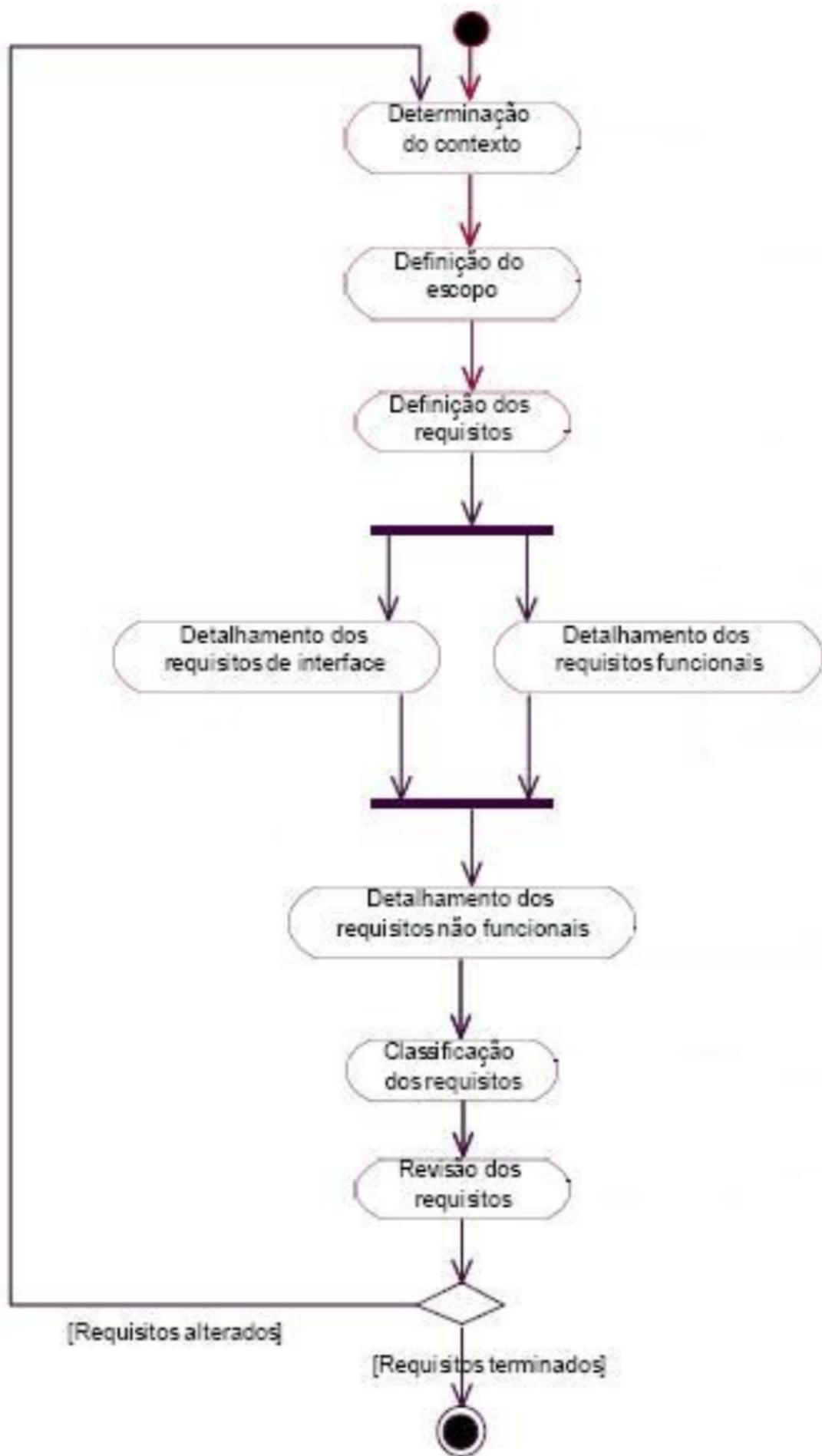


Figura 13: O Fluxo de Requisitos.

Requisitos

- A definição de requisitos do diagrama anterior produz a lista de todos os requisitos funcionais e não-funcionais
- Descreve-se os requisitos de forma sucinta, sem entrar em detalhes

Requisitos

- Costumeiramente, utiliza-se o conceito de casos de uso para descrever as funcionalidades do produto
 - Um Caso de Uso é uma fatia de funcionalidade do sistema, sem superposição nem lacunas, que representam algo de valor para os usuários finais.

Requisitos

- São também identificados o grupo de usuários do produto, denominado Atores
 - Ator é representação dos usuários e outros sistemas que interagem com o produto
- Aí entra o diagrama de casos de uso, que representa o relacionamento de cada ator com cada caso de uso
 - Diagrama de Casos de Uso deixa claro que grupos utilizam quais funções

Requisitos

Tabela 3: Exemplo de requisitos para um produto de software.

Nr	Caso de uso	Descrição
1	Gestão de Hospitais	Cadastramento dos Hospitais com leitos controlados pelo SRLEP .
2	Gestão de Leitos	Cadastro dos Leitos hospitalares controlados pelo SRLEP .
3	Autorização de envio para lista de espera	Registro da autorização da solicitação para constar na lista de espera por leitos.
4	Controle de Internações	Controle das internações em leitos, com possibilidade registro de alta, e consequente liberação do leito, além de transferências para outros leitos.
5	Consulta Escala	Visualização de quadro de escala hospitalar.

Requisitos

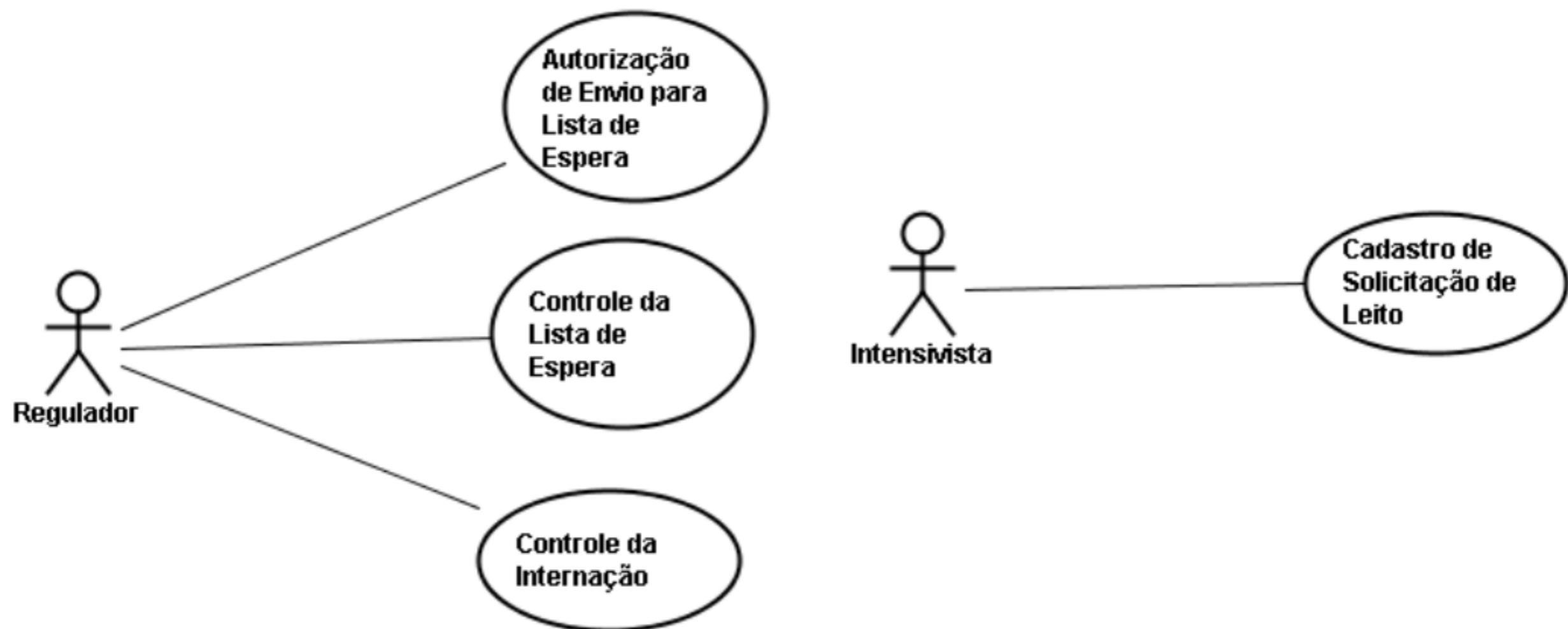


Figura 14: Exemplo de diagrama de caso de uso.

Requisitos

- Requisitos de Interface
- Requisitos Funcionais
- Requisitos Não-Funcionais
- Classificação dos requisitos
- Revisão dos Requisitos

Requisitos

- Durante o detalhamento de Requisitos de Interface são detalhados os aspectos da interface do produto que os usuários consideram requisitos.
- Normalmente são criados esboços de interface. Porém, deve-se ter o cuidado para não descer ao nível de *design* nesses esboços.
- Desenho independente de tecnologia

Requisitos

Gestão de Condições de Internação	
Pesquisa por Condições de Internação	
Condição	Insuficiência hepática grave (até 50 caracteres)
<Pesquisar>	
Condições de internação recuperadas	
Condição	Tipo
Insuficiência respiratória grave	Primária
Coma mixedematoso	Secundária
<Novo> <Alterar>	
Condições de Internação	
Condição*	Insuficiência hepática grave (até 50 caracteres)
Tipo	[Primária; Secundária]
<Salvar>	

Figura 15: Exemplo de um protótipo de tela independente de tecnologia.

Requisitos

- Funcionais:
 - Descrevem explicitamente as funcionalidades e serviços do sistema
 - Documenta como o sistema deve reagir entradas específicas
 - Como deve se comportar em determinadas situações
 - O que o sistema não deve fazer

Requisitos

- Não-Funcionais:
 - Definem propriedades e restrições do sistema
 - Exemplos: segurança, desempenho, espaço em disco
 - Podem ser do sistema todo ou de partes do sistema
 - Requisitos não-funcionais podem ser mais críticos que requisitos funcionais. Se não satisfaz, o sistema é inútil.

Requisitos

- Não-Funcionais:
 - Requisitos do Produto: Especificam o comportamento do software (ex.: desempenho)
 - Requisitos Organizacionais: Consequência de políticas e procedimentos das empresas (ex.: padrões do cliente)
 - Requisitos Externos: Derivados do ambiente ou fatores externos ao sistema (ex.: legislação)

Requisitos

- Classificação
 - determina as prioridades relativas dos requisitos e avalia a estabilidade e a complexidade de realização.

Requisitos

- Revisão
 - determina se todos eles satisfazem os critérios de qualidade de requisitos e se a Especificação dos Requisitos do Software está clara e bem entendida por todas as partes interessadas.

Exercícios

1. Qual o objetivo do fluxo de requisitos?
2. Por que o levantamento de requisitos para produtos novos geralmente é mais difícil que para produtos já existentes?
3. O que deve conter uma Especificação de Requisitos?
4. Quais as atividades de requisitos?
5. Em qual atividade é feita uma descrição dos requisitos de forma sucinta?
6. Como são representados os grupos de usuários do produto?
7. Que notação é utilizada para descrição dos requisitos funcionais?
8. Porque é aconselhado fazer um esboço das telas de forma independente de tecnologia?

Análise

- Objetivos:
 - modelar de forma precisa os conceitos relevantes do domínio do problema, identificados a partir do levantamento de requisitos;
 - verificar a qualidade dos requisitos identificados;
 - detalhar esses requisitos o suficiente para que atinjam o nível de detalhe adequado aos desenvolvedores.

Análise

- Nós utilizaremos métodos baseados em tecnologias Orientada a Objetos
- Resulta um modelo de análise de software
- UML

Análise

- Modelo de análise deve conter detalhes suficientes para servir de base ao desenho do produto
- Cuidado para não incluir detalhes de implementação.
- Atenha-se aos detalhes do problema

Análise

- Quando se usa um Modelo de Análise orientado a objetos, os requisitos funcionais são tipicamente descritos e verificados através dos seguintes recursos de notação:
 - Os casos de uso descrevem o comportamento esperado do produto. Os diagramas de casos de uso descrevem os relacionamentos dos casos de uso entre si e com os atores.
 - As classes representam os conceitos do mundo da aplicação que sejam relevantes para a descrição mais precisa dos requisitos, exibindo os relacionamentos entre essas.
 - As realizações dos casos de uso mostram como objetos das classes descritas colaboram entre si para realizar os principais roteiros que podem ser percorridos dentro de cada caso de uso.

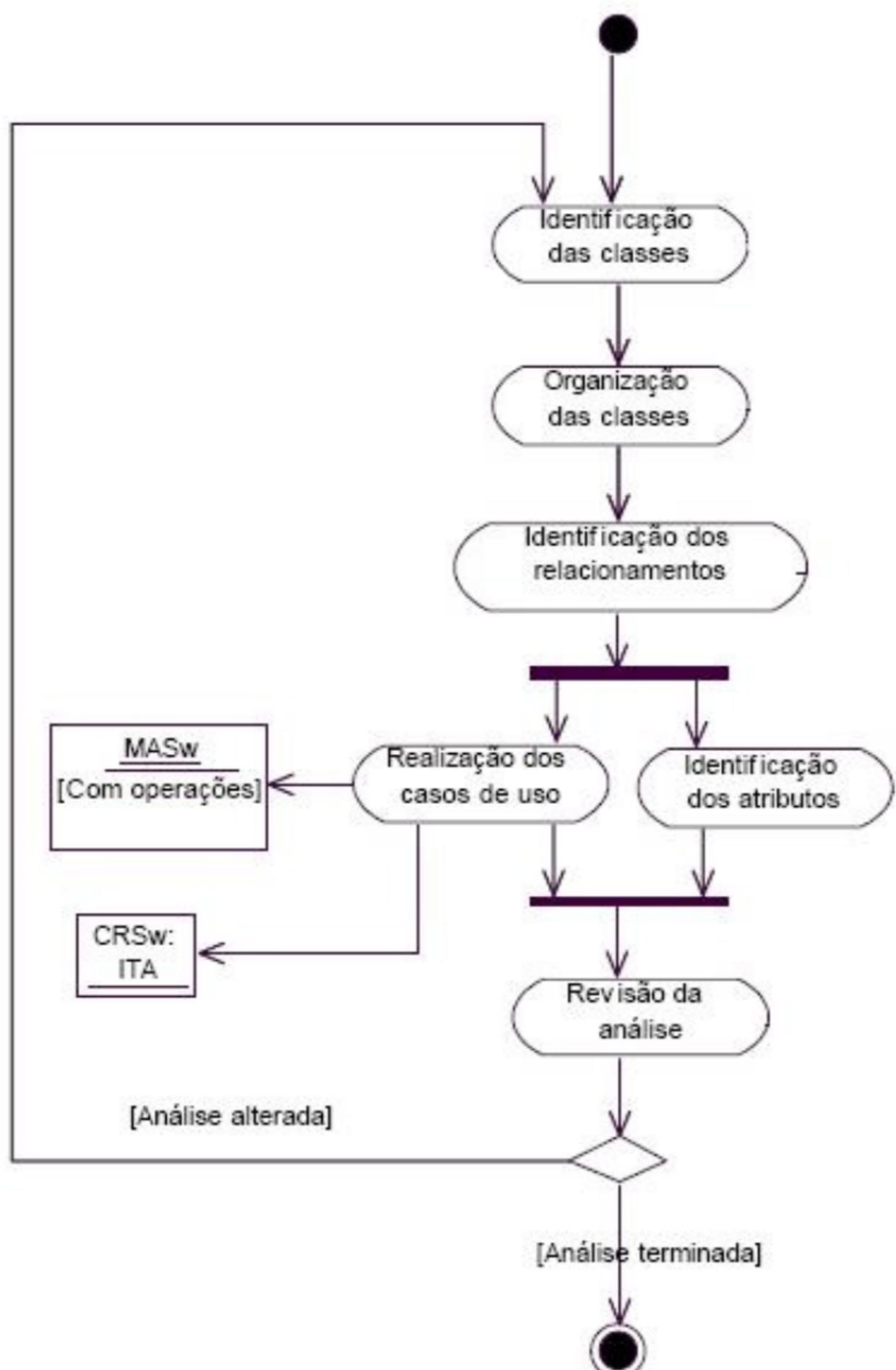


Figura 16: O Fluxo de Análise.

Análise

- Identificação das classes:
 - São analisados os fluxos dos casos de uso e outros documentos relevantes em relação ao produto desejado.
 - Os conceitos candidatos a classes são localizados, e filtrados de acordo com vários critérios.

Análise

- Organização das Classes:
 - Organiza as classes em pacotes lógicos (agrupamentos de classes correlatas) e lhes atribui os estereótipos de entidade, fronteira e controle, dependendo do papel que desempenham no modelo.

Análise

- Identificação de Atributos
 - Levanta os atributos de análise, isto é, propriedades que fazem parte do conceito expresso pela classe.

Análise

- Todas as atividades descritas anteriormente dão origem ao diagrama de classes contendo todos os detalhamentos de atributos, divisão em pacotes e com especificação dos relacionamentos.
- O diagrama de classe fornece uma visão geral do sistema e representa um artefato imprescindível no desenvolvimento.

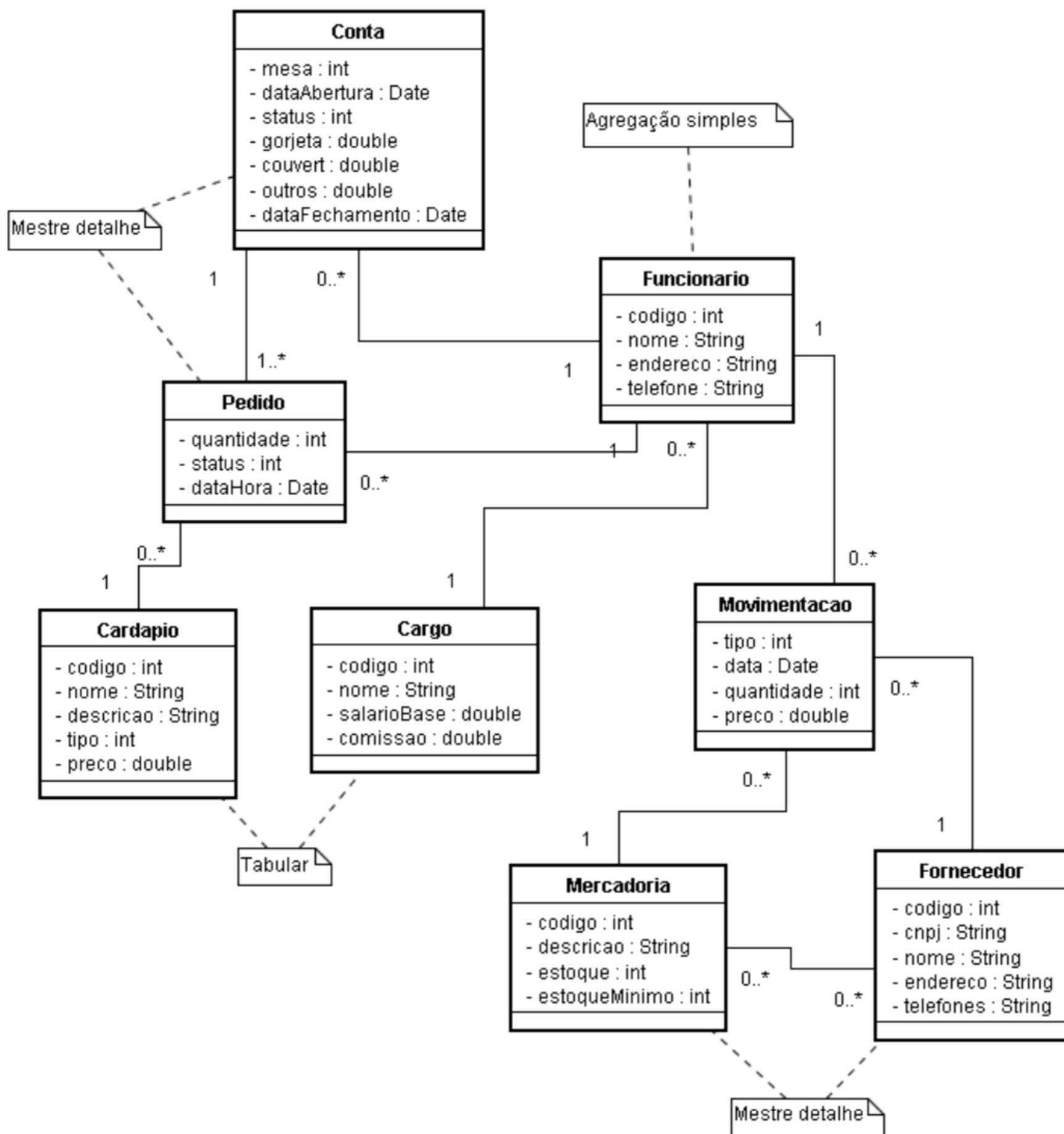


Figura 17: Exemplo de diagrama de classes exibindo classes, atributos e relacionamentos.

Análise

- Revisão da análise
 - Valida o esforço de Análise e o correspondente esforço de Requisitos. Podem acontecer várias revisões informais, individuais ou em grupo (por exemplo, dentro de uma oficina de detalhamento dos requisitos).

Exercícios

1. Qual o objetivo do fluxo de análise?
2. O que é descrito no diagrama de classes?
3. O que é a realização dos casos de uso?
4. Que locais são consultados para se tentar identificar classes?

Desenho (*Design*)

- Tem por objetivo definir uma estrutura implementável para um produto de software que atenda os requisitos
- Diferentemente da análise, que foca no problema, o design foca na solução do problema

Desenho (*Design*)

- O desenho de um produto de software deve considerar os seguintes aspectos:
 - O atendimento dos requisitos não-funcionais, como os requisitos de desempenho e usabilidade;
 - A definição de classes e outros elementos de modelo em nível de detalhe suficiente para a respectiva implementação;
 - A decomposição do produto em componentes cuja construção seja relativamente independente, de forma que eventualmente possa ser realizada por pessoas diferentes, possivelmente trabalhando em paralelo;

Desenho (*Design*)

- O desenho de um produto de software deve considerar os seguintes aspectos:
 - A definição adequada e rigorosa das interfaces entre os componentes do produto, minimizando os efeitos que problemas em cada um dos componentes possam trazer aos demais elementos;
 - A documentação das decisões de desenho, de forma que estas possam ser comunicadas e entendidas por quem vier a implementar e manter o produto;
 - A reutilização de componentes, mecanismos e outros artefatos para aumentar a produtividade e a confiabilidade;
 - O suporte a métodos e ferramentas de geração semi- automática de código.

Desenho (*Design*)

- O foco do fluxo de Desenho é a concepção de estruturas robustas, que tornem a implementação mais confiável e produtiva.

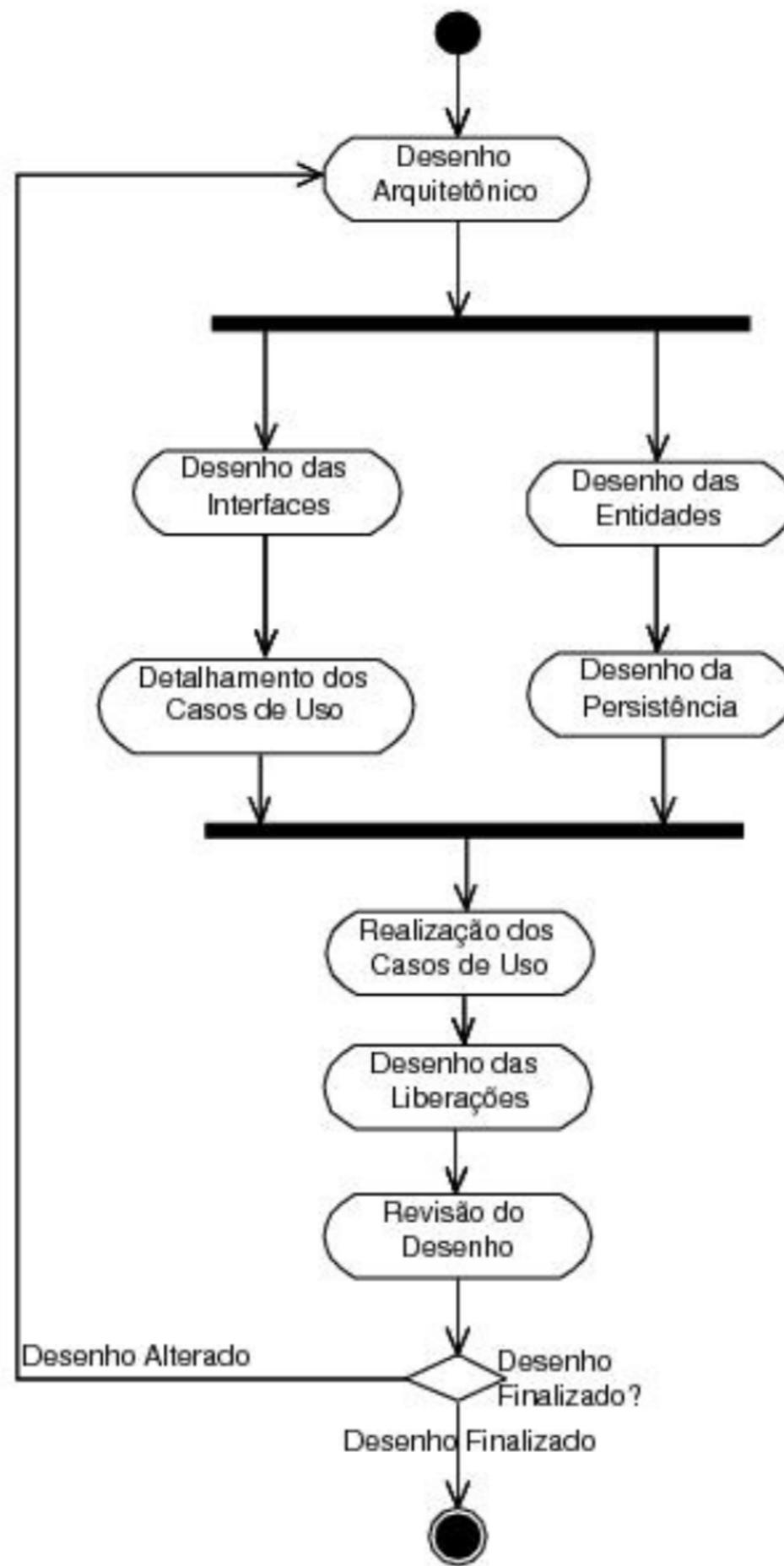


Figura 18: O Fluxo de Desenho.

Desenho (*Design*)

- O foco do fluxo de Desenho é a concepção de estruturas robustas, que tornem a implementação mais confiável e produtiva.

Desenho (*Design*)

- Desenho arquitetônico
 - Trata de aspectos estratégicos de desenho
 - Define a divisão do produto em subsistemas, escolhendo-se as tecnologias mais adequadas
 - As decisões sobre tecnologia devem viabilizar o atendimento dos requisitos não-funcionais e a execução do projeto dentro do prazo e custos estimados

Desenho (*Design*)

- Desenho arquitetônico
 - Uma decisão fundamental é a escolha do ambiente definitivo de implementação, se isso já não fizer parte das restrições de desenho constantes da Especificação dos Requisitos do Software
 - Um exemplo da escolha do ambiente definitivo de implementação seria: vamos utilizar Java, na versão 1.6, com o Hibernate 2.1, utilizando o Framework Struts 2.0, que será executado no container Tomcat 5.5. Todas essas decisões devem ser registradas e comunicadas à equipe
 - Define a arquitetura geral do produto como por exemplo seus subsistemas

Desenho (*Design*)

- A arquitetura de um produto corresponde a todas as tecnologias empregadas para sua construção, aliadas a forma de interconexão entre tais tecnologias.
- Uma empresa com diversos produtos normalmente segue a arquitetura já existente para produtos novos.
- Quando há a necessidade de mudarmos de tecnologia, é necessário muito estudo e testes para se obter o nível de entendimento e maturidade necessário para implementar um novo projeto.

Desenho (*Design*)

- Desenho das Interfaces
 - Trata do desenho das interfaces reais do produto em seu ambiente definitivo de implementação
 - Baseada nos requisitos de interfaces constantes da Especificação dos Requisitos do Software

Desenho (*Design*)

- Detalhamento dos Casos de Uso
 - Resolve os detalhes dos fluxos dos casos de uso de desenho, considerando os componentes reais das interfaces.
 - Todos os fluxos alternativos devem ser detalhados, inclusive os que consistem de emissão de mensagens de exceção ou erro

Desenho (*Design*)

- Desenho das entidades
 - Transforma as classes de entidade do Modelo de Análise nas classes correspondentes do Modelo de Desenho
 - Inclui-se aqui a consideração de possível reutilização dessas classes, ou de transformação delas em componentes fisicamente distintos
 - Devem ser inseridos detalhes como naveabilidade

Desenho (*Design*)

- Desenho de Persistência
 - Trata da definição de estruturas externas de armazenamento persistente, como arquivos e bancos de dados
 - Dois problemas devem ser resolvidos:
 - a definição física das estruturas persistentes externas, como o esquema de um banco de dados
 - a realização de uma ponte entre o modelo de desenho orientado a objetos e os paradigmas das estruturas de armazenamento, que geralmente são de natureza bastante diferente

Desenho (*Design*)

- Realização dos casos de uso
 - Determina como os objetos das classes de desenho colaborarão para realizar os casos de uso

Desenho (*Design*)

- Revisão do desenho
 - Valida o esforço de Desenho, confrontando-o com os resultados dos Requisitos e da Análise.

Exercícios

1. Qual a principal diferença entre a Análise e o Desenho?
2. Quais são os principais aspectos a considerar no Desenho?
3. O que vem a ser a arquitetura de um produto?

Implementação

- O fluxo de Implementação detalha os componentes previstos no desenho, descrevendo todos os componentes de código fonte e código binário, em nível de linguagem de programação ou de acordo com as tecnologias escolhidas

Implementação

- A implementação inclui as seguintes tarefas:
 - planejamento detalhado da implementação das unidades de cada iteração;
 - implementação das classes e outros elementos do modelo de desenho, em unidades de implementação, geralmente constituídas de arquivos de código fonte;
 - verificação das unidades, por meio de revisões, inspeções e testes de unidade;
 - compilação e ligação das unidades;
 - integração das unidades entre si;

Implementação

- A implementação inclui as seguintes tarefas:
 - integração das unidades com componentes reutilizados, adquiridos de terceiros ou reaproveitados de projetos anteriores;
 - integração das unidades com os componentes resultantes das liberações anteriores.

Implementação

- Considera-se que na fase de implementação é gerada a documentação do produto
 - Manuais de usuário
 - Ajuda online
 - Websites integrados ao produto
 - Material de treinamento
 - etc

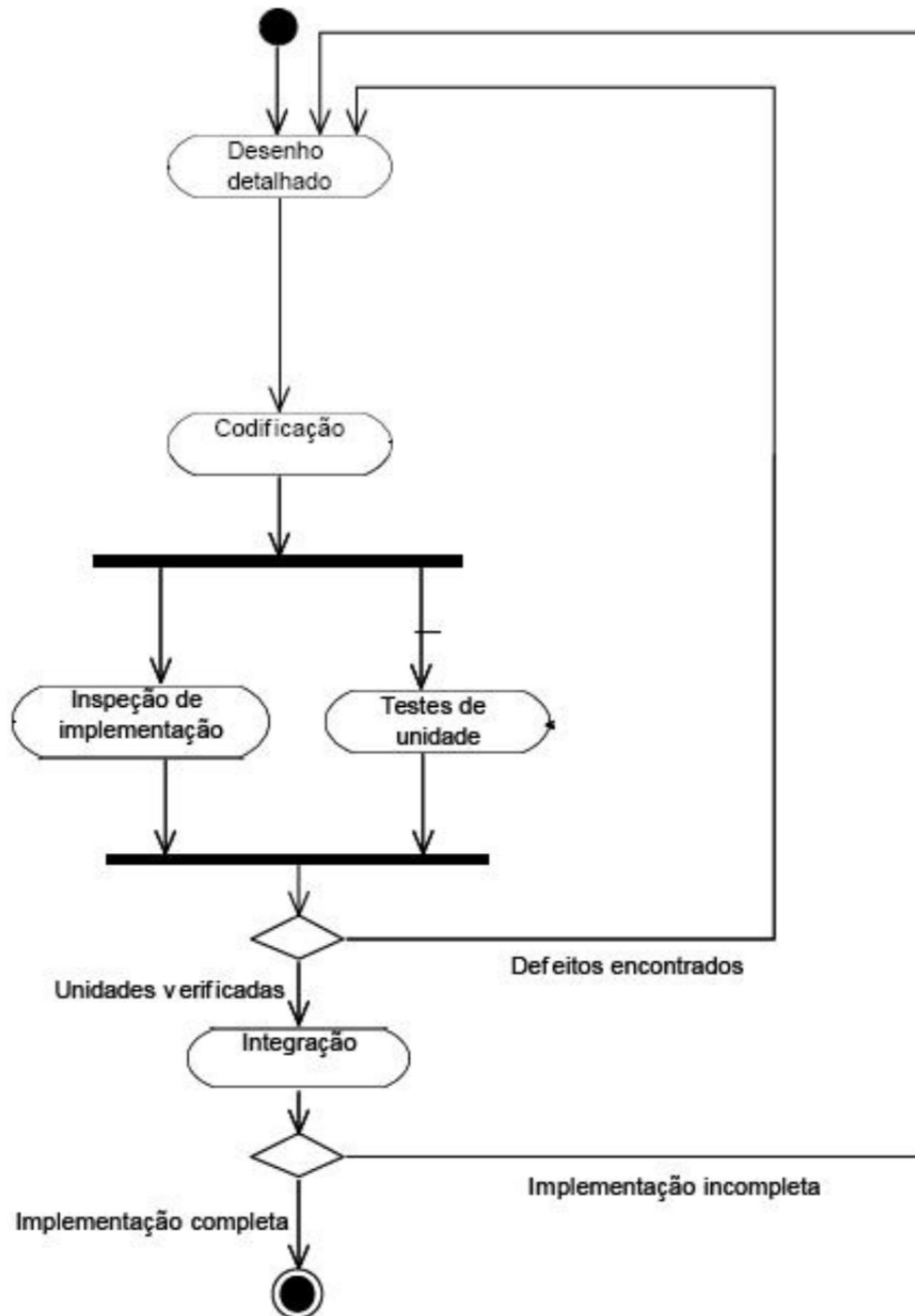


Figura 20: O Fluxo de Implementação.

Implementação

- O fluxo de Implementação contém um grupo de atividades de implementação normal, e duas atividades de categoria à parte, que são as atividades de “Documentação de usuário” e “Prototipagem”

Implementação

- Desenho detalhado
 - Preenche os detalhes restantes do Modelo de Desenho, no nível necessário para guiar codificação

Implementação

- Codificação
 - Tradução do desenho detalhado no código de uma ou mais linguagens de programação
 - Essa codificação é normalmente feita utilizando uma IDE (Integrated Development Environment), que são ferramentas para desenvolvimento que incluem uma série de bibliotecas, atalhos, wizards e outras funcionalidades integradas, de forma a tornar mais fácil a vida do desenvolvedor

Implementação

- Inspeção de implementação
 - Verifica, de forma rigorosa, o desenho detalhado e o código, por meio de um grupo de revisores pares dos autores

Implementação

- Testes de Unidade
 - São feitos usando-se componentes de teste que devem também ter sido desenhados, revistos, codificados e compilados nas atividades anteriores
 - Tecnologia muito utilizada ultimamente é o xUnit
 - Provêm facilidades para criar os testes de unidade

```
/**  
 * Testa a classe triangulo.  
 */  
public class TestTriangulo extends TestCase {  
  
    /**  
     * Construtor padrao.  
     */  
    public TestTriangulo() {  
    }  
  
    public void testeEquilatero()  
    {  
        Triangulo t = new Triangulo(3,3,3);  
        String result = t.classifica();  
        assertEquals(result, "EQUILATERO");  
    }  
  
    public void testeIsosceles()  
    {  
        Triangulo t = new Triangulo(3, 3, 4);  
        String result = t.classifica();  
        assertEquals(result, "ISOSCELES");  
    }  
  
    public void testeEscaleno()  
    {  
        Triangulo t = new Triangulo(3,4,5);  
        String result = t.classifica();  
        assertEquals(result, "ESCALENO");  
    }  
}
```

Implementação

- Integração
 - Liga as unidades implementadas com os componentes construídos em liberações anteriores, reutilizados de projetos anteriores ou adquiridos comercialmente
 - Código resultante passa para o fluxo de testes

Implementação

- Documentação do usuário
 - Tem como principal objetivo produzir uma documentação que auxilie o uso do sistema por parte dos usuários

Implementação

- Documentação do usuário
 - Tem como principal objetivo produzir uma documentação que auxilie o uso do sistema por parte dos usuários

Implementação

- Prototipagem
 - Executa as mesmas atividades da implementação normal, mas de maneira informal e sem maiores preocupações com padronização e documentação, pois o código resultante é sempre descartável

Exercícios

1. Qual o objetivo do fluxo de Implementação?
2. O que significa integrar no fluxo de Implementação?
3. Em qual atividade é feito a programação utilizando alguma linguagem ou tecnologia?
4. O que são as IDEs?
5. O que é um teste de unidade?
6. O que é a documentação de usuário? Em qual atividade ela é desenvolvida?

Testes

- O teste de software trata da verificação dinâmica do funcionamento do programa utilizando um conjunto de casos de teste
- A fase de testes abrange a procura por erros lógicos e também de requisitos
 - Tipicamente, uma organização começa a implementar o fluxo de teste pelas baterias de testes de aceitação. Esse tipo de teste tem por objetivo validar o produto, ou seja, verificar se ele atende aos requisitos identificados.

Testes

- Elementos chave:
 - dinâmica: o teste exige a execução do produto, embora algumas de suas atividades possam ser realizadas antes do produto estar operacional;
 - conjunto finito: o teste exaustivo é geralmente impossível, mesmo para produtos simples;
 - escolhido: é necessário selecionar testes com alta probabilidade de encontrar defeitos, preferencialmente com o mínimo de esforço;
 - comportamento esperado: para que um teste possa ser executado é necessário saber o comportamento esperado, para que ele possa ser comparado com o obtido.

Testes

- Um caso de teste especifica como deve ser testado uma parte do software. Essa especificação inclui as entradas, saídas esperadas, e condições sob as quais os testes devem ocorrer.
- Um procedimento de teste detalha as ações a serem executadas em um determinado caso de teste
- Observe que um caso de teste só faz sentido se for especificado o(s) procedimento(s) de teste associado(s). Sem essa associação não é possível determinar o que deve ser feito com as entradas especificadas no caso de teste

Identificação	Procedimento de Teste Login
Objetivo	Efetuar o login de um usuário no Merci.
Requisitos especiais	A TelaPrincipal deve estar no estado SEM USUARIO.
Fluxo	<ol style="list-style-type: none">1. Preencher o campo login e senha2. Pressionar o botão login.

Identificação	Login inválido com caracteres não permitidos								
Itens a testar	Verificar se a tentativa de login utilizando um identificador inválido, contendo caracteres não permitidos, exibe a mensagem apropriada.								
Entradas	<table border="1"> <thead> <tr> <th>Campo</th><th>Valor</th></tr> </thead> <tbody> <tr> <td><i>Login</i></td><td>pasn!</td></tr> <tr> <td><i>Senha</i></td><td>aaaa</td></tr> </tbody> </table>	Campo	Valor	<i>Login</i>	pasn!	<i>Senha</i>	aaaa		
Campo	Valor								
<i>Login</i>	pasn!								
<i>Senha</i>	aaaa								
Saídas esperadas	<table border="1"> <thead> <tr> <th>Campo</th><th>Valor</th></tr> </thead> <tbody> <tr> <td><i>Login</i></td><td>pasn!</td></tr> <tr> <td><i>Senha</i></td><td>aaaa (exibida com *'s)</td></tr> <tr> <td><i>Mensagem</i></td><td>O campo login deve ter no mínimo 2 e no máximo 8 caracteres alfabéticos.</td></tr> </tbody> </table>	Campo	Valor	<i>Login</i>	pasn!	<i>Senha</i>	aaaa (exibida com *'s)	<i>Mensagem</i>	O campo login deve ter no mínimo 2 e no máximo 8 caracteres alfabéticos.
Campo	Valor								
<i>Login</i>	pasn!								
<i>Senha</i>	aaaa (exibida com *'s)								
<i>Mensagem</i>	O campo login deve ter no mínimo 2 e no máximo 8 caracteres alfabéticos.								
Estado alcançado	SEM USUARIO								
Ambiente	Banco de dados de teste.								
Procedimentos	Procedimento de Teste Login								
Dependências	Não aplicável.								

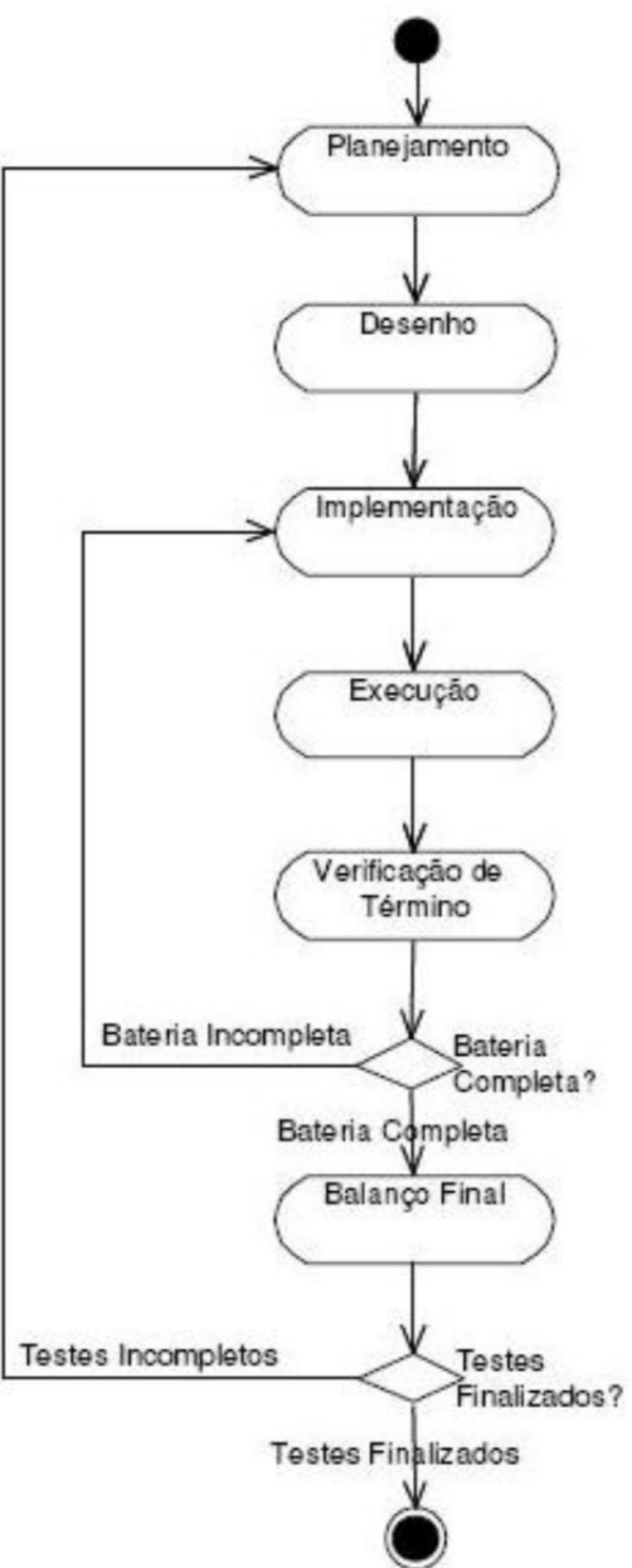


Figura 23: O Fluxo de Teste.

Testes

- O fluxo de teste tem dois grandes grupos:
 - preparação
 - Planejamento da Bateria de Testes
 - realização
 - Execução dos testes, identificação dos bugs, correção dos mesmos e criação de relatórios

Exercícios

1. O que é o teste de software?
2. O que é um procedimento de teste? O que é um caso de teste? Qual a ligação que existe entre esses dois conceitos?
3. As atividades de teste são divididas em dois grandes grupos. Quais são eles?
4. Quais são os objetivos do teste de aceitação?

Levantamento de Requisitos

- O início para toda a atividade de desenvolvimento de software é o levantamento de requisitos
- Algumas literaturas propõem um processo genérico de levantamento e análise

Levantamento de Requisitos

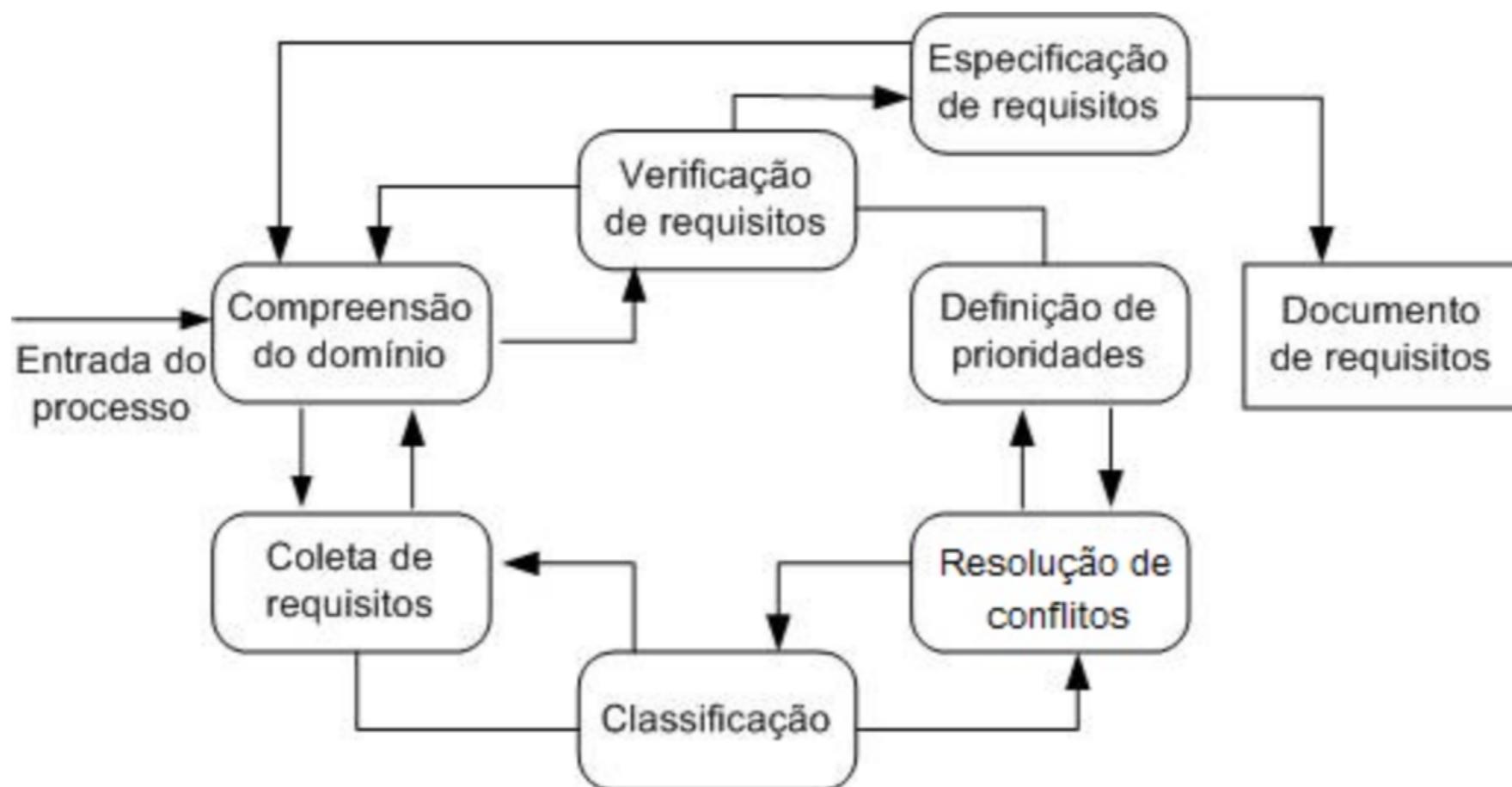
- Esse processo possui as seguintes atividades:
 - Compreensão do domínio: Os analistas devem desenvolver sua compreensão do domínio da aplicação;
 - Coleta de requisitos: É o processo de interagir com os stakeholders do sistema para descobrir seus requisitos. A compreensão do domínio se desenvolve mais durante essa atividade;
 - Classificação: Essa atividade considera o conjunto não estruturado dos requisitos e os organiza em grupos coerentes;

Stakeholder é qualquer pessoa ou organização que tenha interesse, ou seja afetado pelo projeto.

Levantamento de Requisitos

- Esse processo possui as seguintes atividades:
 - Resolução de conflitos: Quando múltiplos stakeholders estão envolvidos, os requisitos apresentarão conflitos. Essa atividade tem por objetivo solucionar esses conflitos;
 - Definição das prioridades: Em qualquer conjunto de requisitos, alguns serão mais importantes do que outros. Esse estágio envolve interação com os stakeholders para a definição dos requisitos mais importantes;
 - Verificação de requisitos: Os requisitos são verificados para descobrir se estão completos e consistentes e se estão em concordância com o que os stakeholders desejam do sistema.

Levantamento de Requisitos



O **levantamento e análise de requisitos** é um processo iterativo, com uma contínua validação de uma atividade para outra

Dificuldades

- O problema de não saber especificar corretamente o que o sistema deverá fazer é muito antigo.
- Um estudo de 1968 mostra que dois terços das empresas analisadas estavam desapontadas com o atendimento recebido em sistemas de informação
- Após muito anos a situação não é muito diferente

Dificuldades

- Algumas das questões são:
 - Na fase de levantamento de requisitos do projeto, onde não é utilizada uma técnica adequada para extrair os requisitos do sistema;
 - A falha do analista em não descrever os requisitos do sistema de modo claro, sem ambigüidades, conciso e consistente com todos os aspectos significativos do sistema proposto.

Dificuldades

- Entre as dificuldades encontradas na fase de levantamento de requisitos:
 - o usuário principal do sistema não sabe o que quer que o sistema faça ou sabe e não consegue transmitir para o analista;
 - requisitos identificados, mas que não são realistas e não identificam os requisitos similares informados por pessoas diferentes. Um stakeholder errado afetará em perda de tempo e dinheiro para ambas as partes envolvidas no desenvolvimento do sistema.

Dificuldades

- Efeitos colaterais do levantamento de requisitos inadequado:
 - diagnóstico pobre com conclusões comprometidas;
 - não identificação das causas dos problemas;
 - custos elevados;
 - prazos vencidos ou comprometedores;
 - omissão de processos fundamentais e descréditos.

Dificuldades

- Por outro lado, um levantamento de requisitos adequado gera:
 - boa definição de projeto;
 - efetividade;
 - informações necessárias a um perfeito diagnóstico e de soluções inteligentes

Técnicas de Levantamento de Requisitos

- As técnicas de levantamento de requisitos têm por objetivo superar as dificuldades relativas a esta fase.
- Todas as técnicas possuem um conceito próprio e suas respectivas vantagens e desvantagens, que podem ser utilizadas em conjunto pelo analista.

Métodos

- As técnicas de levantamento de requisitos se dividem nos seguintes grupos:
 - Métodos de Conversação
 - O método de conversação fornece um meio de comunicação verbal entre duas ou mais pessoas. Sendo uma forma natural de expressar as necessidades, ideias e responder às perguntas, é bastante eficaz para identificar e compreender as necessidades do entrevistado
 - Esses métodos fornecem a maneira natural de expressar as necessidades e as idéias e identificar os requisitos do produto.
 - Métodos de Observação
 - Utilizado para a compreensão do domínio da aplicação, observando as atividades humanas.

Métodos

- As técnicas de levantamento de requisitos se dividem nos seguintes grupos:
 - Métodos Analíticos
 - Conjunto de técnicas para análise de documentação e conhecimento existentes com o intuito de adquirir requisitos através do levantamento de informação pertinentes ao sistema a ser especificado, como por exemplo, domínio do negócio, fluxos de trabalho e características do produto.
 - Métodos Sintéticos
 - Algumas vezes em projetos complexos um único método de levantamento de requisitos não é suficiente para capturar os requisitos detalhadamente. Para solucionar este problema os analistas de requisitos tentam utilizar diferentes métodos de levantamento de requisitos. Por exemplo, em alguns casos é utilizado o método de entrevista antes de se fazer um estudo etnográfico. Ao invés de utilizar a combinação de diferentes técnicas de levantamento de requisitos, é possível utilizar métodos sintéticos, que são formados pela combinação das outras técnicas em uma única.

Métodos de Conversação

- Entrevistas:
 - Uma das técnicas tradicionais mais simples de utilizar
 - Produz bons resultados na fase inicial de obtenção de dados
 - O entrevistador deve dar espaço ao entrevistado para esclarecer suas necessidades

Métodos de Conversação

- Vantagens da Técnica de Entrevistas:
 1. Com um plano geral bem elaborado, o analista terá facilidade em descobrir que informação o usuário está mais interessado e usar um estilo adequado ao entrevistar;
 2. Poder alterar o curso da entrevista de forma a obter informações sobre aspectos importantes que não tinham sido previstos no planejamento da entrevista;
 3. Poder alterar a ordem seqüencial das perguntas;
 4. Poder eliminar perguntas anteriormente planejadas;
 5. Poder incluir perguntas que não estavam na programação da entrevista;
 6. Poder motivar o entrevistado no decorrer do processo;

Métodos de Conversação

- Desvantagens da Técnica de Entrevistas:
 1. Podem ocorrer desvios de curso, no decorrer da entrevista;
 2. Consumir mais tempo e recursos com sua realização;
 3. Tratamento diferenciado para os entrevistados;
 4. É necessário ter um plano de entrevista para que não haja dispersão do assunto principal e a entrevista fique longa, deixando o entrevistado cansado e não produzindo bons resultados;
 5. O usuário tem dificuldade de concentração em reuniões muito longas;
 6. O entrevistado pode não saber expressar corretamente suas necessidades ao analista.

Métodos de Conversação

- Workshop:
 - Trata-se de uma técnica de elicitação em grupo usada em uma reunião estruturada
 - Devem fazer parte do grupo uma equipe de analistas e uma seleção dos stakeholders que melhor representam a organização e o contexto em que o sistema será usado, obtendo assim um conjunto de requisitos bem definidos.

Métodos de Conversação

- Vantagens da Técnica de Workshop:
 1. Obtêm um conjunto de requisitos bem definido;
 2. Trabalho em equipe tornando o levantamento de requisitos mais eficaz;
 3. Baixo custo e resposta relativamente rápida;
 4. Tempo de obtenção de informações é reduzido.

Métodos de Conversação

- Desvantagens da Técnica de Workshop:
 1. Por ser realizado por convocação por dia e horário, pode ocasionar problemas no presenciais dos stakeholders;
 2. Não abre caminho para ideias externas além da equipe de analistas;
 3. Dados excessivamente agregados.

Métodos de Conversação

- Brainstorming:
 - É utilizado normalmente em workshops.
 - Após os workshops serão produzidas documentações que refletem os requisitos e decisões tomadas sobre o sistema a ser desenvolvido.
 - Seu objetivo é uma apresentação do problema/ necessidade a um grupo específico, requerendo assim soluções.

Métodos de Conversação

- Vantagens da Técnica de Brainstorming:
 1. Várias pessoas pensam melhor do que uma (grupo pensante);
 2. Rompe a inibição de idéias;
 3. Generaliza a participação do membros do grupo.

Métodos de Conversação

- Desvantagens da Técnica de Brainstorming:
 1. Disponibilidade de todos pode inviabilizar o levantamento de dados.

Métodos de Conversação

- Questionário:
 - Diferente da entrevista, essa técnica é interessante quando temos uma quantidade grande de pessoas para extrair as mesmas informações.
 - As questões são dirigidas por escrito aos participantes com o objetivo de ter conhecimento sobre opiniões das mesmas questões.

Métodos de Conversação

- Vantagens da Técnica de Questionário:
 1. Atinge um grande número de pessoas;
Menores custos;
 2. Permite que os participantes respondam no momento em que acharem conveniente;
 3. Questões padronizadas garantem uniformidade.

Métodos de Conversação

- Desvantagens da Técnica de Questionário:
 1. Não há garantia de que a maioria dos participantes respondam o questionário;
 2. Os resultados são bastante críticos em relação ao objetivo, pois as perguntas podem ter significados diferentes a cada participante questionado.

Métodos de Conversação

- Grupo Focal (Focus Group):
 - É um grupo de discussão informal e de tamanho reduzido (até 12 pessoas), com o propósito de obter informação qualitativa em profundidade.
 - As pessoas são convidadas para participar da discussão sobre determinado assunto.

Métodos de Conversação

- Vantagens da Técnica de Grupo Focal (Focus Group):
 1. Baixo custo, resposta rápida e Flexibilidade;
 2. Obtêm informações qualitativas a curto prazo;
 3. Eficiente para esclarecer questões complexas no desenvolvimento de projetos;

Métodos de Conversação

- Desvantagens da Técnica de Grupo Focal (Focus Group):
 1. Dificuldades para analisar e interpretar situações;
 2. A amostra pode ser reduzida;
 3. Requer treinamento especializado;
 4. As observações podem ter uma interpretação complicada.

Métodos de Observação

- Observação:
 - A técnica resume-se em visitar o local em foco com a finalidade de observação do mesmo. Permitindo assim, coletar informações de acordo com o cotidiano das operações e execução dos processos diários do local.

Métodos de Observação

- Vantagens da Técnica de Observação:
 1. Capaz de captar o comportamento natural das pessoas;
 2. Nível de intromissão relativamente baixo;
 3. Confiável para observações com baixo nível de inferência.

Métodos de Observação

- Desvantagens da Técnica de Observação:
 1. Polarizada pelo observador;
 2. Requer treinamento especializado;
 3. Efeitos do observador nas pessoas;
 4. Não comprova/esclarece o observado;
 5. Número restrito de variáveis.

Métodos de Observação

- Protocolo de Análise:
 - Análise de protocolo é uma forma de levantamento de requisitos no qual o analista analisa as partes interessadas quando estão envolvidas em algum tipo de tarefas.

Métodos de Observação

- Vantagens da Técnica de Protocolo de Análise:
 1. Processo de extração de registro de tarefas via audio, vídeo ou notas escritas.

Métodos de Observação

- Desvantagens da Técnica de Protocolo de Análise:
 1. O analista deve ter conhecimento suficiente sobre domínio atual, a fim de compreender melhor as tarefas.

Métodos de Observação

- Etnografia:
 - É uma análise de componente social das tarefas desempenhadas numa dada organização. É utilizado para desenvolver um entendimento completo e detalhado.

Métodos de Observação

- Vantagens da Técnica de Etnografia:
 1. Capacidade de observar o comportamento do ambiente, gerando maior profundidade no conhecimento.
 2. Apoia-se no comportamento real;
 3. Permite uma abordagem integral.

Métodos de Observação

- Desvantagens da Técnica de Etnografia:
 1. Dificuldades para analisar e interpretar situações;
 2. A amostra pode ser reduzida;
 3. Requer treinamento especializado;
 4. As observações podem ter uma interpretação complicada.

Exercícios:

1. Quais são os passos para a análise de requisitos?
2. Quais são algumas dificuldades encontradas no processo de levantamento de requisitos?
3. Cite e explique 2 técnicas pertencentes aos Métodos de Conversação
4. Cite e explique 2 técnicas pertencentes aos Métodos de Observação

Métodos Analíticos

- Principais Vantagens:
 1. O estudo do conhecimento de especialistas leva a um processo sucessivo de aumento de maturidade e qualidade;
 2. Reutilização de informação já disponível salva tempo e custo;
- Principais Desvantagens:
 1. Requer dados empíricos, documentação e a opinião de especialistas, e sem estes, não é possível identificar os requisitos;
 2. Podem levar a restrição da visão do produto final;
 3. Lida com informação antiga, e com isso pode levar a replicação de erros existentes;

Métodos Analíticos

- Reuso de Requisitos:
 - Estudo e reutilização de especificações e glossários referente a projetos de sistemas legados ou sistemas de mesma família (com funcionalidades de negócio similares).

Métodos Analíticos

- Vantagens da Técnica de Reuso de Requisitos:
 1. Economia de tempo e dinheiro: Estudos tem mostrado que sistemas similares podem reutilizar acima de 80% de seus requisitos; Pode levar a uma reutilização adicional de outros itens em outras atividades do ciclo de vida de desenvolvimento (ex.: reuso do design de componentes já existentes, testes e código fonte)
 2. Redução de risco: Requerimentos reutilizados tem uma chance maior de serem compreendidos pelos stakeholders visto que já são conhecidos de certa forma;

Métodos Analíticos

- Estudo da Documentação / Análise de Conteúdo:
 - Estudo e reutilização de documentação de diferentes naturezas, para a identificação de requisitos a serem implementados no sistema que se está modelando. Uma grande variedade de documentação pode ser analisada incluindo estrutura organizacional da empresa, padrões de mercado, leis, manuais de usuário, relatório de pesquisas de mercado, glossário de termos de negócio, etc.

Métodos Analíticos

- Desvantagens da Técnica de Estudo da Documentação / Análise de Conteúdo:
 1. Documentos com problemas podem levar a uma falha na definição dos requisitos

Métodos Analíticos

- Laddering:
 - É um método de entrevistas estruturadas, um-a-um, utilizado para o levantamento de conhecimento (o que é importante e por que) de especialistas, e que consiste na criação, revisão e modificação da hierarquia de conhecimento dos especialistas geralmente na forma de diagramas hierárquicos (ex.: diagrama em árvore).

Métodos Analíticos

- Vantagens da Técnica de Laddering:
 1. Cobre um amplo domínio de requisitos;
 2. Necessita de menos tempo para a preparação e execução das sessões de levantamento;
 3. Necessita de menos experiência para a execução das sessões de levantamento;
 4. Provê um formato padrão que é adaptável para a automação computadorizada;

Métodos Analíticos

- Desvantagens da Técnica de Laddering:
 1. Não é capaz de extrair todos os tipos de requisitos;
 2. Necessita da execução combinada de outras técnicas de levantamento de requisitos para sua complementação em determinados domínios;
 3. Não é compatível com todo e qualquer domínio de requisitos, sendo necessário a verificação de sua adequação ao levantamento a ser feito;

Métodos Analíticos

- Sorteio de Cartões
 - Utilizado para capturar informações e idéias sobre estrutura de requisitos de especialistas de domínio. Neste método um conjunto de cartões é distribuído em um grupo de stakeholders onde cada cartão é impresso com a descrição das entidades do domínio.

Métodos Analíticos

- Vantagens da Técnica de Sorteio de Cartões
 1. Ajuda os stakeholders a levantar os conceitos do domínio e distinguir entre problemas de alto e baixo nível;
 2. O resultado do método pode ser utilizado como insumo para outros métodos de levantamento de requisitos;

Métodos Analíticos

- Repertory Grid:
 - Método onde os stakeholders são questionados sobre atributos e valores destes, referentes a uma série de entidades. Com esta informação é montada uma matrix de entidade X atributo.

Exercícios:

1. Cite duas técnicas pertencentes ao método analítico que você achou mais interessante (ou acredita que pode trazer mais resultados) e explique o porque da escolha.
2. Qual a diferença entre a técnica de entrevista (método de conversação) e a técnica de laddering (método analítico)?

Métodos Sintéticos

- Sessões JAD/RAD
 - Consiste em workshops e sessões de grupo nos quais stakeholders e analistas de requisitos se encontram para discutir as características desejadas do produto. Seu objetivo é envolver todos os stakeholders importantes no processo de levantamento, através de reuniões estruturadas e com foco bem definido. Depende diretamente do grau de envolvimento dos stakeholders bem como do líder das sessões JAD.

Métodos Sintéticos

- Sessões JAD/RAD
 - O JAD consiste em 3 fases principais:
 - customização: o analista prepara as tarefas para as sessões como organizar os times, preparar o material, etc.
 - fase de sessões: o analista marca uma ou mais reuniões com os stakeholders. No inicio da sessão JAD o engenheiro de requisitos provê uma visão genérica sobre o sistema e a discussão com os stakeholders continua até o fim do levantamento de requisitos.
 - agrupamento todos os requisitos levantados nas fases anteriores são convertidos em documentos de especificação de requisitos.

Métodos Sintéticos

- Vantagens da Técnica de Sessões JAD/RAD
 1. As discussões que ocorrem na fase de sessões são altamente produtivas porque resolvem dificuldades entre as partes enquanto se dá o desenvolvimento do sistema para a empresa;
 2. Melhor aplicado para grandes e complexos projetos;

Métodos Sintéticos

- Desvantagens da Técnica de Sessões JAD/RAD
 1. Somente projetos que possuem pelo menos uma das características abaixo podem utilizar o JAD:
 - Possuir alto número de stakeholders responsáveis por departamentos cross na empresa;
 - Primeiro projeto na empresa o qual é considerado crítico para o futuro da mesma;
 2. Requer mais recursos se comparado à métodos tradicionais;

Métodos Sintéticos

- Prototipação
 - Utilizado no estágio inicial do projeto. Ajuda aos stakeholders a desenvolver uma forte noção sobre a aplicação a qual ainda não foi implementada, que através da visualização da mesma eles podem identificar os reais requisitos e fluxos de trabalho do sistema. É muito utilizado quando os stakeholders são incapazes de expressar os seus requisitos ou se os mesmos não têm nenhuma experiência com o sistema.

Métodos Sintéticos

- Vantagens da Técnica de Prototipação
 1. Permite alcançar um feedback antecipado dos stakeholders;
 2. Redução de tempo e custo de desenvolvimento devido a detecção dos erros em uma fase inicial do projeto;
 3. Prove alto nível de satisfação dos usuários devido a sensação de segurança ao ver algo próximo do real;

Métodos Sintéticos

- Desvantagens da Técnica de Prototipação
 1. Demanda um alto custo de investimento, em relação à outros métodos, para ser realizado;
 2. Demanda um tempo maior para sua realização devido a complexidade do sistema e a limitações técnicas;

Métodos Sintéticos

- Questionário de Ambiente
 - Permite aos analistas o real entendimento das necessidades dos stakeholders com a coleta detalhada de informações através de observação e interação com as pessoas no ambiente de trabalho. Alguns profissionais são escolhidos e acompanhados a fundo para o completo entendimento de suas práticas de trabalho.

Métodos Sintéticos

- Vantagens da Técnica de Questionário de Ambiente
 1. Permite um levantamento profundo e detalhado das necessidades dos stakeholders;
 2. Pode ser utilizado para resolver problemas extremamente complexos;

Métodos Sintéticos

- Desvantagens da Técnica de Questionário de Ambiente
 1. Dependendo dos processos de trabalho, necessita de uma grande quantidade de tempo e pessoas para ser executado;

Métodos Sintéticos

- Storyboards
 - São sessões interativas que descreve uma sequência de atividades e eventos para um caso em específico para um processo genérico que é esperado que o sistema automatize.

Métodos Sintéticos

- Vantagens da Técnica de Storyboards
 1. Método muito eficiente no esclarecimento de requisitos relacionados a processos, fluxos de dados e tarefas;
 2. Método relativamente barato de ser executado;

Concluindo...

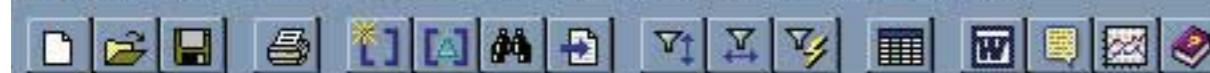
Todos os métodos de levantamento de requisitos possuem vantagens e desvantagens a serem consideradas e nenhum deles é completo dadas as inúmeras variáveis de complexidade, perfil de stakeholders, comunicação, nível de conhecimento do negócio, nível de qualificação dos profissionais de levantamento de requisitos, situações políticas, nível de comprometimento dos stakeholders, etc. Com isso, a utilização de mais de uma técnica, de forma combinada, ou a utilização de técnicas sintéticas, irá ajudar na complementação de possíveis lacunas de levantamento, além de melhorar a qualidade e completude dos requisitos visto que pode ocorrer o batimento cruzado de requisitos similares. Outro fator importante é a utilização de um framework de decisão para a escolha dos métodos mais apropriados dado o contexto do trabalho a ser realizado.

Ferramentas

- RequisitePro
- 5W2H
- Planilha de Requisitos

Ferramentas

- RequisitePro
 - O IBM® Rational® RequisitePro® é uma ferramenta de gestão de requisitos e casos de uso para que times de projetos melhorem a comunicação dos objetivos do projeto, aumenta a colaboração entre o time, reduza riscos do projeto e aumente a qualidade da aplicação desenvolvida.



Learning Project

- Features and Vision
 - ClassicsCD Vision Docu...
 - All Features
 - Requirements Traced to ...
 - FEAT1: ClassicsCD.com...
 - FEAT2: ClassicsCD Adm...
 - FEAT3: Interactive guide t...
 - FEAT4: Item Shall Be Shi...
- Glossary
- Supplementary Requirement...
- Use Cases
- ClassicsCD Requirements ...

Requirements:	Priority	Status	Cost	Difficulty	Stability
FEAT1: ClassicsCD.com Web Shop	High	Approved		Low	High
FEAT1.1: Secure payment method	Medium	Incorporated		Low	Medium
FEAT1.2: Easy browsing	Medium	Proposed		Medium	High
FEAT1.3: Ability to check status of an order	Low	Validated		Low	Medium
FEAT1.4: E-mail notification of new titles of...	Medium	Proposed		Medium	Low
FEAT1.5: Highly scaleable	Low	Proposed		High	Medium
FEAT1.6: Ability to customize the web site	High	Proposed		Low	High
FEAT1.7: User registration good for future...	Medium	Incorporated		Low	Low
FEAT2: ClassicsCD Administration System	Low	Validated		High	High
FEAT2.1: Ability to add/remove offerings	High	Proposed		Low	High
FEAT2.2: Ability to check on customer orders	Medium	Incorporated		High	Medium
FEAT2.3: Maintain customer information	High	Proposed		Medium	Low
FEAT2.4: Generate reports	Medium	Incorporated		Low	Medium
FEAT3: Interactive guide to site through online Help	Low	Proposed		Medium	High
FEAT4: Item Shall Be Shipped Immediately	Medium	Proposed		Medium	Medium
* <Click here to create a requirement>	Medium	Proposed		Medium	Medium

FEAT1: ClassicsCD.com Web Shop
ClassicsCD.com Web Shop

Rational RequisitePro - Learning Project - [UC-FEAT: Use Cases Traced to Features]

File Edit View Requirement Traceability Tools Window Help

Learning Project

- Features and Vision
- Glossary
- Supplementary Requirements...
- Use Cases
 - Arrange Shipment
 - Browse Catalog
 - Check Order Status
 - CheckOut
 - Shop for CD
 - Use-Case Survey
 - Use Cases Traced to Fe...
- ClassicsCD Requirements ...

Relationships:
- direct only

FEAT1: ClassicsCD.com Web Shop

- FEAT1.1: Secure payment method
- FEAT1.2: Easy browsing
- FEAT1.3: Ability to check status of an order
- FEAT1.4: E-mail notification of new titles of...
- FEAT1.5: Highly scaleable
- FEAT1.6: Ability to customize the web site
- FEAT1.7: User registration good for future...

FEAT2: ClassicsCD Administration System

- FEAT2.1: Ability to add/remove offerings
- FEAT2.2: Ability to check on customer...
- FEAT2.3: Maintain customer information
- FEAT2.4: Generate reports

FEAT3: Interactive guide to site through...

FEAT4: Item Shall Be Shipped Immediately

UC1: Arrange shipment

UC1.1: Complete order...
Upon successful completion
of the Checkout use case...

UC1.2: Electronic Warehouse..
The Web shopping application
sends information in the form...

UC1.2.1: Name
Club member name

UC1.2.2: Shipping address
Club member shipping
address

UC1.2.3: Phone number

UC1:
FEAT1:

Ready 70 requirements

Ferramentas

- RequisitePro
 - Pros:
 - Funcionalidade banco de dados integrado
 - Relações entre requisitos rastreáveis
 - Bom tutorial
 - Excelente estrutura e modelo customizável
 - Suporta requisitos hierárquicos e acesso por toda a equipe do projeto

Ferramentas

- RequisitePro
 - Cons:
 - Precisa ser usado um banco de dados externo
 - Pode ser um exagero para alguns projectos de menor dimensão

Ferramentas

- 5W2H
 - Não exatamente uma ferramenta
 - Inicialmente utilizada para administração
 - Foi adaptada para utilizar em levantamento de requisitos (principalmente no meio ágil)

Ferramentas

- 5W2H
 - A técnica:
 1. **W**hat – O que será feito (etapas)?
 2. **W**hy – Por que será feito (justificativa)?
 3. **W**here – Onde será feito (local)?
 4. **W**hen – Quando será feito (tempo)?
 5. **W**ho – Por quem será feito (responsabilidade)?
 6. **H**ow – Como será feito (método)?
 7. **H**ow much – Quanto custará fazer (custo)?

Ferramentas

- 5W2H
 - A técnica adaptada:
 1. Qual é o objetivo do requisito?
 2. Por que o requisito deve ser implementado?
 3. Onde poderá ser visualizado o requisito?
 4. Quais são os pré-requisitos para este requisito?
 5. Qual é o perfil de acesso para este requisito?
 6. Quais são os critérios de aceitação para este requisito?
 7. Quais serão as mensagens de validação para este requisito?

Ferramentas

- Exemplo do uso 5W2H

Ferramentas

O Cliente:

- é uma rede de produtos manipulados.
- a partir da centralização dos pedidos de manipulação e da identificação dos produtos mais comumente solicitados, iniciou-se a produção e distribuição de medicamentos próprios com 20 produtos.
- possui 4 centrais de manipulação em diferentes estados, conta com uma equipe de 40 farmacêuticos e 220 técnicos de manipulação.
- produz 140 medicamentos distintos.
- venda de 22 milhões de unidades/ano.
- a área administrativa dividida em Financeira (Contas a Pagar e Contas a Receber), Recursos Humanos e Administração de Materiais (Controle de Estoque).
- as fórmulas dos produtos foram desenvolvidas pela própria empresa, detendo os direitos autorais e de produção.
- os processos de compra e venda se dá através de dois departamentos: Compras e Vendas.
- tem como objetivo estratégico tornar-se o maior laboratório de criação de fórmulas e manipulação da América Latina até 2015.

Ferramentas

Descrição do Problema:

Um dos problemas da rede está no controle de produção...

- As centrais de manipulação possuem o controle dos estoques realizado de modo manual, o que gera um alto grau de desperdício de matéria-prima.
- As centrais de manipulação utilizam matéria-prima crítica e controlada que, ocasionalmente, deixa de constar no estoque sem que o fim seja identificado.

Ferramentas

Objetivos de Negócio:

- Controlar o uso da matéria-prima ao longo do processo de manipulação.

Objetivo Específico do Sistema:

- O sistema deverá controlar a matéria-prima utilizada, identificando precisamente a necessidade de matéria-prima e controlando seu uso.

Ferramentas

1. Implementar um ambiente de controle de produção de medicamentos manipulados.
2. Para permitir a identificação precisa da necessidade de matéria-prima e controlar o seu uso eliminando o alto grau de desperdício.
3. Em um item de menu específico, conforme layout de tela Homes.
4. Ter sido identificado o fluxo de produção; Ter acesso às fórmulas dos medicamentos.
5. Responsável pelo controle da produção, por exemplo: técnico em manipulação.
6. O estoque de segurança para matérias-primas críticas deverá ser de 5 dias de produção e para matérias prima simples deverá ser de 3 dias de produção; O sistema deverá sinalizar a necessidade de compra de matéria-prima quando tiver estoque para produção para, pelo menos, 7 dias úteis, sem considerar o prazo do estoque de segurança; O sistema não deverá disparar um aviso de compra de matéria-prima crítica se não houver produção de medicamentos que utilizem este tipo de insumo nos últimos 30 dias.
7. Para o critério de aceitação: O sistema deverá sinalizar a necessidade de compra de matéria-prima quando tiver estoque para produção para, pelo menos, 7 dias úteis, sem considerar o prazo do estoque de segurança.

A mensagem de validação deverá ser:

"Há estoque de matéria-prima X para produção de medicamento durante 7 dias úteis. Deseja efetuar novo pedido de compra? Sim/Não"

Ferramentas

- Planilha de Requisitos

RFXXX – Título do Requisito				
Especificação funcional/não funcional detalhada: Descrição da especificação funcional/não funcional				
Regras de negócio: Regras que o requisito deve seguir				
Prioridade	Complexidade	Status	Versão	Autor
Alta, média ou baixa	Alta, média ou baixa	Status: Avaliação, Execução, etc	Versão: Nome dos Autores Atual	

Ferramentas

- Planilha de Requisitos

RFXXX – Título do Requisito				
Especificação funcional/não funcional detalhada: Descrição da especificação funcional/não funcional				
Regras de negócio: Regras que o requisito deve seguir				
Prioridade	Complexidade	Status	Versão	Autor
Alta, média ou baixa	Alta, média ou baixa	Status: Avaliação, Execução, etc	Versão: Nome dos Autores Atual	

Retomando

- Requisitos:
 - Requisitos são objetivos ou restrições estabelecidas por clientes e usuários que definem as suas diversas propriedades do sistema. Os requisitos de software são, obviamente, aqueles dentre os requisitos de sistema que dizem respeito a propriedades do software.
 - São separados em requisitos funcionais e requisitos não-funcionais

Retomando

- Requisitos Funcionais:
 - Os requisitos funcionais são a descrição das diversas funções que clientes e usuários querem ou precisam que o software ofereça. Eles definem a funcionalidade desejada do software. O termo função é usado no sentido genérico de operação que pode ser realizada pelo sistema, seja através comandos dos usuários ou seja pela ocorrência de eventos internos ou externos ao sistema.

Retomando

- Requisitos Funcionais:
 - São exemplos de requisitos funcionais:
 - "o software deve possibilitar o cálculo dos gastos diários, semanais, mensais e anuais com pessoal".
 - "o software deve emitir relatórios de compras a cada quinze dias"
 - "os usuários devem poder obter o número de aprovações, reprovações e trancamentos em todas as disciplinas por um determinado período de tempo.

Retomando

- Requisitos Funcionais:
 - A especificação de um requisito funcional deve determinar o que se espera que o software faça, sem a preocupação de como ele faz. É importante diferenciar a atividade de especificar requisitos da atividade de especificação que ocorre durante o design do software. No design do software deve-se tomar a decisão de quais as funções o sistema efetivamente terá para satisfazer àquilo que os usuários querem.

Retomando

- Requisitos Não-Funcionais:
 - Requisitos não-funcionais são as qualidades globais de um software, como manutenibilidade, usabilidade, desempenho, custos e várias outras. Normalmente estes requisitos são descritos de maneira informal, de maneira controversa (por exemplo, o gerente quer segurança mas os usuários querem facilidade de uso) e são difíceis de validar.

Retomando

- Requisitos Não-Funcionais:
 - São exemplos de requisitos não-funcionais:
 - "a base de dados deve ser protegida para acesso apenas de usuários autorizados".
 - "o tempo de resposta do sistema não deve ultrapassar 30 segundo".
 - "o software deve ser operacionalizado no sistema Linux"
 - "o tempo de desenvolvimento não deve ultrapassar seis meses".

Exemplos

RF001 – Cadastro de clientes

Especificação funcional detalhada:

Esta funcionalidade deverá permitir a inserção de novos clientes. Os próprios clientes poderão selecionar a opção de se cadastrar no site informando: nome, CPF, RG, endereço padrão, data de nascimento, e-mail e senha.

Regras de negócio:

- Os campos nome, CPF, RG, data de nascimento e e-mail serão obrigatórios;
- Usuário necessita ter idade maior ou igual a 18 anos para poder se cadastrar;
- Serão validados os dados do cliente para verificar a integridade dos mesmos;
- A senha deverá possuir dígitos numéricos, letras e caracteres especiais – tendo no mínimo 8 caracteres;
- Os campos CPF e RG devem ser únicos (verificação para CPFs repetidos já cadastrados).

Prioridade	Complexidade	Status	Versão	Autor
------------	--------------	--------	--------	-------

Alta	Média	Avaliação	1	Carlos Eduardo, Cristian, Nathália e Rômulo.
------	-------	-----------	---	--

RF002 – Edição de clientes

Especificação funcional detalhada:

Esta funcionalidade deverá permitir a edição dos clientes já cadastrados. Os próprios clientes poderão selecionar a opção de edição, podendo modificar os campos: endereço padrão, e-mail e senha.

Regras de negócio:

- Cumprir com as regras de negócio do RF001 (se aplicáveis).

Prioridade	Complexidade	Status	Versão	Autor
Média	Baixa	Avaliação	1	Carlos Eduardo, Cristian, Nathália e Rômulo.

RF003 – Exclusão de clientes

Especificação funcional detalhada:

Esta funcionalidade deverá permitir a deleção de clientes. Os próprios clientes, se desejarem, poderão deletar seu cadastro.

Regras de negócio:

- É obrigatório estar logado no sistema e informar a senha para confirmação de exclusão de cadastro;
- Exibir mensagem de confirmação antes da exclusão.

Prioridade	Complexidade	Status	Versão	Autor
------------	--------------	--------	--------	-------

Baixa	Baixa	Avaliação	1	Carlos Eduardo, Cristian, Nathália e Rômulo.
-------	-------	-----------	---	--

RF004 – Consulta de clientes

Especificação funcional detalhada:

Permitir a consulta de clientes cadastrados no sistema mostrando uma listagem dos resultados. Os parâmetros dos filtros poderão ser por: cidade, estado, país, data de nascimento.

Regras de negócio:

- Os dados da consulta serão exibidos em ordem alfabética;
- Somente o administrador do sistema terá acesso a esta funcionalidade.

Prioridade	Complexidade	Status	Versão	Autor
Baixa	Baixa	Avaliação	1	Carlos Eduardo, Cristian, Nathália e Rômulo.

Exemplos

RF005 – Listagem dos produtos similares

Especificação funcional detalhada:

Serão exibidos ao cliente produtos similares aos já comprados pelo mesmo ou que estão sendo consultados no momento.

Regras de negócio:

- Serão considerados similares os produtos que compartilharem: marca, modelo e categoria;
- Serão exibidos quatro produtos similares por vez, com a opção de paginação para consultar mais produtos similares. Sendo no máximo vinte produtos.

Prioridade	Complexidade	Status	Versão	Autor
------------	--------------	--------	--------	-------

Média	Alta	Avaliação	1	Carlos Eduardo, Cristian, Nathália e Rômulo.
-------	------	-----------	---	--

RF006 – Avaliação de produtos

Especificação funcional detalhada:

Após a entrega de um produto, o cliente receberá um e-mail convidando-o a avaliar o produto recém adquirido. O produto poderá ser avaliado em diferentes quesitos, tais como: tamanho do calçado, conforto, resistência, flexibilidade – sendo levada em conta a categoria.

Regras de negócio:

- Só será possível avaliação do produto que o cliente comprou;
- O cliente terá no máximo quinze dias para avaliar o produto;
- Será mantido o anonimato do avaliador.

Prioridade	Complexidade	Status	Versão	Autor
------------	--------------	--------	--------	-------

Baixa	Média	Avaliação	1	Carlos Eduardo, Cristian, Nathália e Rômulo.
-------	-------	-----------	---	--

RF007 – Listagem de avaliação de produtos

Especificação funcional detalhada:

A listagem da avaliação de um produto será exibida sempre que o mesmo estiver sendo consultado. De forma que o cliente poderá verificar as notas específicas nos diferentes quesitos (RF006). A avaliação também será exibida na tela de seleção dos produtos (listagem dos produtos). Porém neste caso será mostrada uma avaliação geral do produto.

Regras de negócio:

- A listagem da avaliação de cada produto e para os diferentes quesitos será dada por pontos. De forma que cem pontos será a melhor avaliação e zero a pior. Sendo feita uma média entre todos os quesitos para gerar a avaliação geral do produto.

Prioridade	Complexidade	Status	Versão	Autor
------------	--------------	--------	--------	-------

Baixa	Média	Avaliação	1	Carlos Eduardo
-------	-------	-----------	---	----------------

RF008 – Envio de boleto por e-mail

Especificação funcional detalhada:

A funcionalidade de envio de boleto por e-mail se dará assim que o cliente efetivar a compra. Caso ele escolha a opção de pagamento por boleto bancário, o mesmo será enviado para o e-mail cadastrado no seu perfil de cliente.

Regras de negócio:

- Só estarão disponíveis os boletos para os bancos que tiverem integração com o sistema;
- Uma cópia do boleto será salva para eventuais dúvidas/problemas;
- O cliente deverá notificar o sistema com uma confirmação do recebimento do e-mail, caso contrário, em futuras compras, não estará disponível a opção de pagamento por boleto com envio por e-mail.

Prioridade	Complexidade	Status	Versão	Autor
------------	--------------	--------	--------	-------

Alta	Média	Avaliação	1	Cristian
------	-------	-----------	---	----------

Exemplos

RNF001

O sistema terá uma interface amigável ao usuário primário sem se tornar cansativa aos usuários mais experientes.

Deve ser acessível a pessoas com deficiência visual. Seguir os padrões da W3C

Prioridade	Complexidade	Status	Versão	Autor
Alta	Alta	Elaboração	1	Nathália

RNF002

O sistema deverá ser compatível com diversos navegadores disponíveis (e com suas versões): Internet Explorer 6+, Safari, Opera, Mozilla Firefox 3+, Chrome

Prioridade	Complexidade	Status	Versão	Autor
Alta	Média	Elaboração	1	Carlos Eduardo

RNF003

Para mais acessibilidade, o sistema deverá ser portável a Tablets e Smartphones. Aumentando a abrangência de clientes e acessos.

Prioridade	Complexidade	Status	Versão	Autor
Média	Média	Elaboração	1	Carlos Eduardo

RNF004

O sistema deverá estar integrado com o sistema atual de estoque para que seja visível ao cliente se o produto que ele está consultando está disponível em estoque ou não. A integração tem como objetivo também a atualização do estoque no momento que alguma compra for efetuada.

Prioridade	Complexidade	Status	Versão	Autor
Alta	Alta	Elaboração	1	Carlos Eduardo

Exercícios

- Descrever Requisitos CRUD de usuário para uma rede social
- Descrever 2 Requisitos Não-Funcionais

Casos de Uso

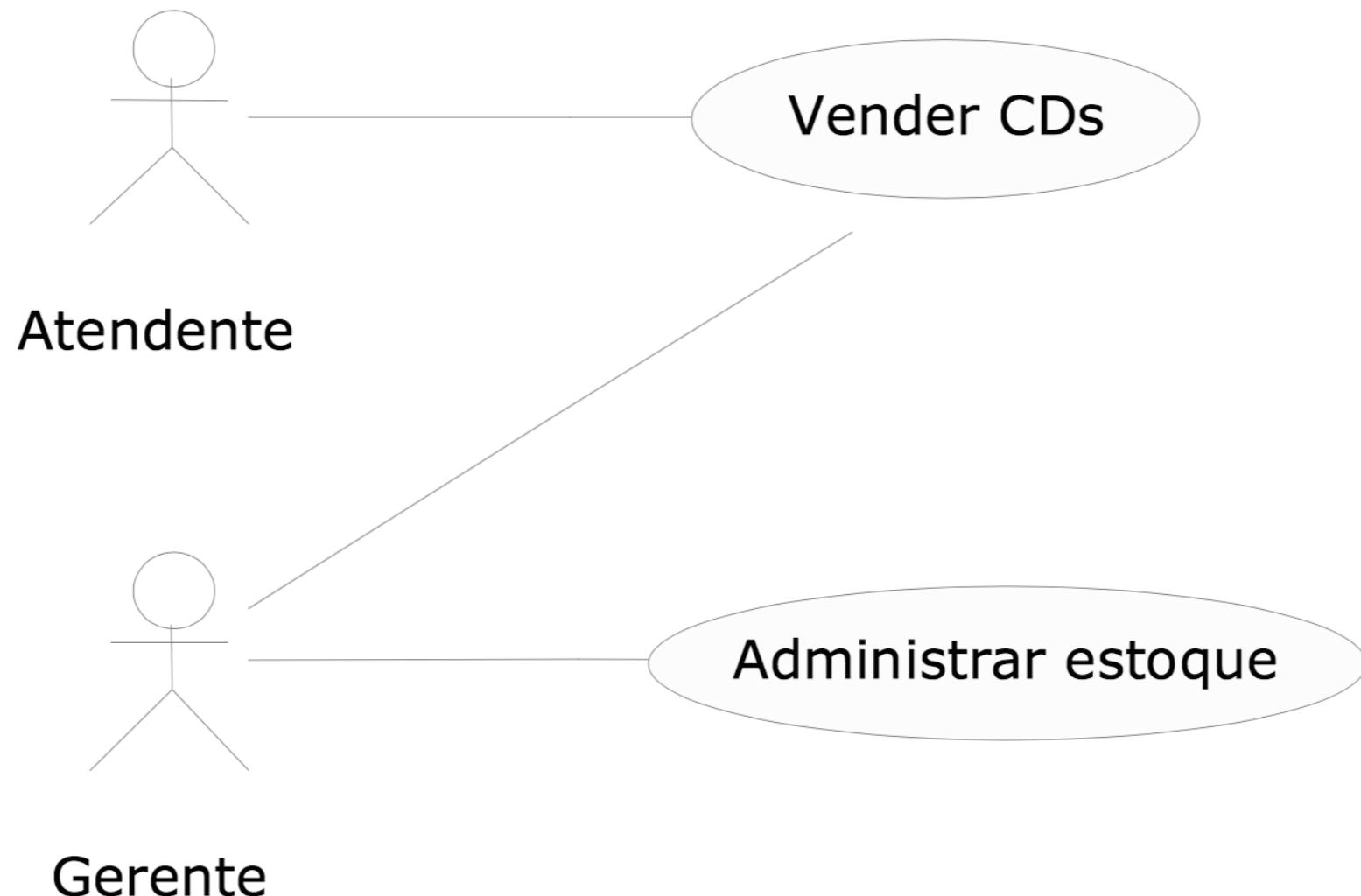
- Diagrama e descrição de casos de Uso
 - Técnica de modelagem de requisitos
 - Descreve o que o sistema faz
- Segundo Ivan Jacobson , podemos dizer que um caso de uso é um "documento narrativo que descreve a sequência de eventos de um ator que usa um sistema para completar um processo".

Casos de Uso

- Descrevem como os usuários interagem com o sistema (as funcionalidades do sistema)
- Dão uma visão externa do sistema
- O conjunto de casos de uso deve ser capaz de comunicar a funcionalidade e o comportamento do sistema para o cliente
- **Descrevem o que o sistema faz, mas NÃO especificam como isso deve ser feito**

Casos de Uso

- Exemplo de diagrama:



Descrição de um Caso de Uso

VENDER CDs - CASO DE USO

NOME

Vender CDs

DESCRÍÇÃO SUCINTA

Atendente vende um ou mais CDs a um usuário.

ATORES

1. Atendente

PRÉ-CONDIÇÕES

1. Ter executado o caso de uso "CDU000_Validar Senha"

FLUXO BÁSICO

1. O Atendente seleciona a opção "Vender CDs".
2. O Sistema exibe a lista de CDs.
3. O Atendente seleciona os CDs, informando as respectivas quantidades.
4. O Sistema exibe a lista de clientes.
5. O Atendente seleciona o cliente.
6. O Atendente seleciona a opção "Vender".
7. O Sistema exibe as informações da venda: CDs, quantidades e o cliente.
8. O Atendente confirma as informações da venda.
9. O Sistema efetua a venda, verificando a regra RN1.
 - 9.1. O Atendente seleciona o tipo de venda "A Prazo" ou "À Vista"
 - 9.2. O Sistema deve executar o caso de uso "CDU001a_Vender CDs a prazo" ou o caso de uso "CDU001b_Vender CDs à vista", de acordo com a opção selecionada pelo atendente no passo anterior.
 - 9.3. O Sistema atualiza o estoque de acordo com a regra RN2.
10. O Sistema emite a Nota Fiscal conforme ED1.
11. O caso de uso é encerrado.

FLUXOS ALTERNATIVOS

(A1) Alternativa ao Passo 4 – Cliente não cadastrado

- 1.a O Atendente seleciona a opção "Cadastrar Cliente"
- 1.b O Sistema executa o caso de uso "CDU002_Cadastrar Cliente"
- 1.c O Sistema retoma ao Passo 4.

(A2) Alternativa ao Passo 8 – Informações Incorretas

- 2.a O Atendente não confirma as informações da venda.
- 2.b O Sistema retoma ao Passo 2.

(A3) Alternativa ao Passo 9 – A regra RN1 não é atendida

- 3.a O Sistema exibe a mensagem "Não há produtos disponíveis em estoque."
- 3.b O caso de uso é encerrado.

Diagrama de Casos de Uso

- Elementos do diagrama:
 - Atores
 - Casos de Uso
 - Relacionamentos
 - Associação
 - Generalização
 - Dependência: Inclusão ou Exclusão
 - Fronteira do Sistema

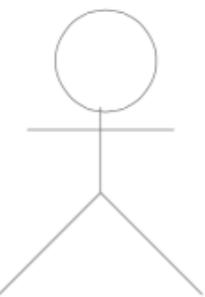
Diagrama de Casos de Uso

- Atores:
 - Representam os papéis desempenhados por elementos externos ao sistema
 - humano (usuário), dispositivo de hardware ou outro sistema (cliente)
 - Elementos que interagem com o sistema

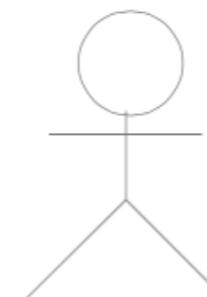
Diagrama de Casos de Uso

- Atores:

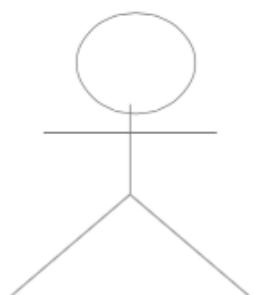
Notação:



Secretária



Diretor



Sistema de
Relatórios

Diagrama de Casos de Uso

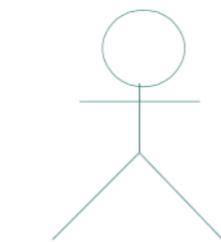
- Exemplo: Loja de CDS
 - Identificando os atores
 - Uma loja de CDs possui discos para venda. Um cliente pode comprar uma quantidade ilimitada de discos para isto ele deve se dirigir à loja. A loja possui um atendente cuja função é atender os clientes durante a venda dos discos. A loja também possui um gerente cuja função é administrar o estoque para que não faltem discos. Além disso é ele quem dá folga ao atendente, ou seja, ele também atende os clientes durante a venda dos discos.

Diagrama de Casos de Uso

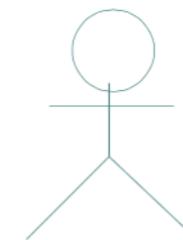
- Exemplo: Loja de CDS
 - Identificando os atores
 - Uma loja de CDs possui discos para venda. Um cliente pode comprar uma quantidade ilimitada de discos para isto ele deve se dirigir à loja. A loja possui um **atendente** cuja função é atender os clientes durante a venda dos discos. A loja também possui um **gerente** cuja função é administrar o estoque para que não faltem discos. Além disso é ele quem dá folga ao atendente, ou seja, ele também atende os clientes durante a venda dos discos.

Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando os atores



Gerente



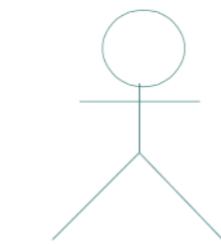
Atendente

E o cliente?

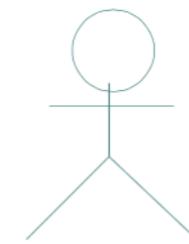
– Não é ator pois ele não interage com o sistema!

Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando os atores



Gerente



Atendente

E o cliente?

- Não é ator pois ele não interage com **ESTE** sistema!

Diagrama de Casos de Uso

- Caso de Uso
 - Representa uma funcionalidade do sistema (um requisito funcional)
 - É iniciado por um ator ou por outro caso de uso
 - Usualmente nomeia-se um caso de uso iniciando por um verbo

Notação:



Nome do Caso de Uso

Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando os casos de uso
 - Uma loja de CDs possui discos para venda. Um cliente pode comprar uma quantidade ilimitada de discos para isto ele deve se dirigir à loja. A loja possui um atendente cuja função é atender os clientes durante a **venda dos discos**. A loja também possui um gerente cuja função é **administrar o estoque** para que não faltem discos. Além disso é ele quem dá folga ao atendente, ou seja, ele também atende os clientes durante a **venda dos discos**.

Diagrama de Casos de Uso

- Exemplo: Loja de CDs
 - Identificando os casos de uso

Vender CDs

Administrar estoque

Diagrama de Casos de Uso

- Relacionamentos de associação:
 - Indica que há uma interação (comunicação) entre um caso de uso e um ator
 - Um ator pode se comunicar com vários casos de uso
 - Associações **não** representam fluxo de informação

Diagrama de Casos de Uso

- Relacionamentos de associação:

Notação:

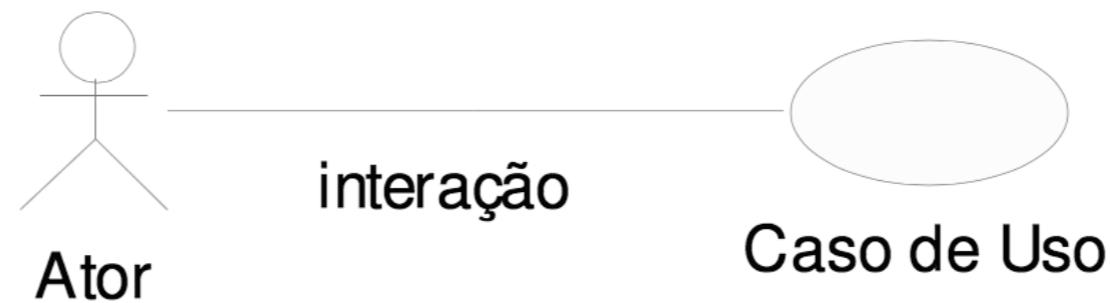


Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando os relacionamentos de associação
 - Uma loja de CDs possui discos para venda. Um cliente pode comprar uma quantidade ilimitada de discos para isto ele deve se dirigir à loja. A loja possui um **atendente** cuja função é atender os clientes durante a venda dos discos. A loja também possui um **gerente** cuja função é **administrar o estoque** para que não faltem discos. Além disso é ele quem dá folga ao atendente, ou seja, ele também atende os clientes durante a venda dos discos.

Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando os relacionamentos de associação

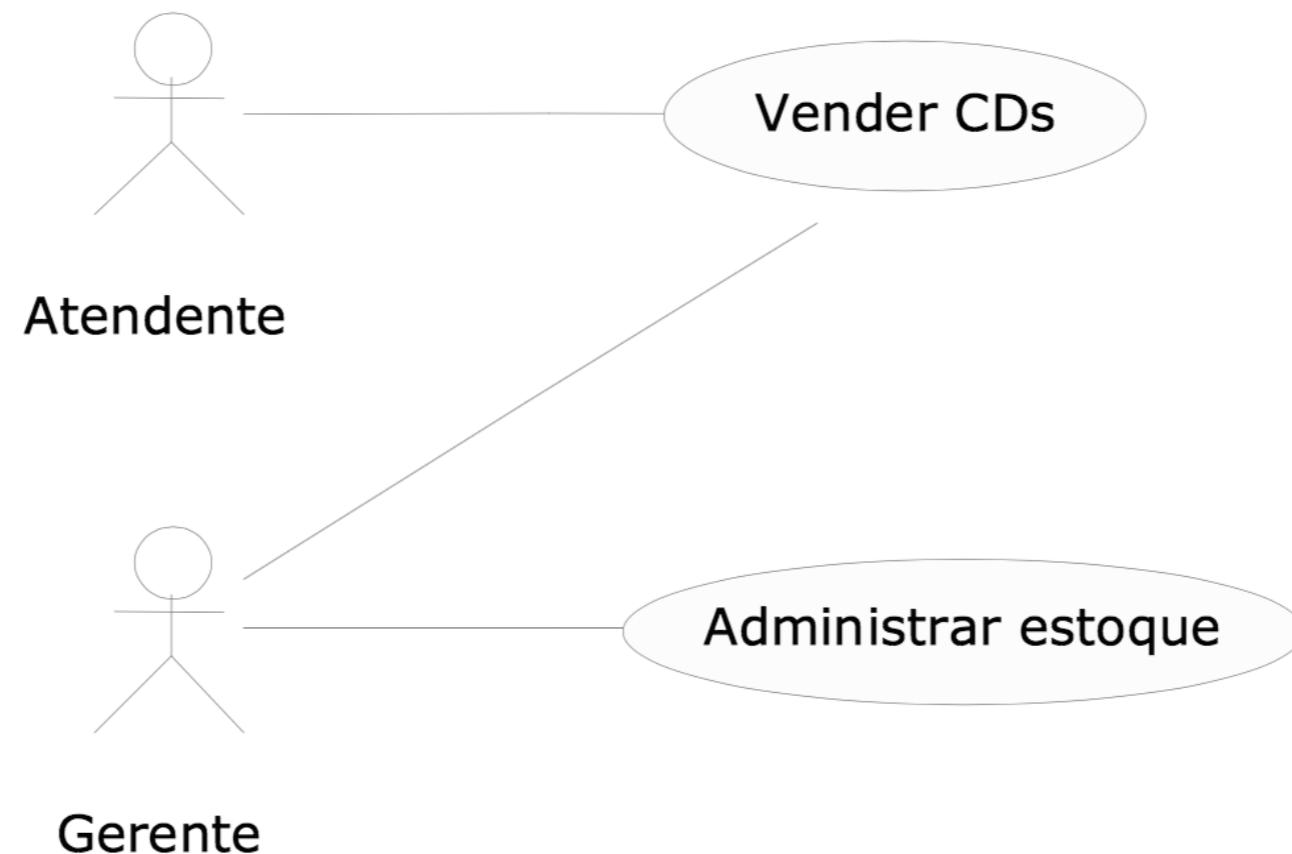


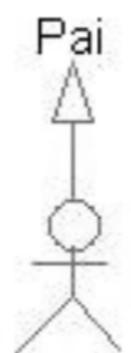
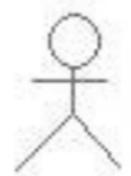
Diagrama de Casos de Uso

- Relacionamentos de generalização:
 - Generalização de atores
 - Quando dois ou mais atores podem se comunicar com o mesmo conjunto de casos de uso
 - Um filho (herdeiro) pode se comunicar com todos os casos de uso que seu pai se comunica.

Diagrama de Casos de Uso

- Relacionamentos de generalização:
 - Generalização de atores

Notação:



Filho

Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando generalização de atores

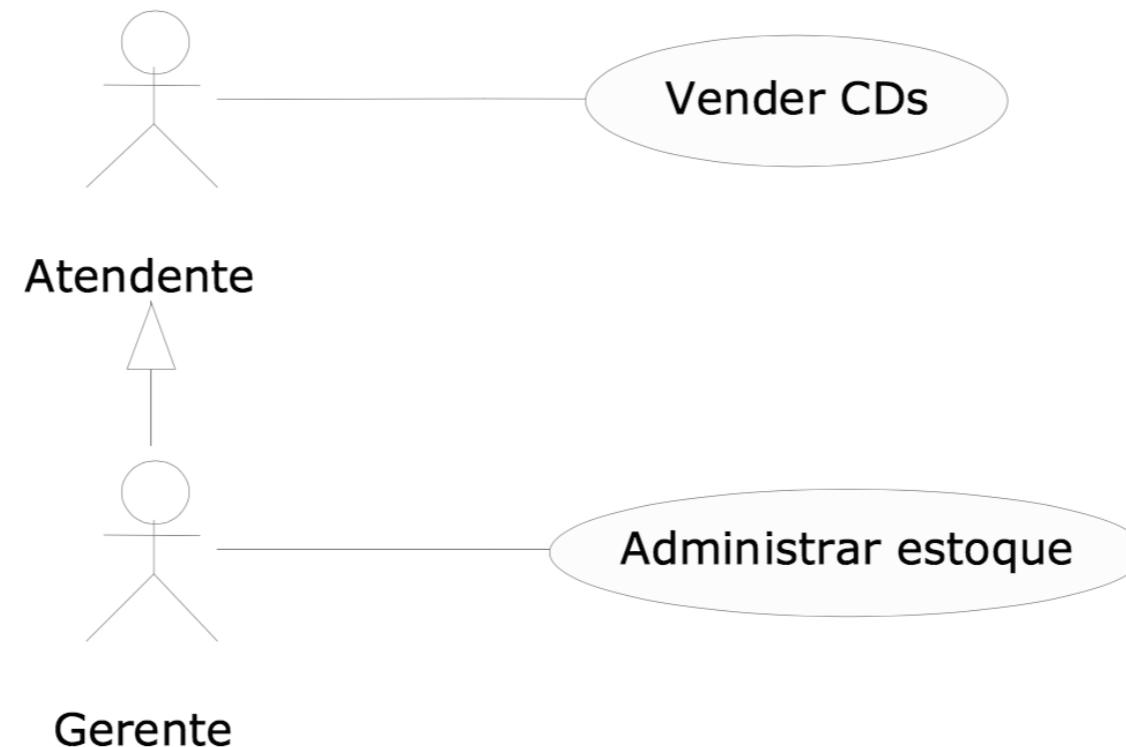


Diagrama de Casos de Uso

- Relacionamentos de generalização:
 - Generalização de casos de uso
 - O caso de uso filho herda o comportamento e o significado do caso de uso pai
 - O caso de uso filho pode incluir ou sobrescrever o comportamento do caso de uso pai
 - O caso de uso filho pode substituir o caso de uso pai em qualquer lugar que ele apareça
 - Deve ser aplicada quando uma condição resulta na definição de diversos fluxos alternativos.

Diagrama de Casos de Uso

- Relacionamentos de generalização:
 - Generalização de casos de uso

Notação:

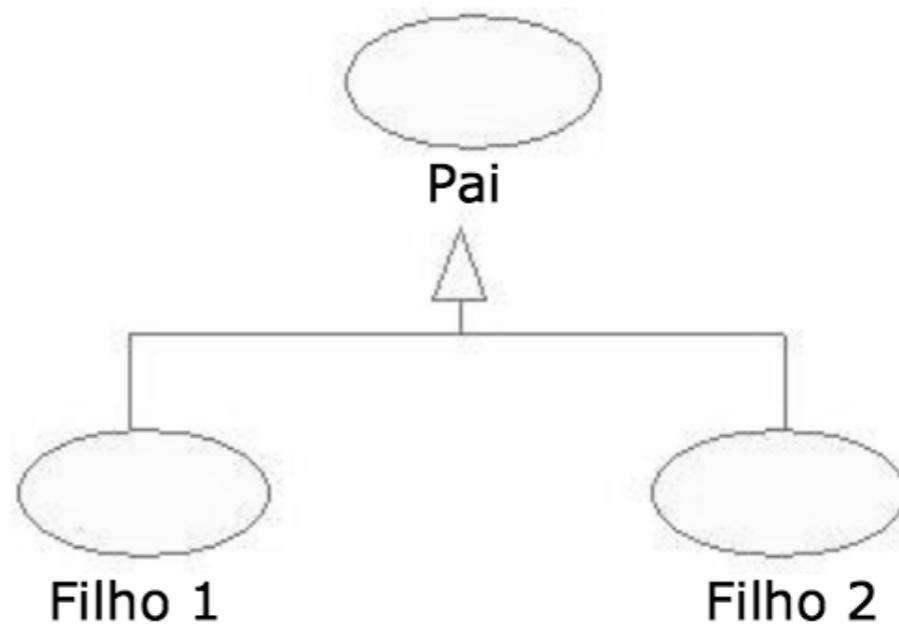


Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando generalização de casos de uso
 - NOVOS REQUISITOS: As vendas podem ser à vista ou a prazo. Em ambos os casos o estoque é atualizado e uma nota fiscal, entregue ao consumidor.
 - No caso de uma venda à vista, clientes cadastrados na loja e que compram mais de 5 CDs de uma só vez ganham um desconto de 1% para cada ano de cadastro.
 - No caso de uma venda a prazo, ela pode ser parcelada em 2 pagamentos com um acréscimo de 20%. As vendas a prazo podem ser pagas no cartão ou no boleto. Para pagamento com boleto, são gerados boletos bancários que são entregues ao cliente e armazenados no sistema para lançamento posterior no caixa. Para pagamento com cartão, os clientes com mais de 10 anos de cadastro na loja ganham o mesmo desconto das compras a vista.

Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando generalização de casos de uso

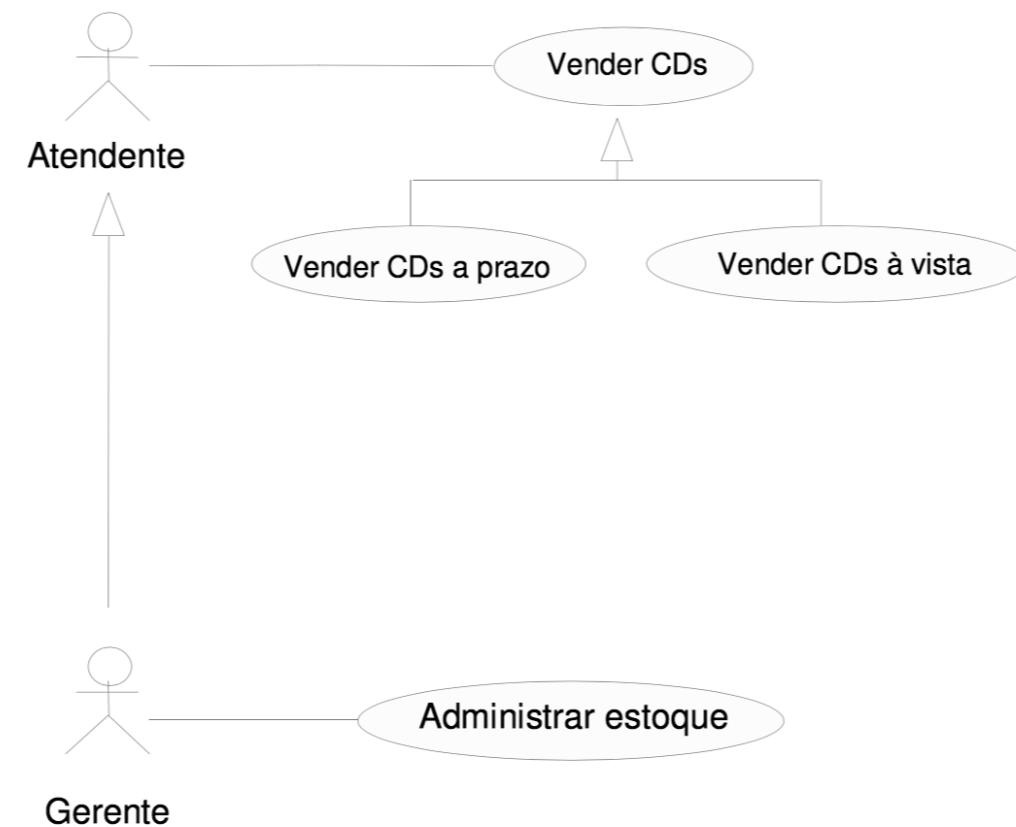


Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando MAIS generalizações de casos de uso
 - NOVOS REQUISITOS: As vendas podem ser **à vista** ou **a prazo**. Em ambos os casos o estoque é atualizado e uma nota fiscal, entregue ao consumidor.
 - No caso de uma venda à vista, clientes cadastrados na loja e que compram mais de 5 CDs de uma só vez ganham um desconto de 1% para cada ano de cadastro.
 - No caso de uma venda a prazo, ela pode ser parcelada em 2 pagamentos com um acréscimo de 20%. As vendas a prazo podem ser pagas no **cartão** ou no **boleto**. Para pagamento com boleto, são gerados boletos bancários que são entregues ao cliente e armazenados no sistema para lançamento posterior no caixa. Para pagamento com cartão, os clientes com mais de 10 anos de cadastro na loja ganham o mesmo desconto das compras a vista.

Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando generalização de casos de uso

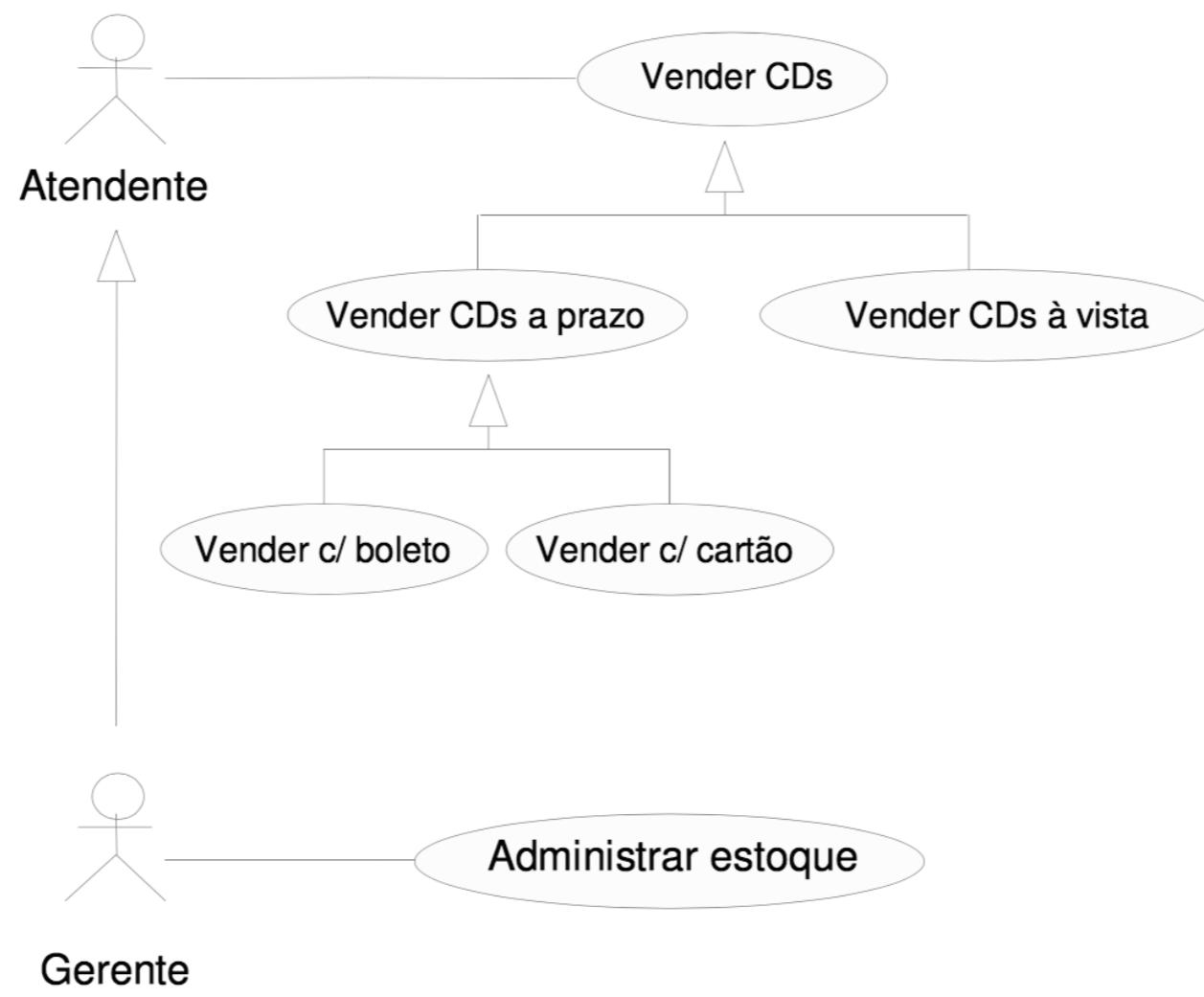


Diagrama de Casos de Uso

- Relacionamentos de dependência:
 - Extensão
 - Representa uma variação/extensão do comportamento do caso de uso base
 - O caso de uso estendido só é executado sob certas circunstâncias
 - Separa partes obrigatórias de partes opcionais
 - Partes obrigatórias: caso de uso base
 - Partes opcionais: caso de uso estendido

Diagrama de Casos de Uso

- Relacionamentos de dependência:
 - Extensão

Notação: <<extends>>



Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando dependência: extensão
 - NOVOS REQUISITOS:
 - No caso de uma venda à vista, clientes cadastrados na loja e que compram mais de 5 CDs de uma só vez ganham um **desconto** de 1% para cada ano de cadastro.
 - No caso de uma venda a prazo... Para pagamento com cartão, os clientes com mais de 10 anos de cadastro na loja ganham o mesmo **desconto** das compras à vista.

Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando dependência: extensão

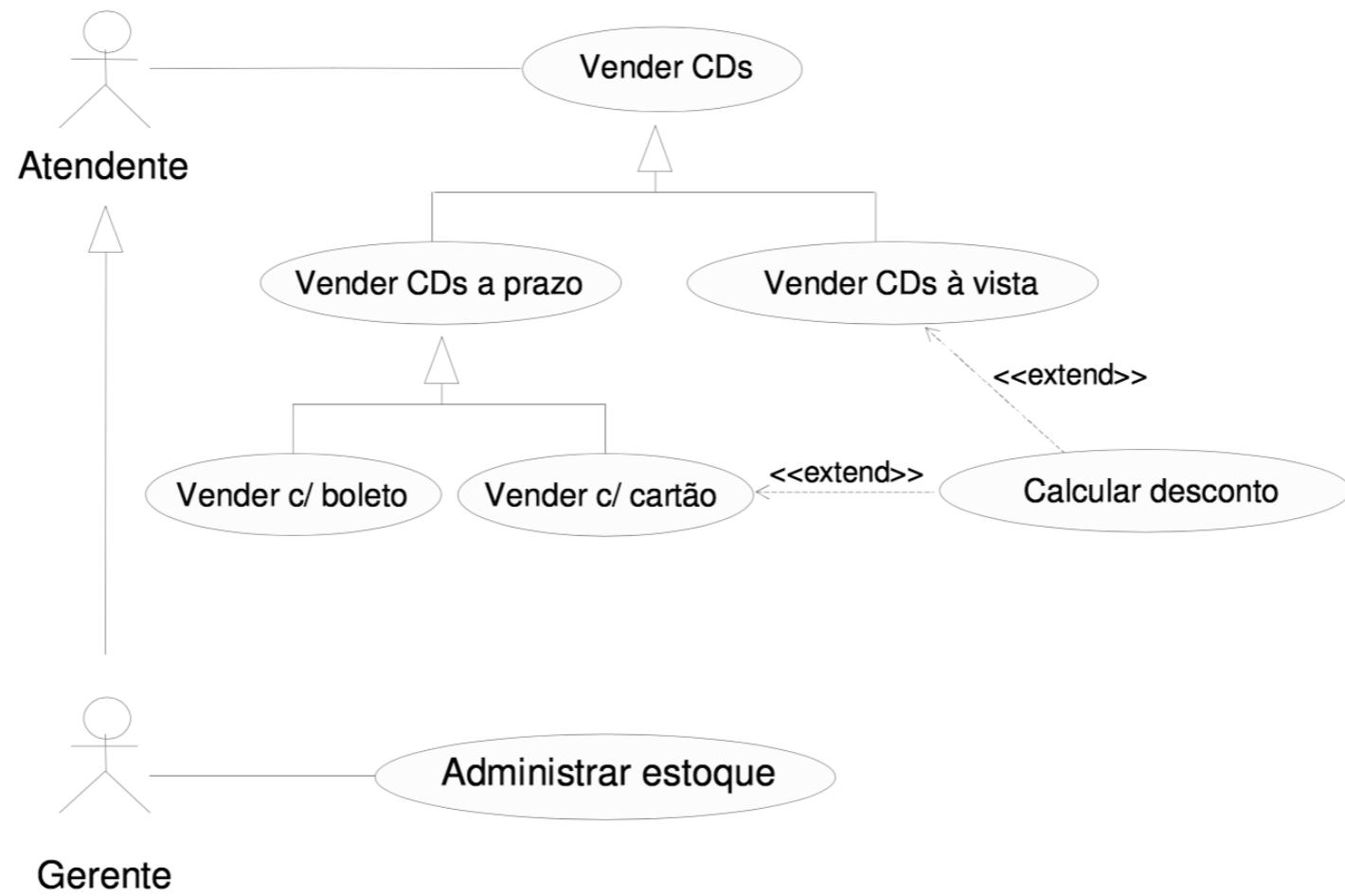


Diagrama de Casos de Uso

- Relacionamentos de dependência:
 - Inclusão
 - Evita repetição ao fatorar uma atividade comum a dois ou mais casos de uso
 - Um caso de uso pode incluir vários casos de uso

Notação: <<includes>>



Diagrama de Casos de Uso

- Exemplo: Loja de CDS
 - Identificando dependência: inclusão
 - NOVOS REQUISITOS:
 - Para efetuar vendas ou administrar estoque, atendentes e gerentes terão que **validar** suas respectivas senhas de acesso ao sistema.

Diagrama de Casos de Uso

- Exemplo: Loja de CDs
 - Identificando dependência: inclusão

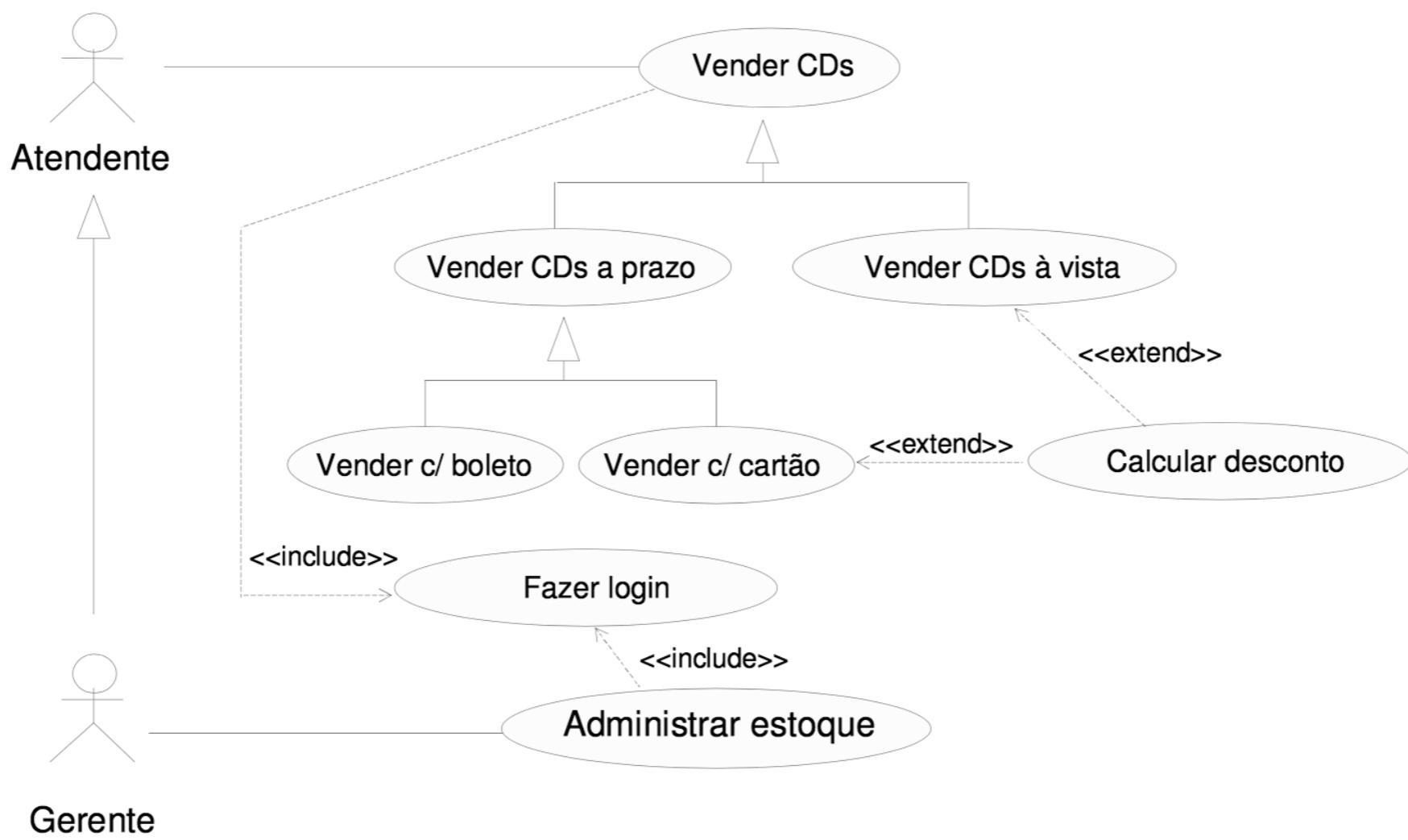


Diagrama de Casos de Uso

- Fronteira do Sistema
 - Serve para definir a área de atuação do sistema

Notação:

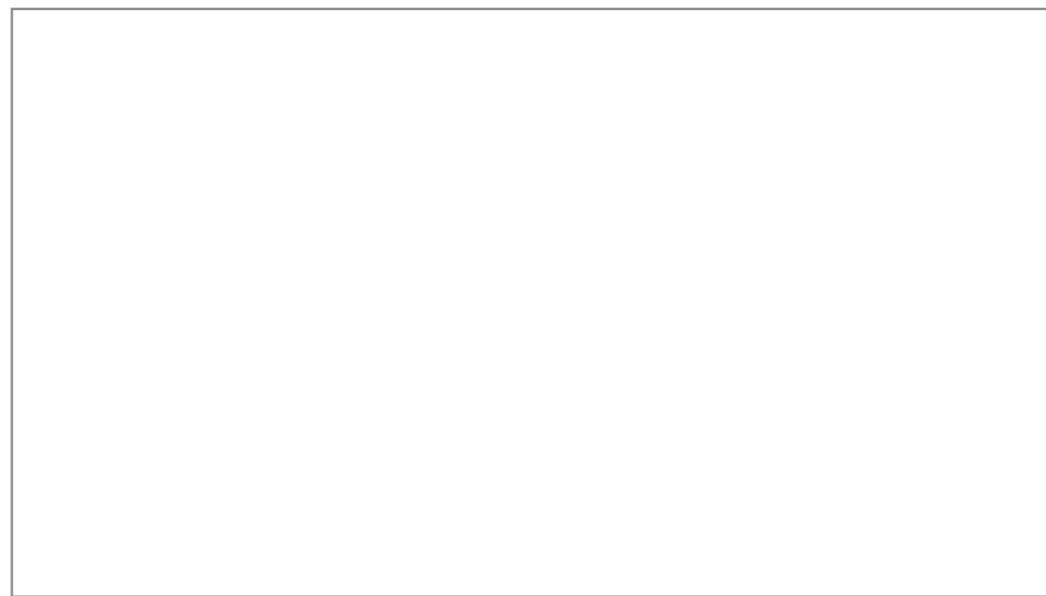
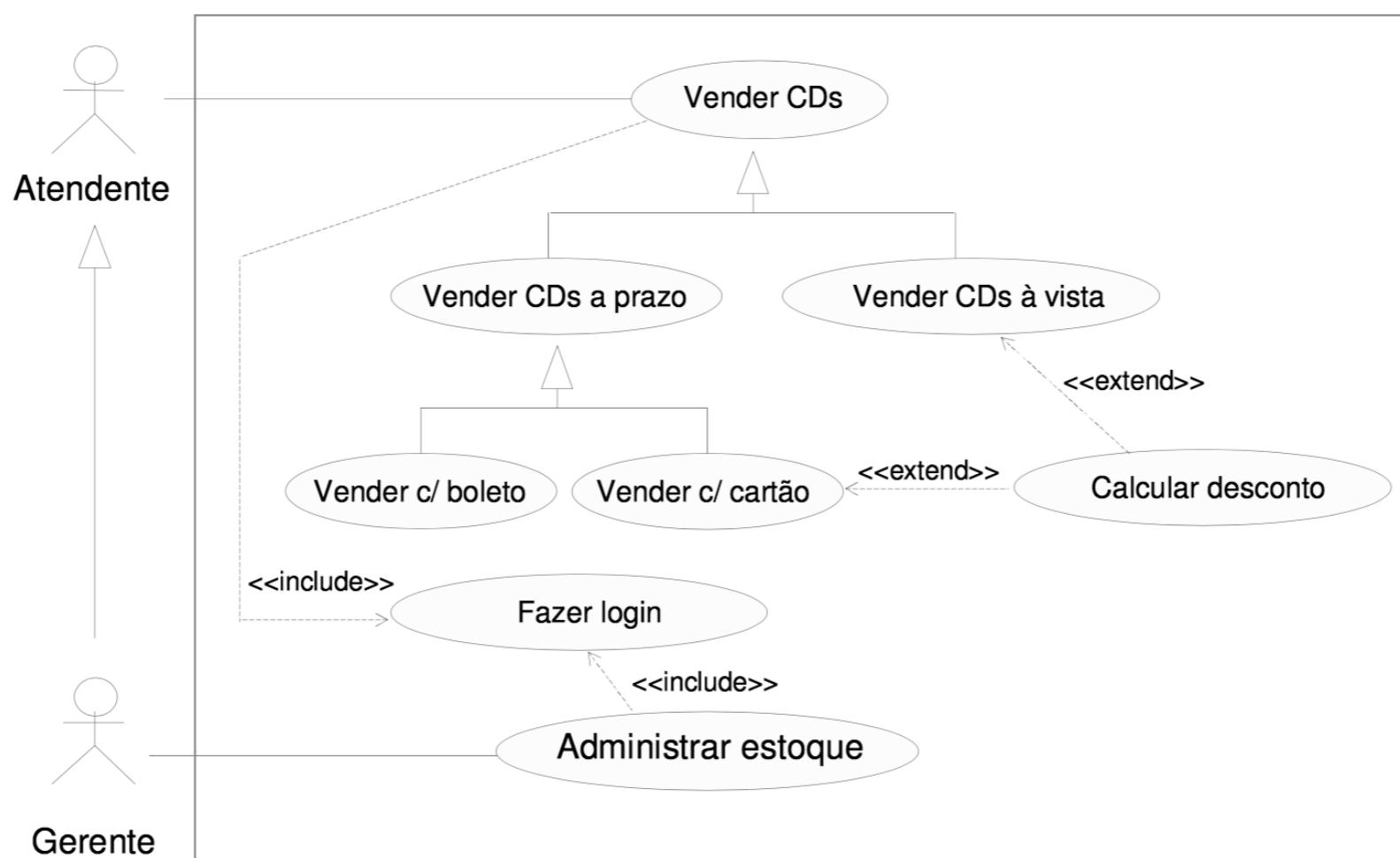


Diagrama de Casos de Uso

- Exemplo: Loja de CDs
 - Identificando a fronteira do sistema



Descrição de Casos de Uso

- A descrição é mais importante do que o diagrama
- UML não especifica padrão
- Pode ser:
 - Informal
 - Típica
 - Detalhada

Descrição de Casos de Uso

- Descrição Informal
 - Contém o nome do caso de uso uma descrição textual de sua funcionalidade

Caso de Uso 01 – Cadastrando Cliente (Descrição informal)

O Cliente inicia o cadastro preenchendo a ficha cadastral e enviando a documentação necessária para o dep. de Cadastro. O Assistente de Cadastro examina a documentação enviada. Estando a documentação em ordem, o Gerente de Cadastro valida os dados da ficha cadastral e marca o cliente como aprovado.

Se houverem problemas com os documentos enviados, o Assistente de Cadastro informa documentação irregular. O Cliente envia a documentação regularizada para o Assistente de Cadastro.

Se houverem problemas com os dados da ficha cadastral, o Gerente de Cadastro informa irregularidade dados cadastrais. O Cliente corrige os dados cadastrais.

Descrição de Casos de Uso

- Descrição Típica
- Contém:
 - Identificação do ator que iniciou o caso de uso
 - Pré-requisitos (se houver) do caso de uso
 - Descrição textual do:
 - Fluxo normal
 - Fluxos alternativos (se houver)

Descrição de Casos de Uso

- Descrição Típica

Caso de Uso 01 – Cadastrando Cliente (Descrição típica)

Ator Primário: Cliente

Precondições: Nenhuma

Fluxo Normal

- 1 – Cliente preenche ficha cadastral,
- 2 – Assistente de Cadastro informa recebimento documentação cadastral
- 3 – Gerente de Cadastro informa aprovação de Cliente

Fluxo Alternativo: documentação incompleta ou com erro

- 2a – Assistente de Cadastro Informa documentação irregular.
 - 2b – Cliente envia documentação corrigida para cadastro.
- Retoma ao passo 2.

Fluxo Alternativo: irregularidade nos dados cadastrais

- 3a – Gerente de Cadastro informa irregularidade dados cadastrais
- 3b – Cliente atualiza dados cadastrais.
- 3c - Retorna ao passo 3.

Descrição de Casos de Uso

- Descrição Detalhada
- Contém:
 - Nome
 - Descrição sucinta
 - Atores
 - Pré-condições
 - Pós-condições
 - Fluxo básico
 - Fluxos alternativos
 - Fluxos de exceção
 - Estruturas de dados
 - Regras de negócio
 - Observações

VENDER CDs - CASO DE USO

NOME

Vender CDs

Descrição Sucinta

Atendente vende um ou mais CDs a um usuário.

Atores

1. Atendente

Pré-Condições

1. Ter executado o caso de uso "CDU000_Validar Senha"

Fluxo Básico

1. O Atendente seleciona a opção "Vender CDs".
2. O Sistema exibe a lista de CDs.
3. O Atendente seleciona os CDs, informando as respectivas quantidades.
4. O Sistema exibe a lista de clientes.
5. O Atendente seleciona o cliente.
6. O Atendente seleciona a opção "Vender".
7. O Sistema exibe as informações da venda: CDs, quantidades e o cliente.
8. O Atendente confirma as informações da venda.
9. O Sistema efetua a venda, verificando a regra RN1.
 - 9.1. O Atendente seleciona o tipo de venda "A Prazo" ou "À Vista".
 - 9.2. O Sistema deve executar o caso de uso "CDU001a_Vender CDs a prazo" ou o caso de uso "CDU001b_Vender CDs à vista", de acordo com a opção selecionada pelo atendente no passo anterior.
 - 9.3. O Sistema atualiza o estoque de acordo com a regra RN2.
10. O Sistema emite a Nota Fiscal conforme ED1.
11. O caso de uso é encerrado.

Fluxos Alternativos

(A1) Alternativa ao Passo 4 – Cliente não cadastrado

- 1.a O Atendente seleciona a opção "Cadastrar Cliente".
- 1.b O Sistema executa o caso de uso "CDU002_Cadastrar Cliente".
- 1.c O Sistema retorna ao Passo 4.

(A2) Alternativa ao Passo 8 – Informações Incorretas

- 2.a O Atendente não confirma as informações da venda.
- 2.b O Sistema retorna ao Passo 2.

(A3) Alternativa ao Passo 9 – A regra RN1 não é atendida

- 3.a O Sistema exibe a mensagem "Não há produtos disponíveis em estoque.".3.b O caso de uso é encerrado.

Estrutura de Dados

(ED1) Nota Fiscal

- 1.1. CPF do cliente
- 1.2. Nome do cliente
- 1.3. Endereço do cliente
- 1.4. CNPJ da loja
- 1.5. Razão social da loja
- 1.6. Endereço da loja
- 1.7. Data da compra
- 1.8. Código dos produtos comprados
- 1.9. Descrição dos produtos comprados
- 1.10. Valores dos produtos comprados
- 1.11. Valor total da compra
- 1.12. Valor do desconto
- 1.13. Valor final da compra

Regras de Negócio

(RN1) O produto deve estar disponível em estoque.

(RN2) O sistema deve atualizar o estoque de produtos, i.e., para cada produto selecionado para venda, o sistema deve subtrair a quantidade vendida da quantidade disponível em estoque.

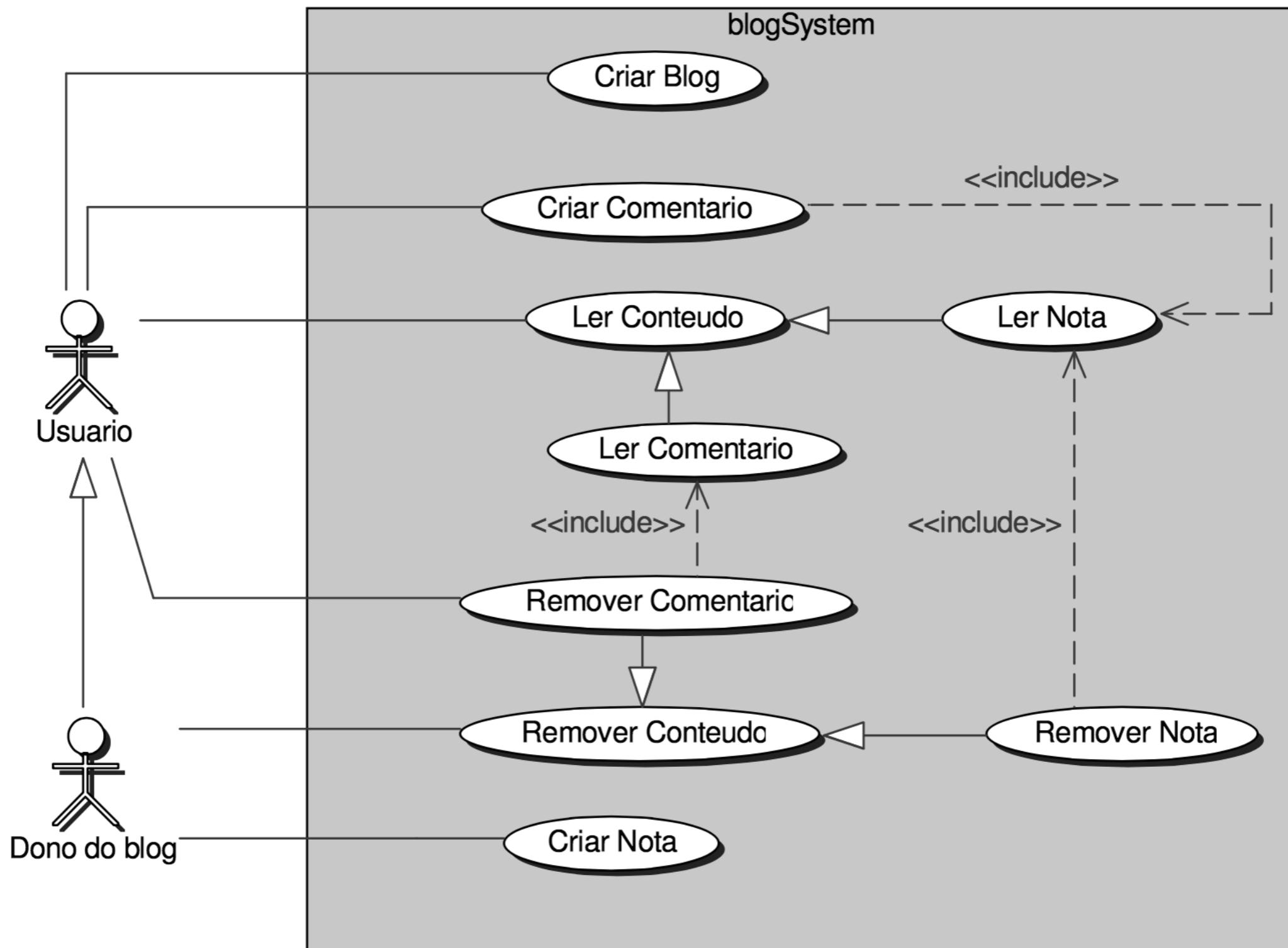
Exemplo

- BLOG:
 - Um blog tem um título e uma data de criação e além disso é um conjunto de conteúdos.
 - Estes conteúdos (mensagens) podem ser notas ou comentários sobre as notas. Tanto notas quanto comentários têm características comuns como o texto e a data de sua criação.
 - Todo usuário possui:
 - E-mail (deve ser único, ou seja, não há mais de um usuário com o mesmo e-mail)

Exemplo

- BLOG:
 - O sistema deve:
 - Permitir a criação de blogs
 - Permitir a utilização de blogs
 - Qualquer usuário pode ler conteúdos
 - Somente o dono do blog pode criar notas
 - Qualquer usuário pode criar comentários. Para criar um comentário o usuário precisa ler as notas.
 - Somente o dono do blog pode remover conteúdos. Para remover um conteúdo ele precisará ler o conteúdo. Caso ele remova um comentário, o autor do comentário deve ser notificado por e-mail.

Exemplo



Ferramentas

- Omundo
- Jude
- Together
- IBM Rational Rose
- Violet
- Astah Community

Diagrama de Atividades

- É o diagrama com maior ênfase ao nível de algoritmo da UML e provavelmente um dos mais detalhistas.
- A partir da UML 2.0 tornou-se um diagrama totalmente independente.

Diagrama de Atividades

- Apresenta muitas semelhanças com os antigos fluxogramas.
- Este diagrama preocupa-se em descrever os passos a serem percorridos para a conclusão de um método ou algoritmo específico e não um processo completo como é o diagrama de seqüência.

Diagrama de Atividades

- Possui três estados obrigatórios:
 - Estado Inicial
 - Estado Final
 - Estado de Ação

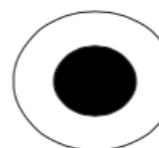
Diagrama de Atividades

- Elementos do Diagrama:
 - Estados Iniciais e Finais
 - Todos os diagramas de atividades possuem pelo menos um estado inicial e um estado final (podem haver vários)
 - Estado inicial indica o início do processo
 - Estado final indica o fim do processo

NOTAÇÃO:



Estado Inicial



Estado Final

Diagrama de Atividades

- Elementos do Diagrama:
 - Atividades
 - Retângulos com bordas arredondadas que representam as atividades
 - Ação que deve ser feita
 - Quando finalizada transfere a execução para a próxima atividade (transição)

NOTAÇÃO:

Solicitar Produto

Diagrama de Atividades

- Elementos do Diagrama:
 - Transições
 - Setas contínuas que representam fluxo de trabalho de uma atividade para outra
 - Caminho a ser seguido para conclusão do processo



Diagrama de Atividades

- Elementos do Diagrama:
 - Decisões
 - Losango utilizado para controlar os desvios do fluxo de controle
 - Ramificação: uma entrada e duas saídas
 - Mesclar: duas entradas e uma saída
 - Utiliza-se uma expressão lógica

NOTAÇÃO:



Diagrama de Atividades

- Elementos do Diagrama:
 - Bifurcação e União
 - Barra sólida usada para atividades paralelas
 - Bifurcação: divisão do fluxo de controle
 - União: sincronização das atividades

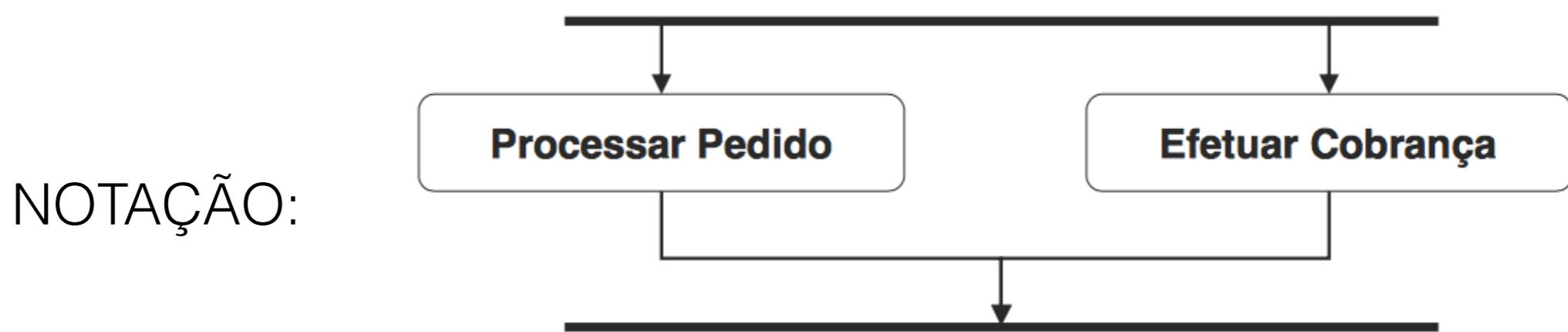
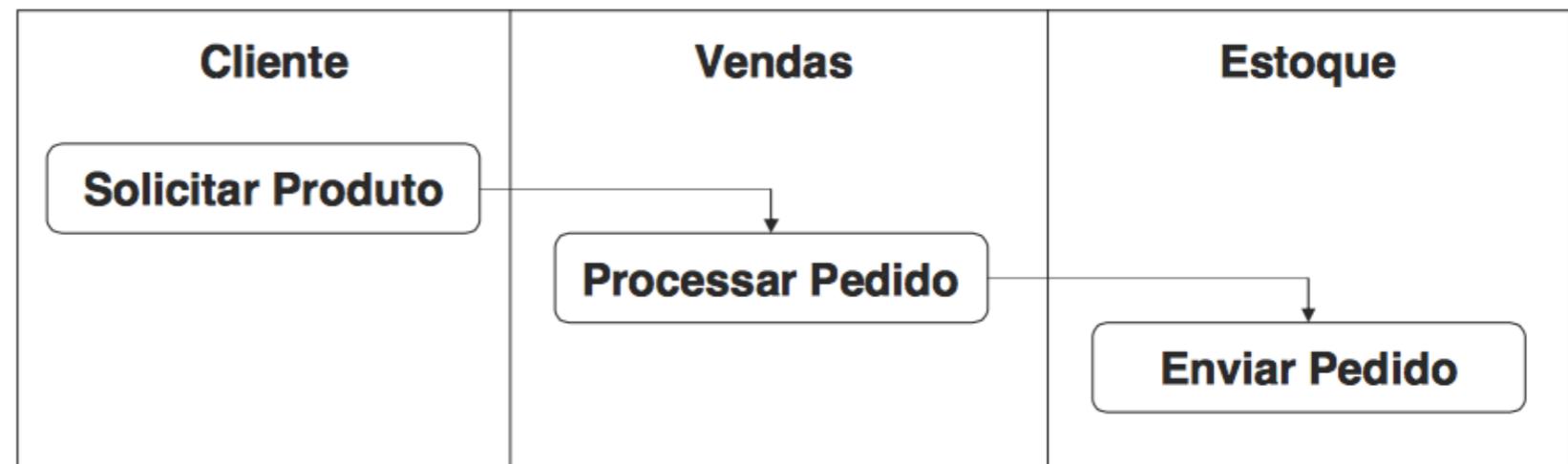


Diagrama de Atividades

- Elementos do Diagrama:
 - Raias
 - São uma forma de organização lógica das atividades
 - Podem estar associadas a objetos, componentes do sistema ou a atores

NOTAÇÃO:



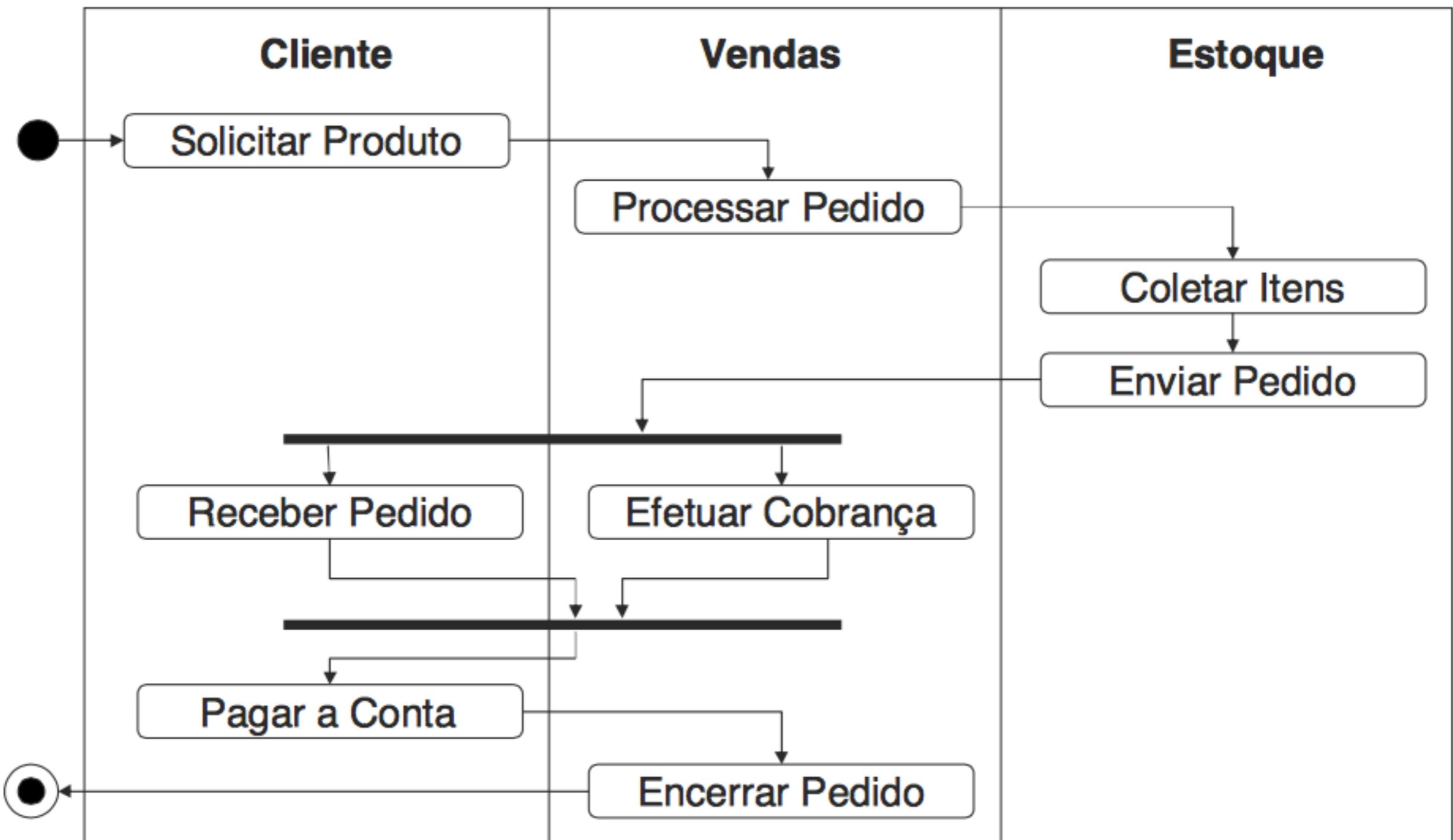


Diagrama de Atividades

- Fluxos de Controle
 - Quando uma ação está completa, o fluxo de controle passa imediatamente à próxima ação.
 - O fluxo é especificado utilizando setas de fluxo para mostrar o caminho de uma ação seguinte.

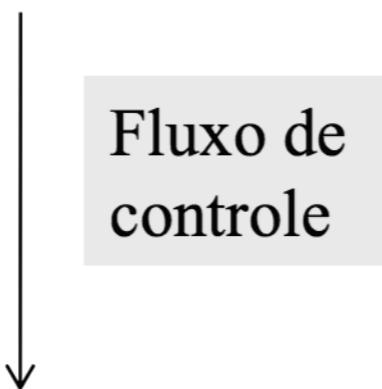


Diagrama de Atividades

- Ponto de Decisão
- Representa um ponto do fluxo de controle onde deve ser realizado um teste, uma tomada de decisão
- As transições geradas por um Ponto de Decisão necessitam ser providas de uma Condição de Guarda (texto entre colchetes) para determinar qual a condição do teste.

Diagrama de Atividades

- Exemplo - Ponto de Decisão

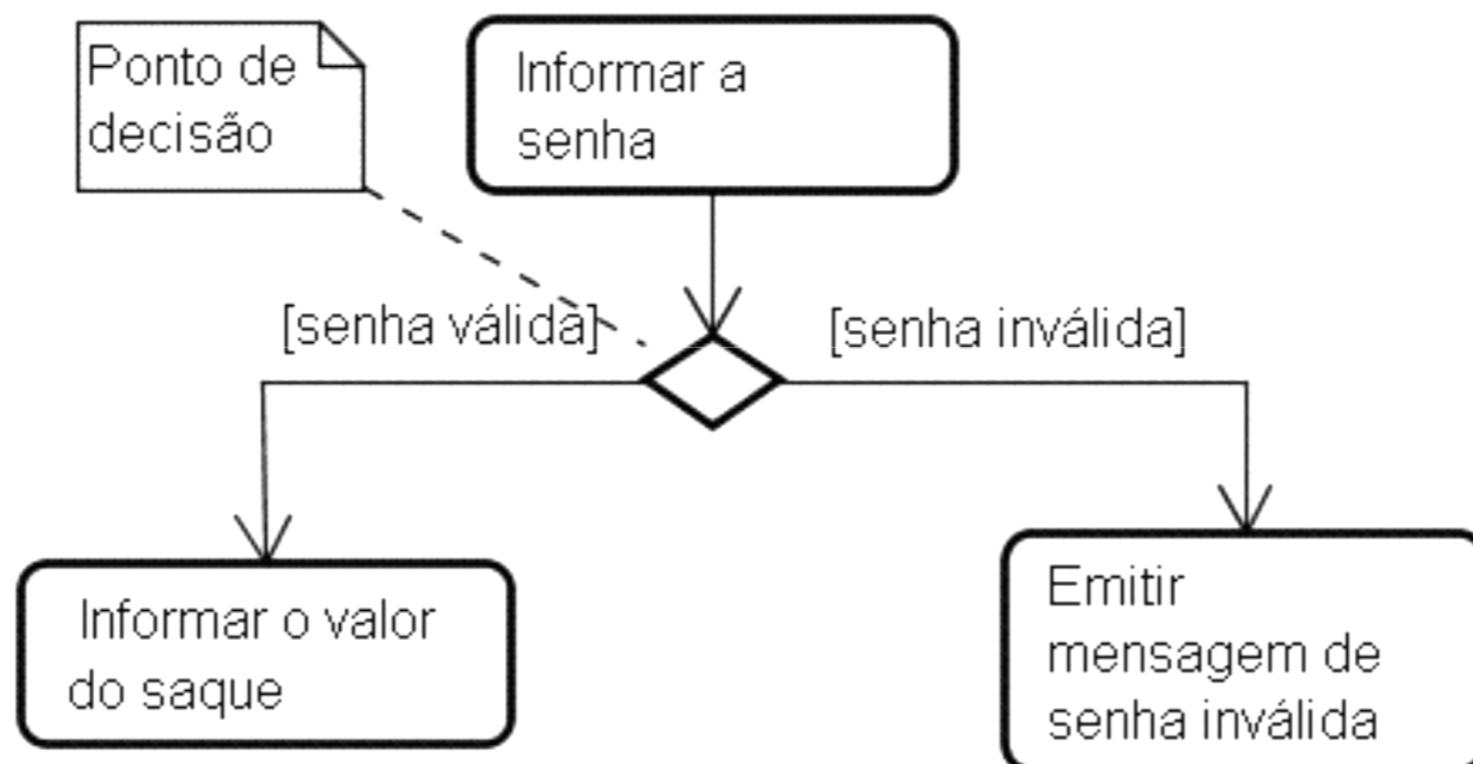


Diagrama de Atividades

- Exemplo - Diagrama de Atividades

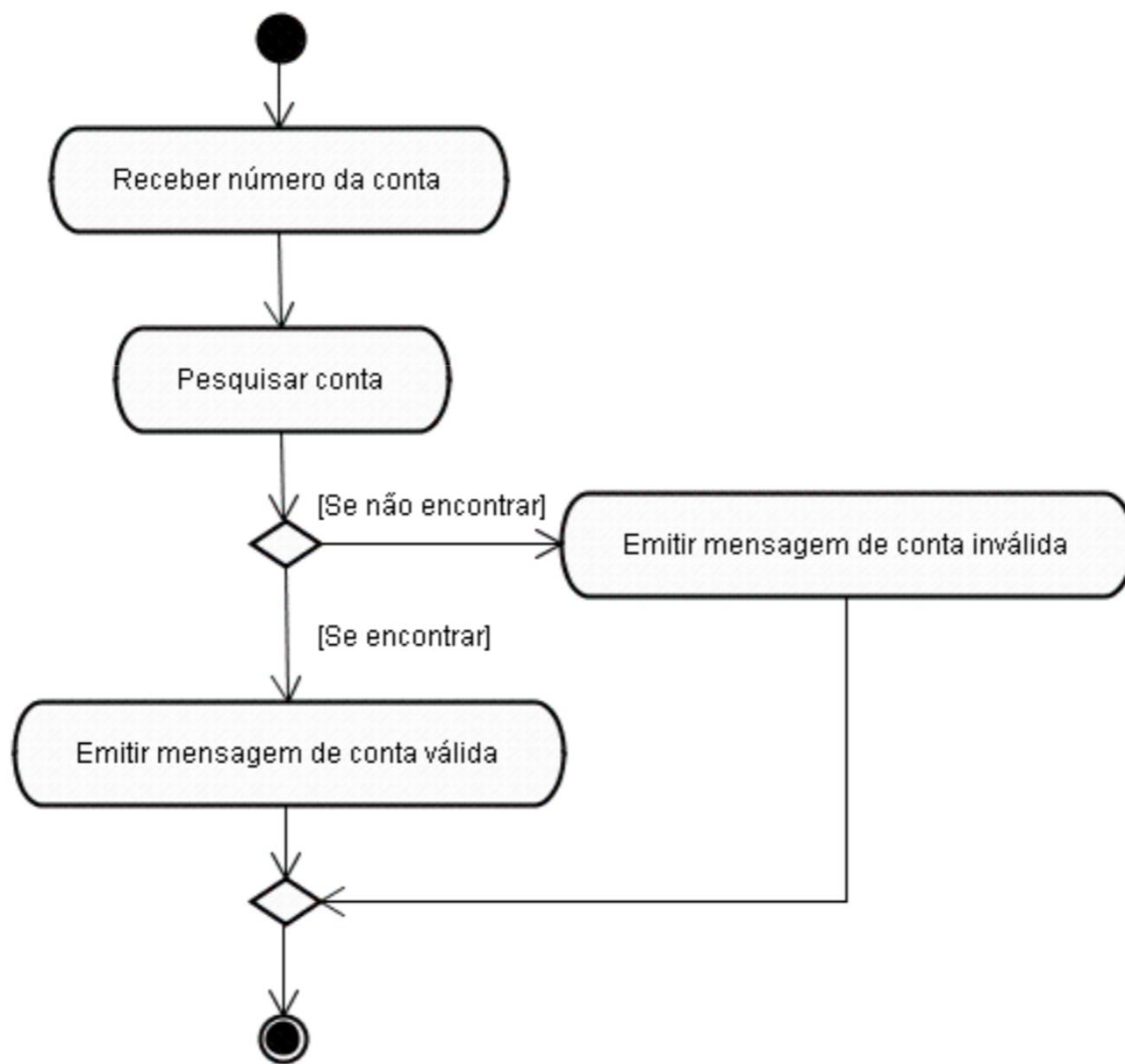


Diagrama de Atividades

- Casos de Utilização do Diagrama de Atividades
 - Modelagem dos processos do negócio
 - Modelagem da lógica de um caso de uso
 - Modelagem da lógica de uma operação complexa

Diagrama de Atividades

- Modelagem dos processos do negócio
 - O processo de negócio também é um processo de entendimento
 - As vezes os modelos são construídos para melhorar o entendimento de um determinado problema
 - Nesse caso, o enfoque está em entender o comportamento do sistema no decorrer de diversos casos de uso

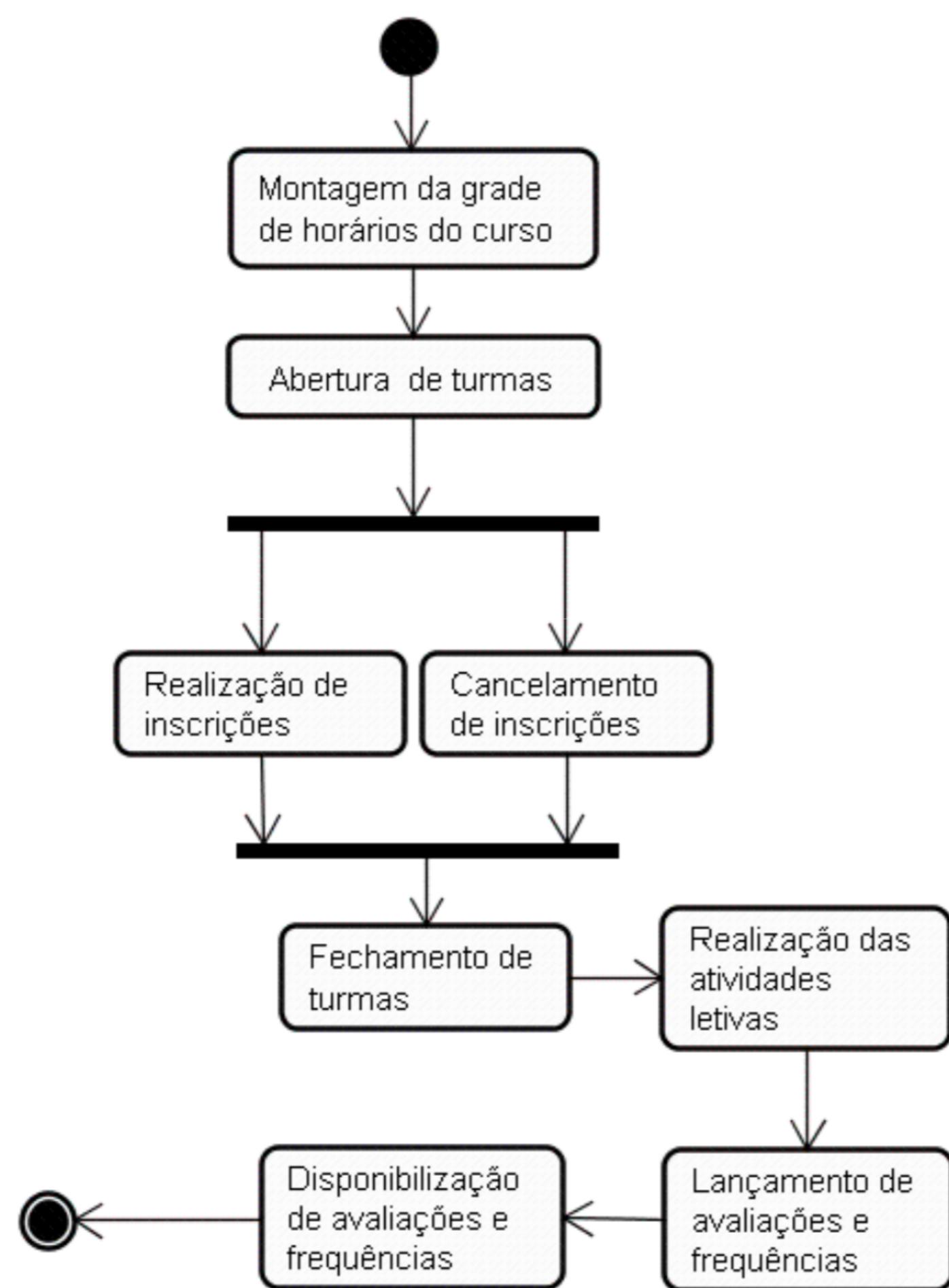


Diagrama de Atividades

- Modelagem da lógica de um caso de uso
 - Na descrição de um caso de uso, não há uma sintaxe clara para indicar decisões, iterações e fluxos executados em paralelo. É comum utilizar frases como “O passo P ocorre até que a condição C seja verdadeira” ou “Vai para o passo 9 do Fluxo Principal”.
 - Nessas situações, é interessante complementar a especificação do caso de uso com um diagrama de atividades.
 - O diagrama de atividades deve ser usado para complementar a especificação e não para substituí-la.

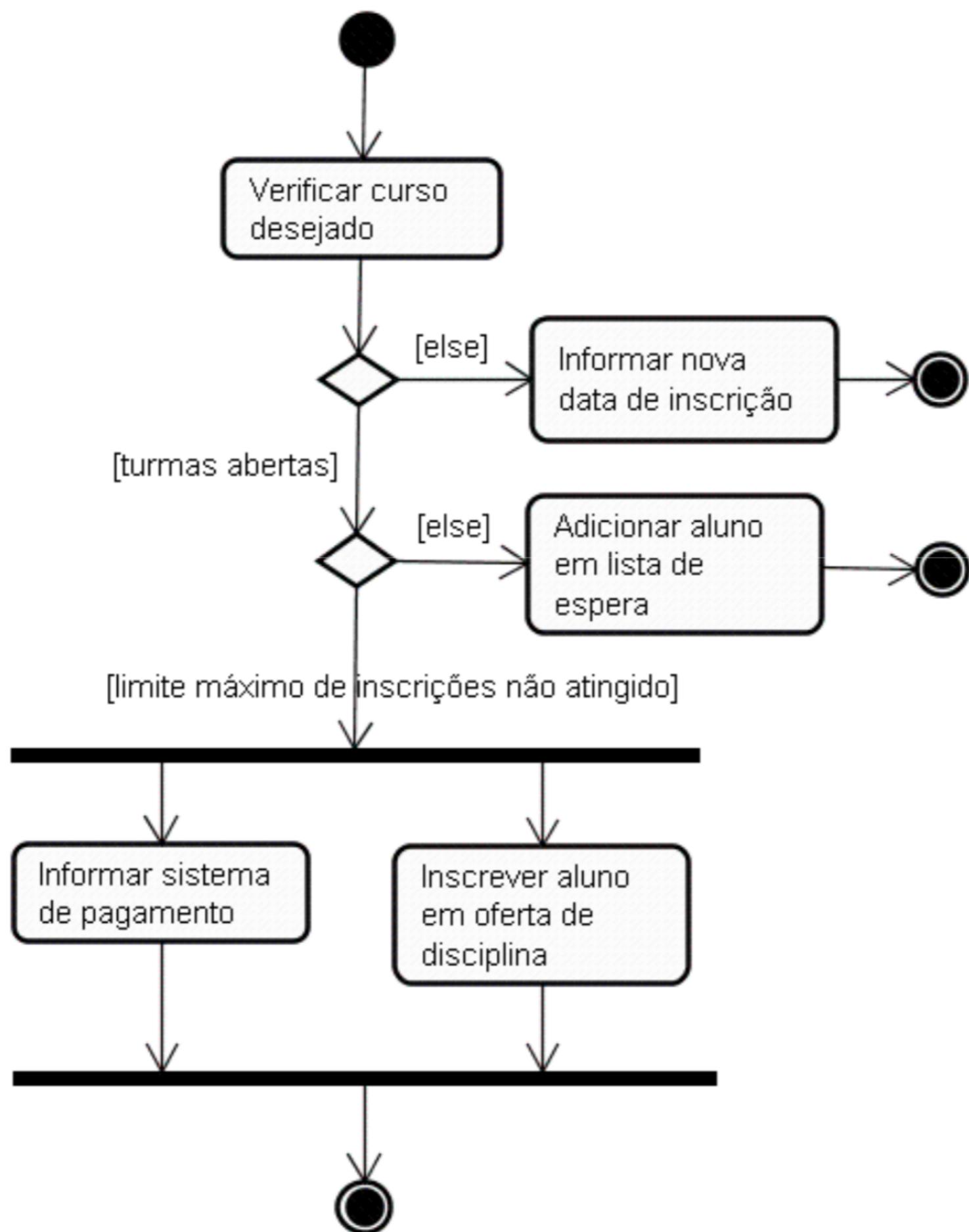
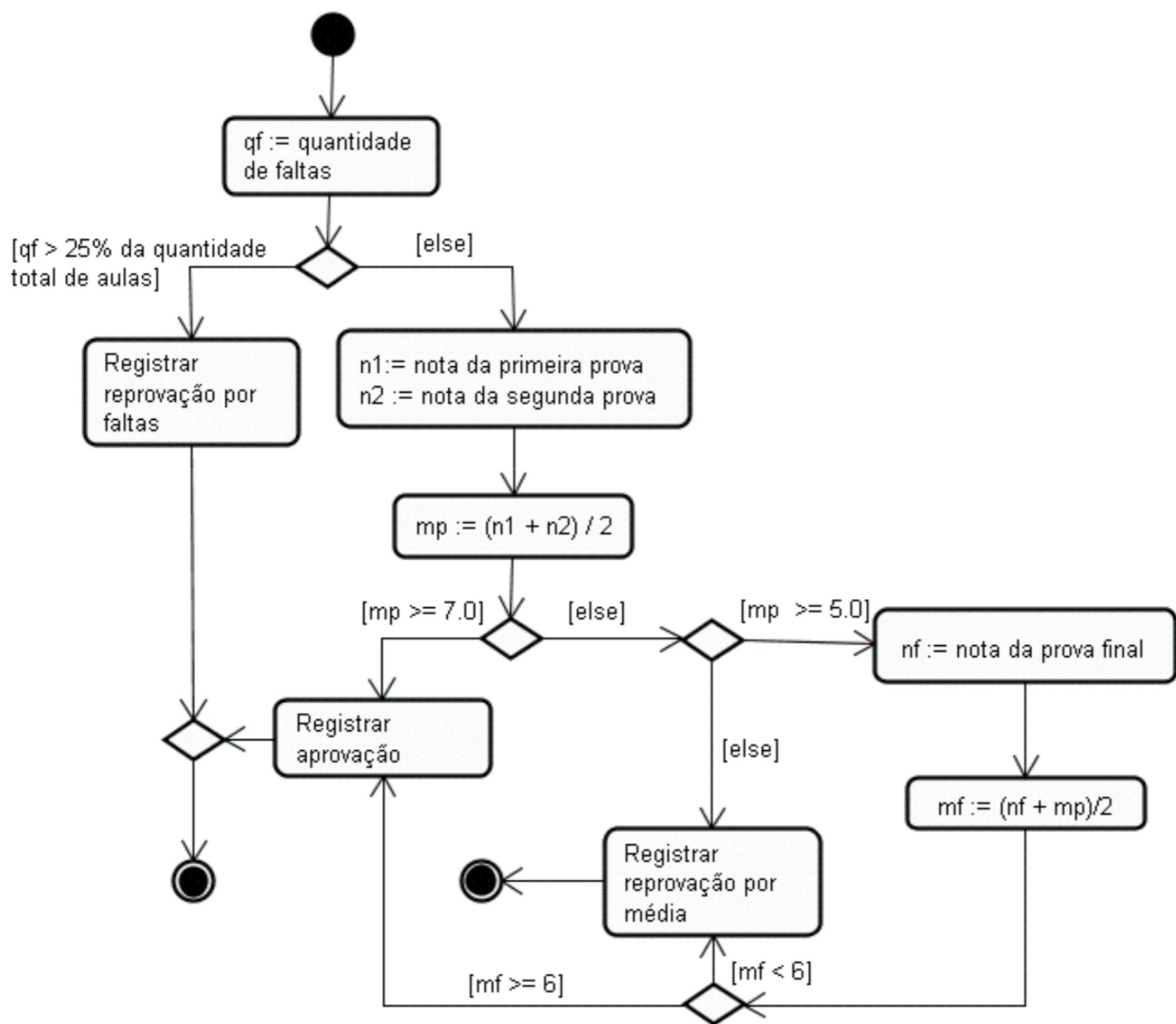


Diagrama de Atividades

- Modelagem da lógica de uma operação complexa
- Em alguns casos, quando uma operação de uma classe de controle implementa uma regra de negócio, pode haver a necessidade de descrever a lógica dessa operação ou da própria regra de negócio.
- Diagramas de atividades também podem ser usados com esse objetivo

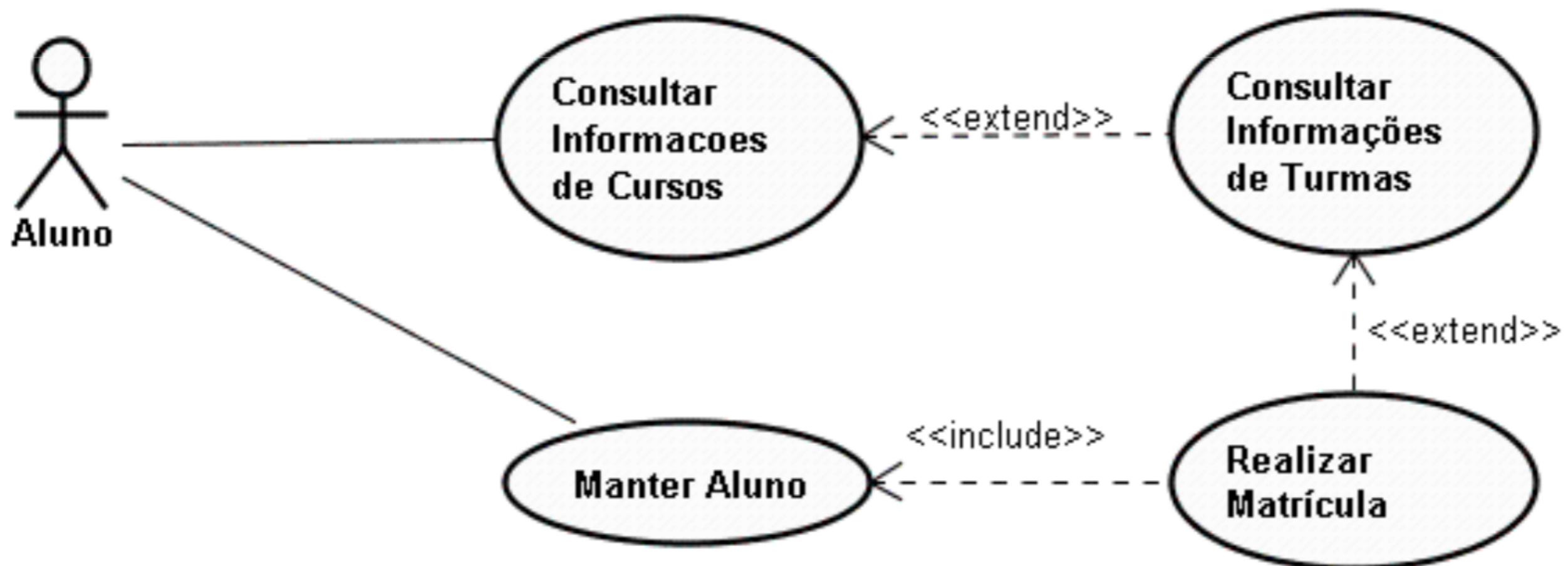
Diagrama de Atividades

- Descrição de uma regra de negócio
 - A nota de um aluno em uma disciplina (um valor de 0 a 10) é obtida pela média de duas avaliações durante o semestre, A1 e A2, ou pela freqüência nas aulas.
 - Se o aluno obtiver nota maior ou igual a 7.0 (sete), será aprovado.
 - Se o aluno obtiver nota maior ou igual a 5.0 (cinco) e menor que 7.0 (sete), deverá fazer a avaliação final.
 - Se o aluno obtiver nota menor que 5.0 (cinco) será reprovado. Se o aluno obtiver uma freqüência menor que 75% em uma turma, será automaticamente reprovado.
 - Após a prova final, o aluno será considerado aprovado, se sua média final for maior ou igual a 6.0 (seis), caso contrário, será reprovado.



Exercícios

1. Analise o Diagrama de Casos de Uso abaixo, referente a um módulo de matrícula e construa um Diagrama de Atividades para demonstrar modelagem dos processos do negócio.



Exercícios

2. Leia, interprete a descrição do caso de uso abaixo e complemente a sua especificação através de um Diagrama de Atividades

Projeto: Controle de Cursos

Nome: Manter Aluno

Descrição: Este caso de uso permite a inclusão, exclusão, alteração e consulta de alunos, pela atendente

Ator Principal: Aluno

Ator Secundário: Atendente

Pré-condição: A atendente deverá estar devidamente identificada pelo sistema

Fluxo Principal:

1. A Atendente informa o código do aluno [A1]
2. A Atendente solicita a busca
3. O sistema pesquisa os dados do aluno
4. O sistema exibe os dados do aluno [A2]
5. A Atendente edita os dados do aluno [A3]
6. A Atendente solicita a gravação dos dados
7. O sistema valida os dados informados
8. O sistema grava os dados do aluno [A4]
9. Fim do caso de uso

Fluxos Alternativos: A1. Novo Aluno

1. A Atendente solicita a inclusão de um novo aluno
2. O sistema solicita os dados do novo aluno
3. A Atendente informa os dados do aluno
4. Vai para o passo 6 do fluxo principal

Exercícios

A2. Aluno não encontrado

1. O sistema informa a situação à atendente
2. Vai para o passo 1 do Fluxo Principal

A3. Exclusão de Aluno

1. Atendente solicita exclusão do aluno
2. O sistema solicita confirmação da exclusão
3. [se confirmação positiva] Sistema exclui aluno
4. Vai para o passo 9 do fluxo principal

A4. Dados inválidos

1. Se algum dado do aluno estiver em desacordo com as regras de validações e restrições, o sistema informa situação à Atendente
2. Vai para o passo 5 do fluxo principal

Pós-condições: Os dados são incluídos, alterados ou excluídos conforme solicitação do aluno

Restrições e Validações:

1. Nenhum campo poderá ser deixado em branco
2. O campo CPF deverá ser preenchido somente com números
3. O ano de nascimento deverá ser informado com 4 dígitos

Exercícios

3. Construa um diagrama de Atividades para o seguinte processo de negócio

A autorização do pagamento tem início após um pedido ter sido realizado pelo cliente.

Ao mesmo tempo, a disponibilidade para cada um dos itens do pedido é verificada pelo depósito.

Se a quantidade requisitada de um determinado item existe em estoque, tal quantidade é associada ao pedido, caso contrário, a quantidade do item será alterada (se houver em quantidade menor), se a quantidade em estoque for igual a zero, o item será excluído.

O pedido é enviado pelo depósito ao cliente quando todos os itens estiverem associados e o pagamento estiver autorizado.

O pedido será cancelado se a ordem de pagamento não tiver sido autorizada.

Exercícios

4. Construa um diagrama de Atividades para o seguinte processo de negócio

O sócio deve se dirigir ao atendente e apresentar seu código, ou, caso não lembre, seu nome

O atendente pesquisará então o sócio para verificar se este realmente se encontra registrado, se a pessoa em questão não estiver registrada, a locação deve ser recusada

Caso o sócio esteja cadastrado, o sistema deve verificar se este possui alguma pendência, ou seja, se possui alguma locação ainda não devolvida. Se houver alguma pendência a locação deverá ser recusada

Se o sócio não possuir pendências, então o atendente irá registrar a locação, bem como cada uma das cópias locadas

Diagrama de Classes

- Apresenta como as classes interagem entre si e qual a responsabilidade de cada classe na realização das operações solicitadas pelos atores.

Diagrama de Classes

- Definições:
- Classe -> é um grupo de objetos, sendo que cada objeto é um exemplo de um determinado grupo.
- Diagrama de Classes -> é uma representação da estrutura e relações das classes que servem de modelo para os objetos.

Diagrama de Classes

- As classes do diagrama possuem:
 - Nome: sempre deve ser iniciado com letra maiúscula
 - Atributos: Visibilidade ou nível de encapsulamento; Nome (deve ser iniciado com letra minúscula); Tipo de dados

Diagrama de Classes

- As classes do diagrama possuem:
 - Operações ou métodos:
 - Visibilidade ou nível de encapsulamento
 - Nome (deve ser iniciado por letra minúscula)
 - Lista de parâmetros (se houver)
 - Tipo de retorno
 - Associações entre si
 - Nome (opcional)
 - Multiplicidades
 - Navegabilidade (opcional)

Diagrama de Classes

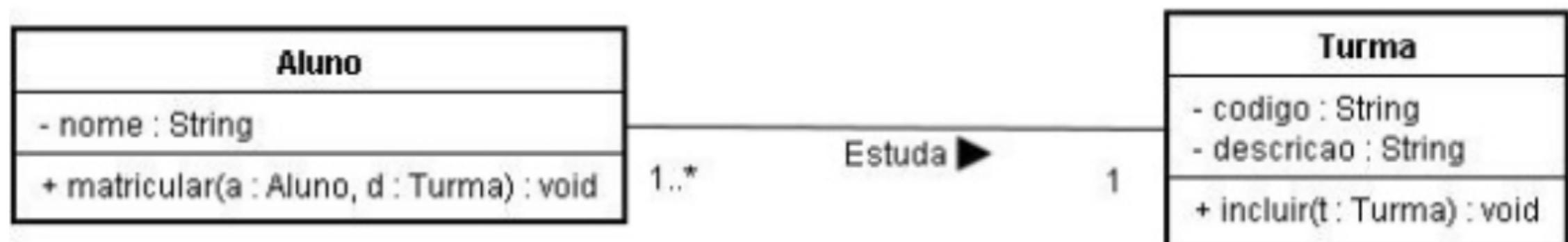


Diagrama de Classes

- Atributos
 - Visibilidade ou nível de encapsulamento
 - - private
 - # protected
 - + public
 - default
 - Nome: demonstram as características dos objetos
 - Tipo de dados: São os mesmos tipos usados em Java: String, boolean, int, float, double, Date, etc...

Diagrama de Classes

- Operações ou Métodos
 - Visibilidade ou nível de encapsulamento: os mesmos usados para os atributos
 - Nome: o nome do método deve expressar a ação que ele realiza, por exemplo incluirAluno(). Não deve possuir espaços nem começar com dígitos.
 - Lista de Parâmetros: Deverá vir entre parênteses e separados por vírgula.
 - Tipo de Retorno:
 - Informa que tipo de dado o método deverá retornar após sua execução
 - Se o método não retornar nada, deverá ser usada a palavra void no tipo de retorno

Diagrama de Classes

- Associação entre Classes
- Para representar o fato de que objetos podem se relacionar uns com os outros, utiliza-se a associação
- Representa que duas classes possuem uma ligação (link), significando por exemplo que elas “conhecem uma a outra”

Diagrama de Classes

- Associação
- Representada através de um segmento de reta ligando as classes cujos objetos se relacionam
- Nome da associação: Quando usado, deverá ser escrito junto à linha que representa a associação. Normalmente um verbo (não é obrigatório)
- Multiplicidades: Cada associação em um diagrama de classes possui duas multiplicidades, uma em cada extremo da linha de associação
- Naveabilidade ou direção de leitura: indica como a associação deve ser lida

Diagrama de Classes

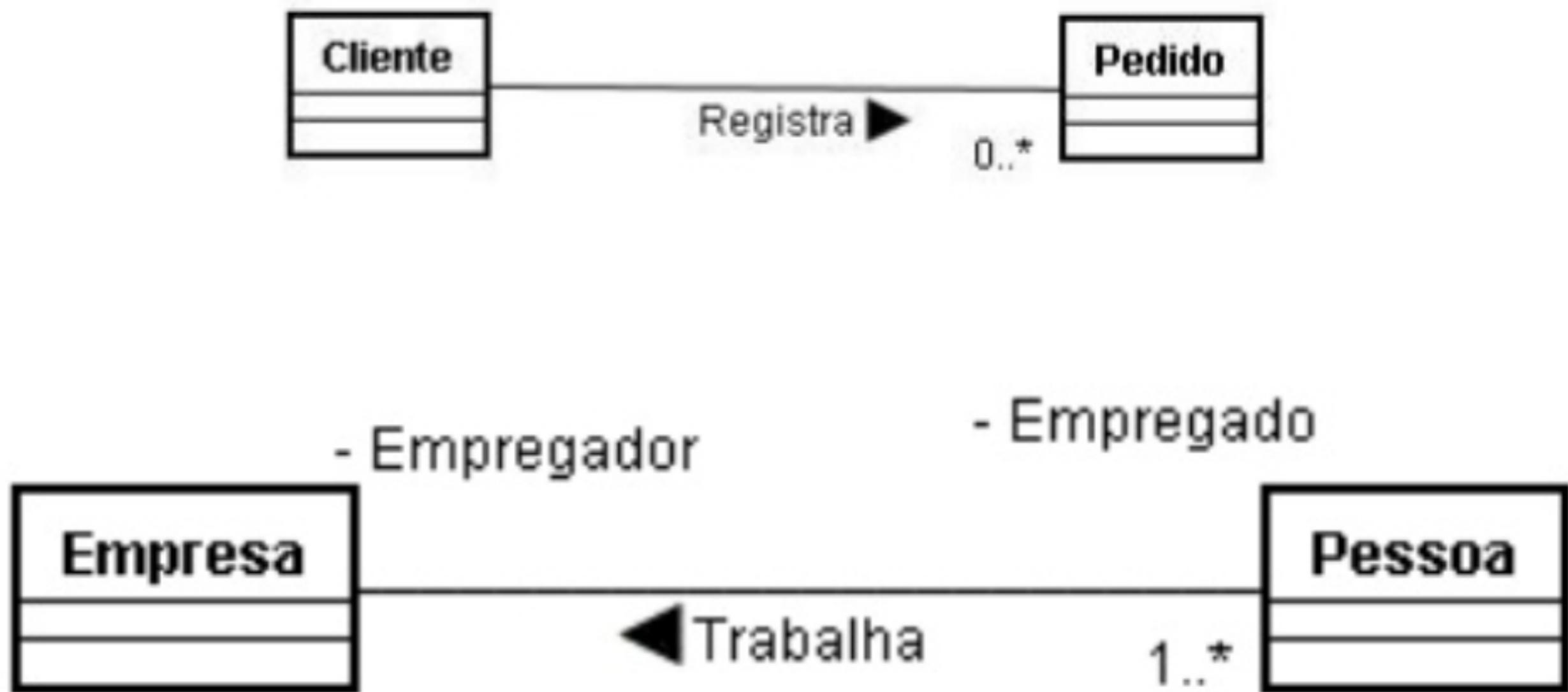


Diagrama de Classes

Opções de Multiplicidade

Nome	Simbologia
Apenas Um	1..1 (ou 1) (ou em branco)
Zero ou Muitos	0..* (ou *)
Um ou Muitos	1..*
Zero ou Um	0..1
Intervalo Específico	$I_{\text{início}} \dots I_{\text{fim}}$

Diagrama de Classes

- Classe Associativa
- É uma classe que está ligada a uma associação, ao invés de estar ligada a outras classes.
- É normalmente necessária quando duas ou mais classes estão associadas, e é necessário manter informações sobre esta associação (histórico)

Diagrama de Classes

- Classe Associativa

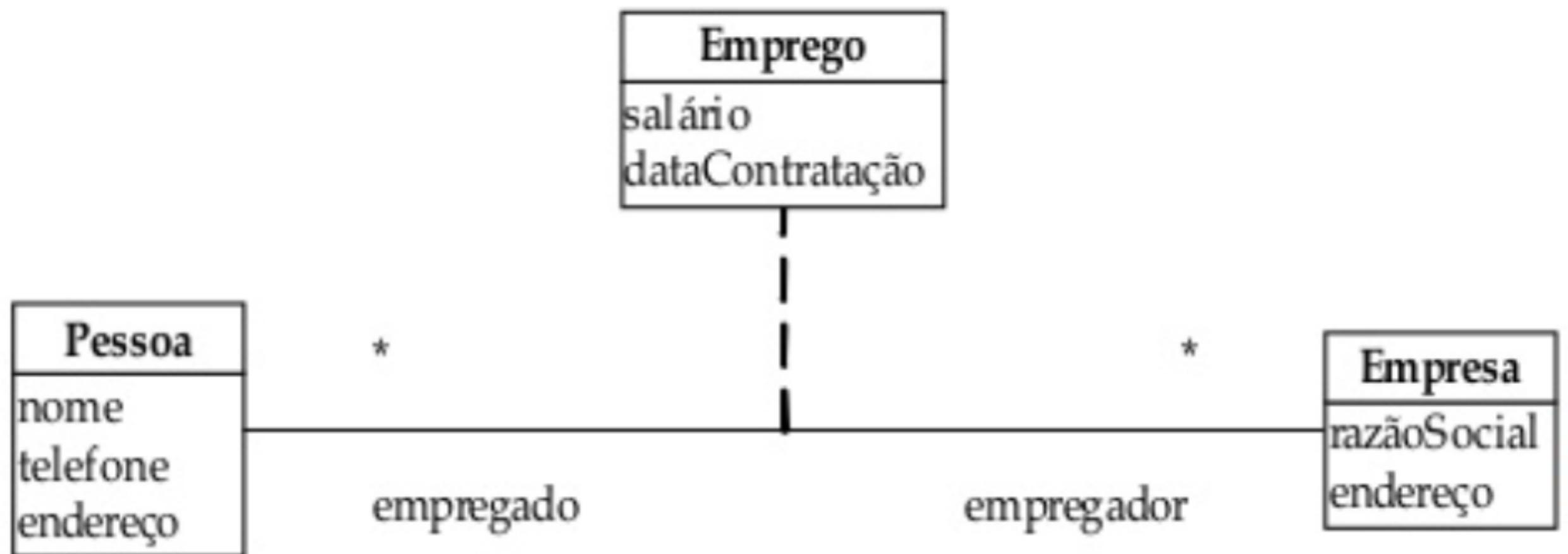


Diagrama de Classes

- Agregação
- É um caso especial de associação e, consequentemente, multiplicidades, nome da associação e papéis, podem ser usados normalmente.
- Utilizada para representar conexões que guardam uma relação todo-parte entre si
- Em uma agregação, um objeto está contido no outro, ao contrário da associação
- Onde se puder usar uma agregação, uma associação também poderá ser utilizada.

Diagrama de Classes

- Agregação
- Características Particulares: agregações são assimétricas - se um objeto A é parte de um objeto B, B não pode ser parte de A.

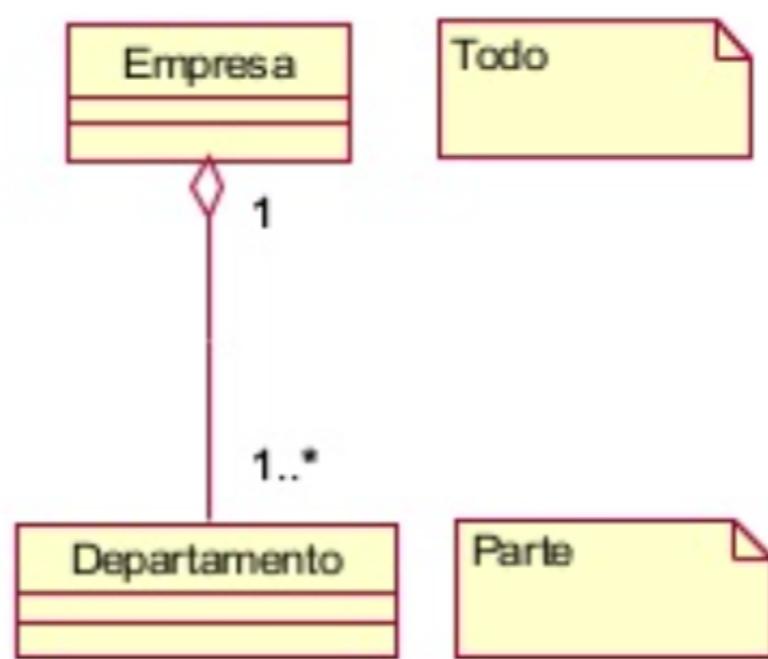


Diagrama de Classes

- Notação para uma agregação
- Representada através de uma linha conectando as classes relacionadas, com um losango perto da classe que representa o todo.

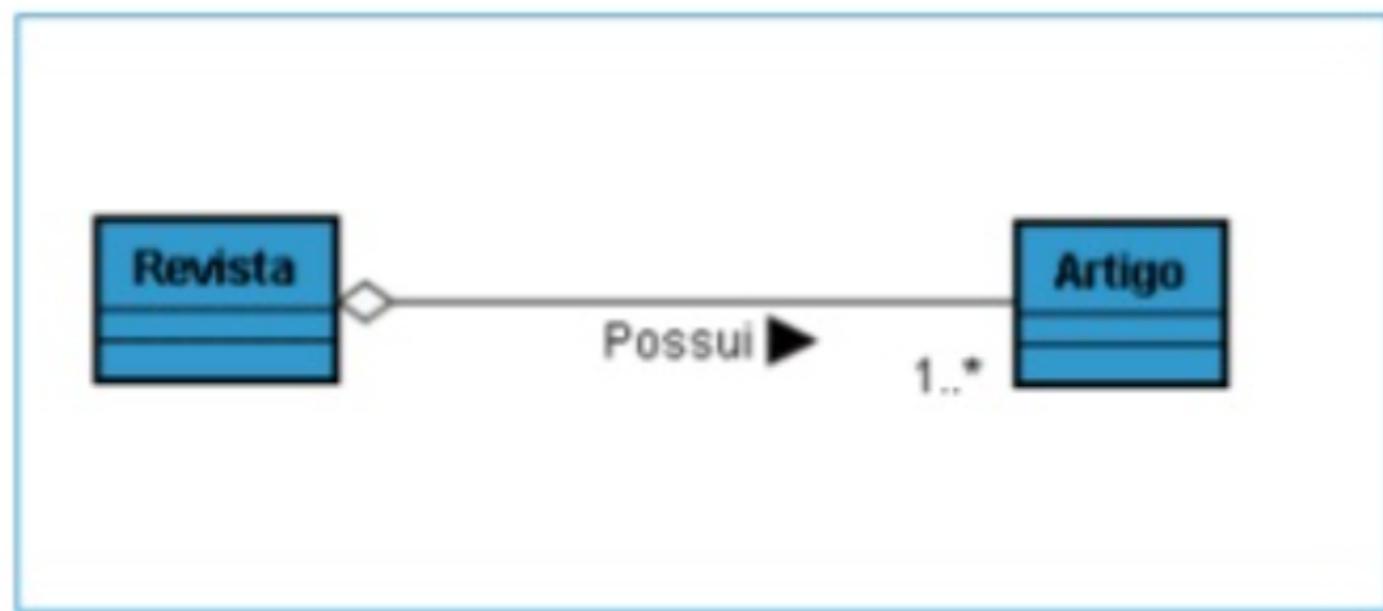


Diagrama de Classes

- Especialização/Generalização
- É um tipo de relacionamento similar à associação de mesmo nome em um Diagrama de Casos de Uso.
- Seu objetivo é identificar classes-mãe, chamadas gerais e classes-filhas, chamadas especializadas.
- Significa ser capaz de incorporar os atributos e métodos de uma outra classes previamente definida

Diagrama de Classes

- Especialização/Generalização

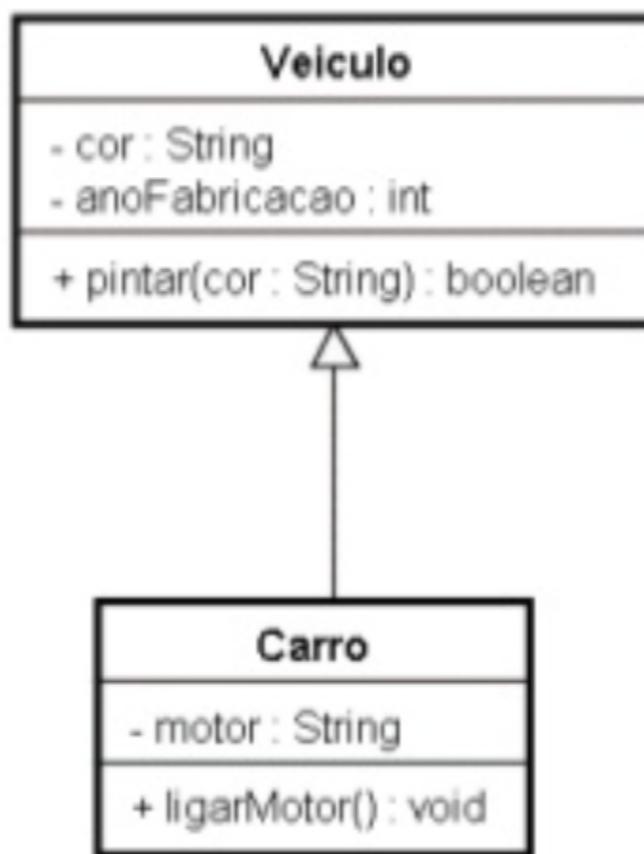


Diagrama de Classes

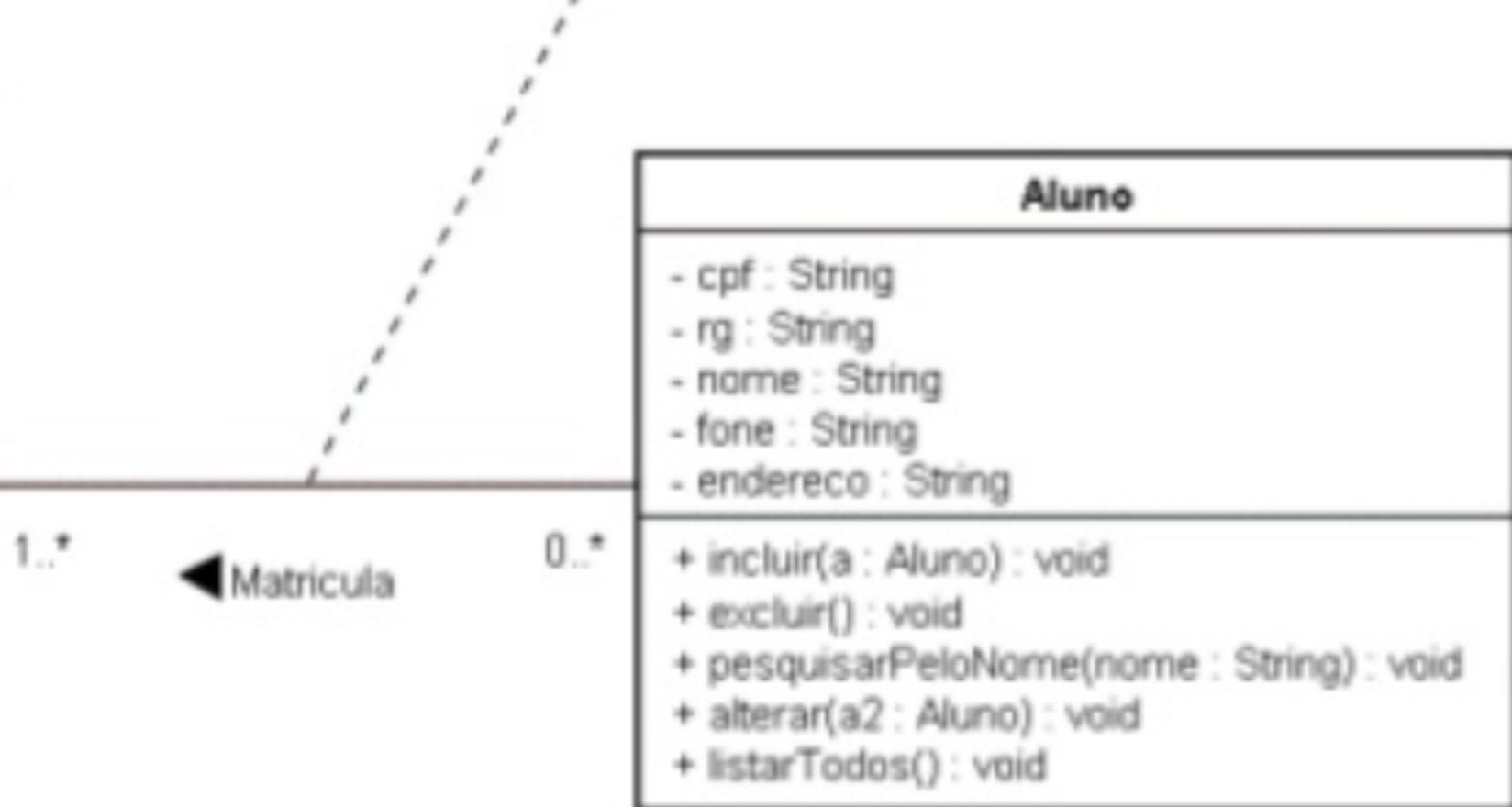
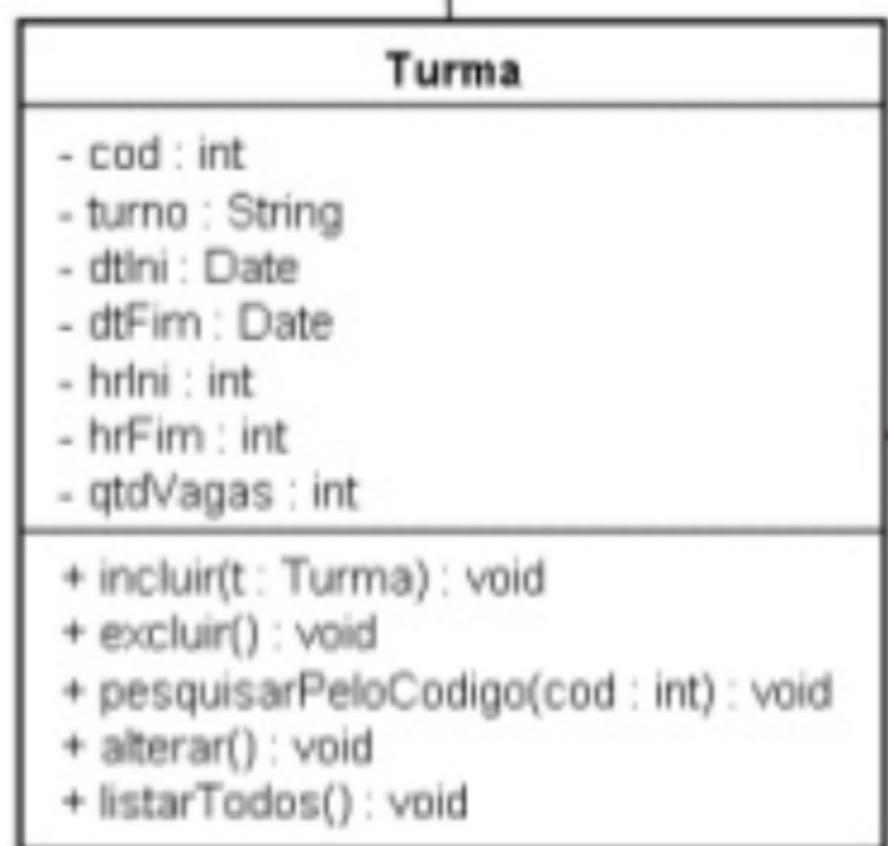
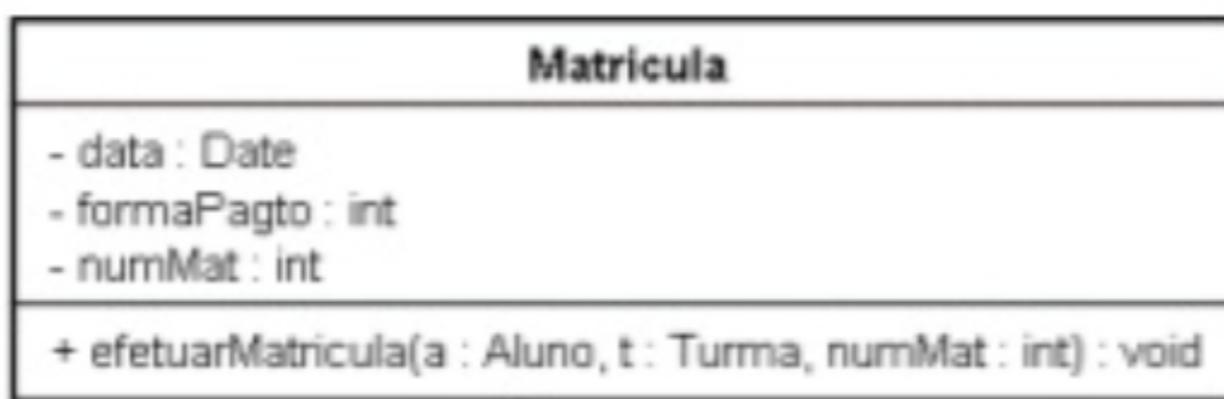
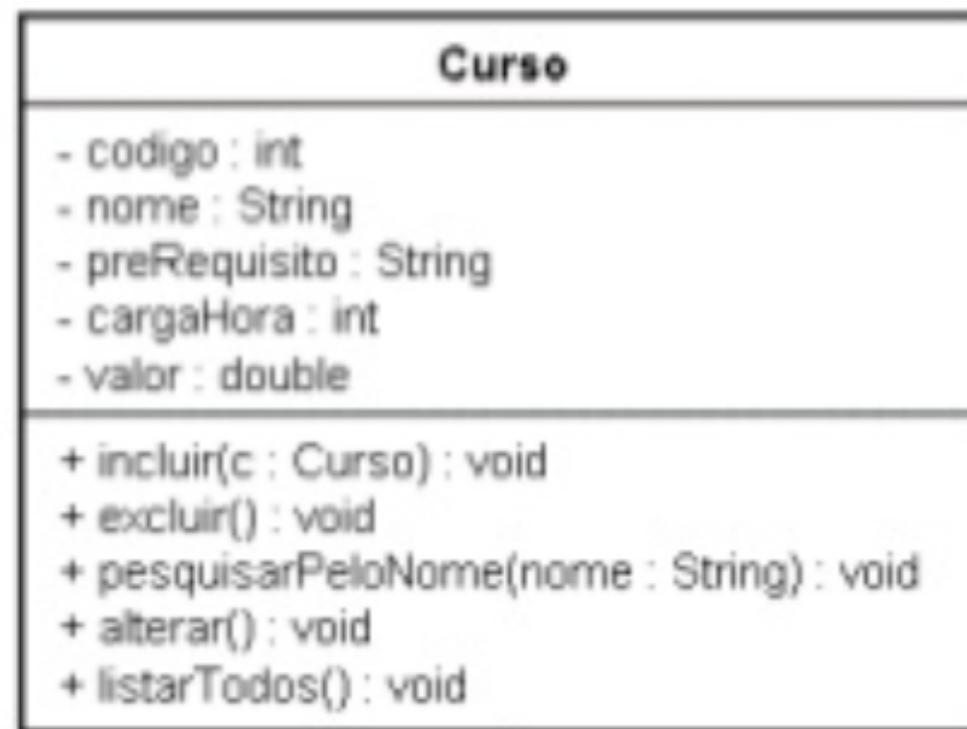
- Diagrama de classes vai demonstrar a estrutura estática das classes de um sistema
- Na fase de análise, tendo em mãos o diagrama de casos de uso, podemos definir o diagrama de classes do sistema.
- O modelo de classes evolui durante o desenvolvimento do sistema.
- À medida que o sistema é desenvolvido, o modelo de classes pode ser incrementado com novos detalhes.

Diagrama de Classes

EXEMPLO - CONTROLE DE CURSOS

Desenvolva o Diagrama de Classes para um sistema de cursos de informática equivalente ao módulo de matrícula de acordo com os seguintes fatos:

- Um curso pode ter muitas turmas, no entanto, uma turma se relaciona exclusivamente com um único curso.
- Uma turma pode ter diversos alunos matriculados, no entanto uma matrícula refere-se exclusivamente a uma determinada turma. Cada turma tem um número mínimo de matrículas para iniciar o curso.
- Um aluno pode realizar muitas matrículas, mas cada matrícula refere-se exclusivamente a uma turma específica e a um único aluno.



EXERCÍCIOS

1. Lucélia controla em Excel uma planilha com a sua lista de compras mensal. Ela cadastra o nome do produto, a unidade de compra, a quantidade prevista para um mês, a quantidade que efetivamente será comprada, o preço estimado (atualizado todo mês) e o valor total de cada compra.

Produto	Unidade	Qtd Mês	Qtd Compra	Preço estimado
Arroz	5Kg	2	1	10,00
Feijão	Kg	6	6	3,50
Açúcar	5Kg	3	2	3,60
Carne	Kg	6	7,5	11,00

Identifique as classes, atributos, métodos e relacionamentos desse cenário



NOTAS

- A quantidade de compra pode variar em virtude de sobra de um mês para o outro, ou da necessidade de um gasto maior no mês. Por exemplo: um almoço em família.
- As compras são feitas pela própria Lucélia. Por esse motivo, ela não vê necessidade de relacionar as marcas dos produtos.
- Mensalmente, Lucélia analisa o quanto pagou por cada produto, e se achar necessário, atualiza o preço estimado de cada produto

EXERCÍCIOS

- I. Lucélia não tem mais tempo de fazer as compras pessoalmente. Precisou detalhar o produto, de forma a lhe permitir delegar essa tarefa a outra pessoa. Além disso, não quer que paguem um valor absurdo por algum produto. Sendo assim, incluiu em sua planilha as colunas “preço máximo já comprado” e “preço máximo a pagar” no mês corrente, onde esta última coluna é calculada a partir da coluna anterior acrescida de 5%. O “preço máximo já comprado” é inserido na planilha, a partir das compras efetivamente já realizadas

Quais são os atributos e/ou métodos que precisam ser incluídos nas classes do exercício anterior para refletir o novo cenário?

EXERCÍCIOS

1. Lucélia está assustada com a variação do preço de um supermercado para outro. Tem feito compras (ou pedido para fazer) em até três supermercados diferentes. Sendo assim, resolveu melhorar sua planilha. Criou uma segunda planilha que contém o preço mais baixo que ela encontrou num determinado mês, indicando a que supermercado pertence.

Produto	Mês de compra	Valor	Supermercado
Arroz	Julho	7,00	XXX
Arroz	Agosto	6,80	YYY
Arroz	Setembro	10,00	XXX
Feijão	Julho	2,10	XXX
Feijão	Agosto	3,50	YYY



EXERCÍCIOS

1. (Continuação)

Quais são os atributos e/ou métodos que precisam ser incluídos nas classes do exercício anterior para refletir o novo cenário? Verifique se há necessidade de criar novas classes.



EXERCÍCIOS

1. (Continuação)

Quais são os atributos e/ou métodos que precisam ser incluídos nas classes do exercício anterior para refletir o novo cenário? Verifique se há necessidade de criar novas classes.



EXERCÍCIOS

1. Lucélia quer saber qual o supermercado apresentou mais produtos baratos, num determinado mês.
 - a) A resolução dessa situação se dá por meio da inclusão de um método. Sugira um nome para esse método, indicando sua assinatura.
 - c) A que classe pertence esse método?
 - e) Comente com seus colegas como seria o algoritmo para a implementação desse método



EXERCÍCIOS

1. Lucélia deseja desconsiderar o preço de um determinado mês para cálculos de maior ou menor valor, ou ainda do supermercado mais vantajoso.

Exemplos:

- a) Saber qual o Supermercado apresentou mais produtos baratos, num determinado mês. Supondo que ela comprou um produto numa promoção que ofereceu 50% de desconto, esse valor não pode ser parâmetro para suas compras futuras.
- c) Para calcular o `precoMaximoComprado`, a aplicação pega o maior valor. Vamos supor que um determinado produto teve queda de preço. Isso significa que os meses de preço alto não podem ser considerados para a próxima compra

Em que classe deve ser incluído um atributo para resolver essa questão e qual o tipo desse atributo?



Modelo Entidade Relacionamento

- Também conhecido como modelo ER ou MER
- Modelo conceitual utilizado na Engenharia de Software para descrever objetos (entidades) envolvidos em um domínio de negócios com suas características (atributos) e como elas se relacionam entre si (relacionamentos)
- Em geral, este modelo representa de forma abstrata a estrutura que possuirá o banco de dados da aplicação.

Entidades

Os objetos ou partes envolvidas um domínio, também chamados de entidades, podem ser classificados como físicos ou lógicos, de acordo sua existência no mundo real.

Entidades físicas: são aquelas realmente tangíveis, existentes e visíveis no mundo real, como um cliente (uma pessoa, uma empresa) ou um produto (um carro, um computador, uma roupa). Já as entidades lógicas são aquelas que existem geralmente em decorrência da interação entre ou com entidades físicas, que fazem sentido dentro de um certo domínio de negócios, mas que no mundo externo/real não são objetos físicos (que ocupam lugar no espaço). São exemplos disso uma venda ou uma classificação de um objeto (modelo, espécie, função de um usuário do sistema).

Entidades

As entidades são nomeadas com substantivos concretos ou abstratos que representem de forma clara sua função dentro do domínio. Exemplos práticos de entidades comuns em vários sistemas são Cliente, Produto, Venda, Turma, Função, entre outros.

Entidades

Entidades fortes: são aquelas cuja existência independe de outras entidades, ou seja, por si só elas já possuem total sentido de existir. Em um sistema de vendas, a entidade produto, por exemplo, independe de quaisquer outras para existir.

Entidades fracas: ao contrário das entidades fortes, as fracas são aquelas que dependem de outras entidades para existirem, pois individualmente elas não fazem sentido. Mantendo o mesmo exemplo, a entidade venda depende da entidade produto, pois uma venda sem itens não tem sentido.

Entidades associativas: esse tipo de entidade surge quando há um relacionamento do tipo muitos para muitos (explicado a seguir). Nestes casos, é necessária a criação de uma entidade intermediária cuja identificação é formada pelas chaves primárias (explicado mais adiante) das outras duas entidades. No contexto de uma aplicação de vendas, como precisamos relacionar vendas e produtos numa relação muitos para muitos, a entidade produto não pode referenciar diretamente a venda, nem o inverso, pois isso caracterizaria um relacionamento um para um, ou um para muitos. Sendo assim, criamos uma entidade intermediária para representar os itens da venda, que tanto possuem a identificação do produto, quanto da venda em que estão contidos. Neste caso específico, também caberiam a esta entidade informações como quantidade de itens e desconto unitário, por exemplo.

Relacionamentos

Uma vez que as entidades são identificadas, deve-se então definir como se dá o relacionamento entre elas. De acordo com a quantidade de objetos envolvidos em cada lado do relacionamento, podemos classificá-los de três formas:

Relacionamento 1..1 (um para um): cada uma das duas entidades envolvidas referenciam obrigatoriamente apenas uma unidade da outra. Por exemplo, em um banco de dados de currículos, cada usuário cadastrado pode possuir apenas um currículo na base, ao mesmo tempo em que cada currículo só pertence a um único usuário cadastrado.

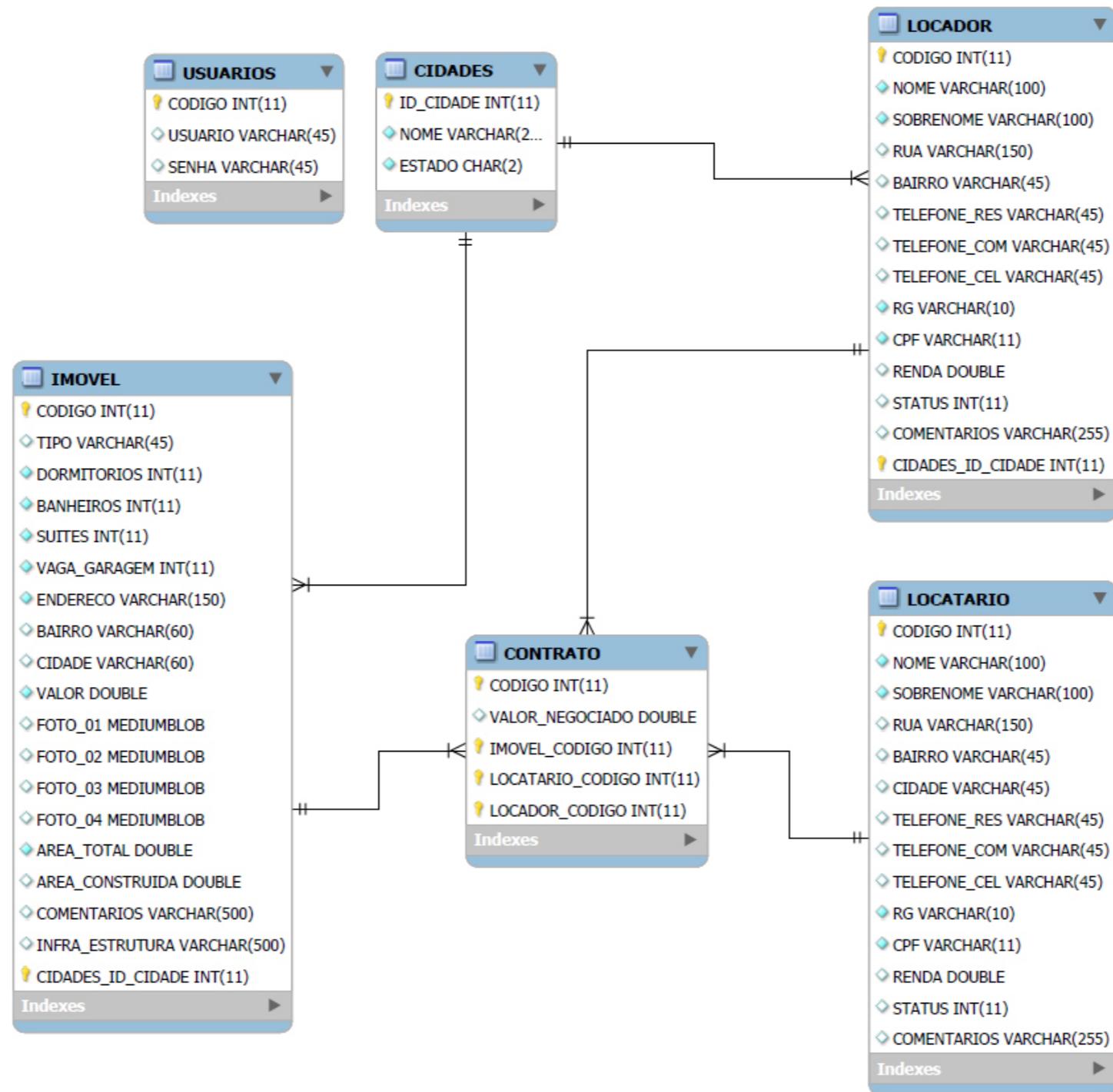
Relacionamento 1..n ou 1..* (um para muitos): uma das entidades envolvidas pode referenciar várias unidades da outra, porém, do outro lado cada uma das várias unidades referenciadas só pode estar ligada uma unidade da outra entidade. Por exemplo, em um sistema de plano de saúde, um usuário pode ter vários dependentes, mas cada dependente só pode estar ligado a um usuário principal. Note que temos apenas duas entidades envolvidas: usuário e dependente. O que muda é a quantidade de unidades/exemplares envolvidas de cada lado.

Relacionamento n..n ou *..* (muitos para muitos): neste tipo de relacionamento cada entidade, de ambos os lados, podem referenciar múltiplas unidades da outra. Por exemplo, em um sistema de biblioteca, um título pode ser escrito por vários autores, ao mesmo tempo em que um autor pode escrever vários títulos. Assim, um objeto do tipo autor pode referenciar múltiplos objetos do tipo título, e vice versa.

Atributos

Atributos são as características que descrevem cada entidade dentro do domínio. Por exemplo, um cliente possui nome, endereço e telefone. Durante a análise de requisitos, são identificados os atributos relevantes de cada entidade naquele contexto, de forma a manter o modelo o mais simples possível e consequentemente armazenar apenas as informações que serão úteis futuramente. Uma pessoa possui atributos pessoais como cor dos olhos, altura e peso, mas para um sistema que funcionará em um supermercado, por exemplo, estas informações dificilmente serão relevantes.

Diagrama MR



Exercício

- Cria o Modelo Relacional de um sistema simples de compra
- Nesse sistema, cada cliente possui um login
- Um cliente possui diversas compras
- As compras são formadas por produtos