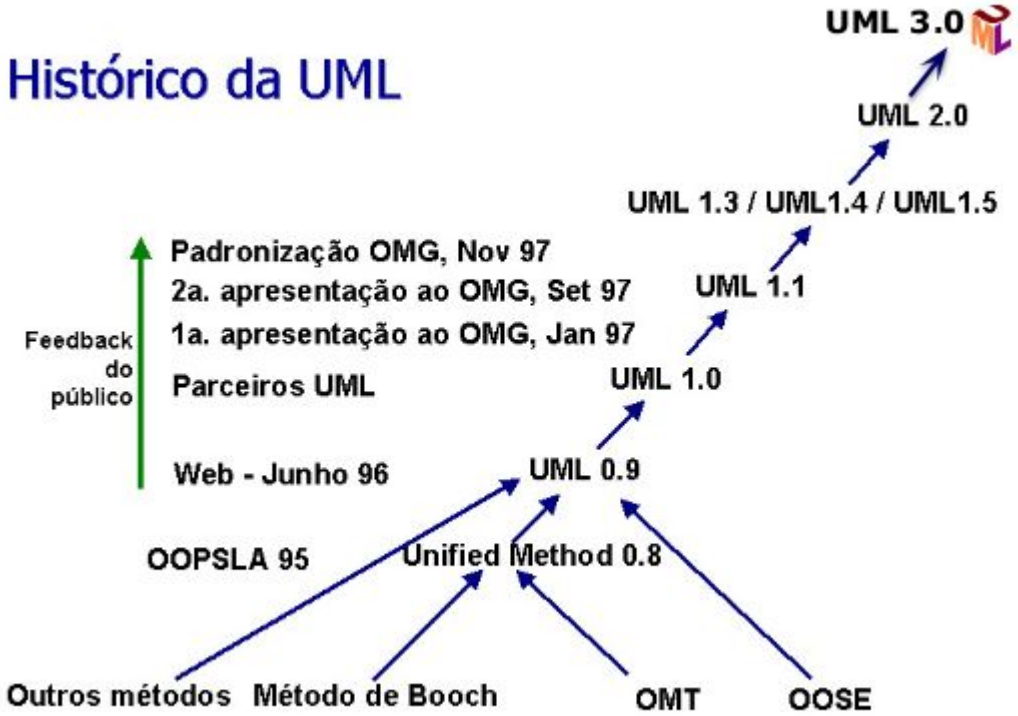
Modelagem de Software

Prof. Pedro Henrique Dias Valle

Adaptado de:
Prof. Fabrício Martins Mendonça

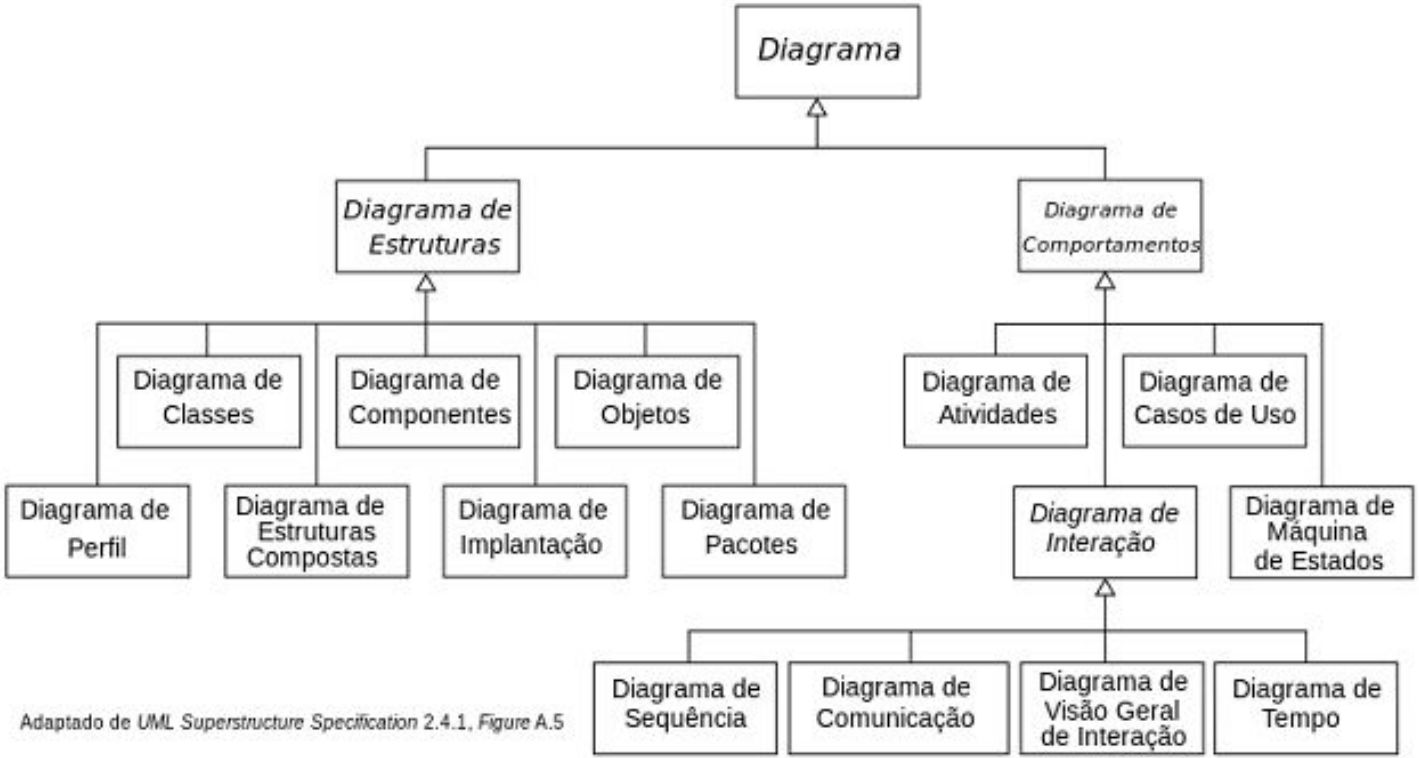
Introdução a UML



Introdução a UML

- A UML é aplicável em:
  - Modelagem de processos de negócio
  - Modelagem de casos de uso
  - Modelagem de classes e objetos
  - Modelagem de interação entre objetos
  - Modelagem de componentes
  - Modelagem de implantação de componentes

Diagramas da UML



Adaptado de *UML Superstructure Specification* 2.4.1, *Figure* A.5

A 1 km de distância...

- Caixas representando as classes
- Linhas representando os relacionamentos

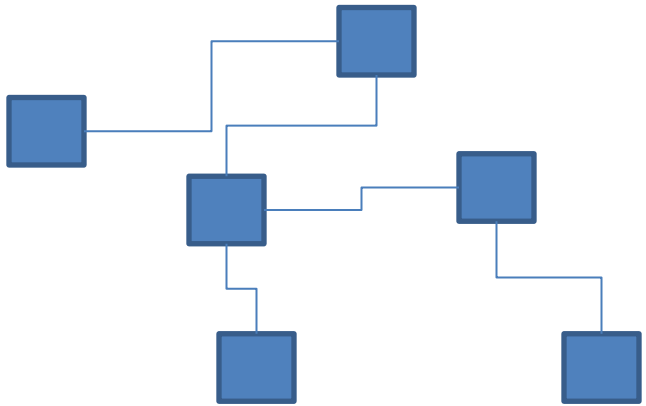
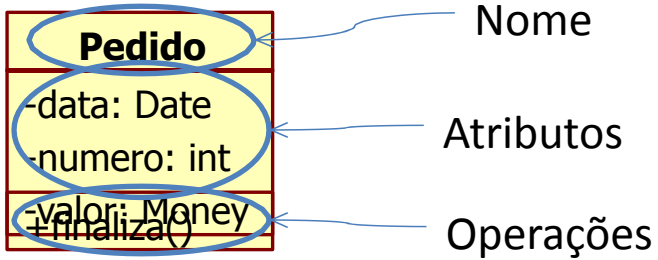


Diagrama de Classes - O que é?

- Diagrama mais utilizado da UML
- Representa os tipos (classes) de objetos de um sistema
  - Propriedades desses tipos
  - Funcionalidades providas por esses tipos
  - Relacionamentos entre esses tipos
- Pode ser mapeado diretamente para uma linguagem de programação orientada a objetos
  - Ajuda no processo transitório dos requisitos para o código
  - Pode representar visualmente o código do sistema

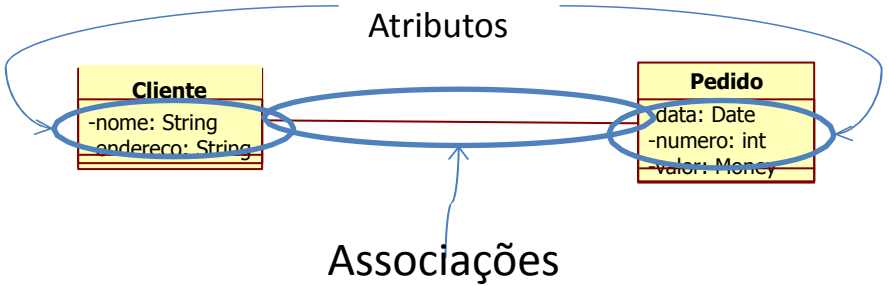
A 1 km de distância...

- As classes são representadas por caixas contendo
  - Nome (obrigatório)
  - Lista de atributos
  - Lista de operações



Propriedades

- Classes são descritas via suas propriedades
  - Primitivas: representadas como atributos
  - Compostas: representadas como associações para outras classes
- Quando transformadas para código, as propriedades se tornam sempre campos (atributos) da classe



A 1 centímetro de distância… dos atributos

- Atributos são descritos via
  - Visibilidade
  - Nome
  - Tipo
  - Multiplicidade
  - Valor padrão

- endereco : String[1] = “Sem Endereço”

Atributos (visibilidade)

- Privado (-)
  - Somente a própria classe pode manipular o atributo
  - Indicado na maioria dos casos
- Pacote (~)
  - Qualquer classe do mesmo pacote pode manipular o atributo
- Protegido (#)
  - Qualquer subclasse pode manipular o atributo
- Publico (+)
  - Qualquer classe do sistema pode manipular o atributo

Atributos (nome e tipo)

- O nome do atributo corresponde ao nome que será utilizado no código fonte
  - É aceitável utilizar nomes com espaço e acentos na fase de análise
- O tipo do atributo corresponde ao tipo que será utilizado no código fonte
  - Tipos primitivos da linguagem
  - Classes de apoio da linguagem (String, Date, Money, etc.)

- endereco : String

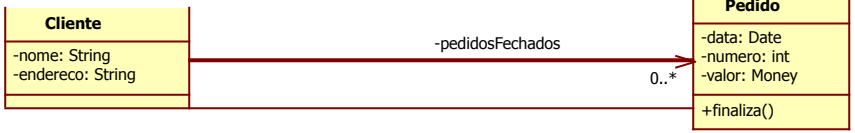
Atributos (multiplicidade)

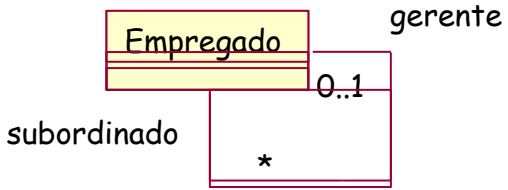
- Representa o número de elementos de uma propriedade
- Estrutura X..Y onde
  - Opcional: X = 0
  - Mandatório: X = 1
  - Somente um valor: Y = 1
  - Multivalorado: Y > 1
- Valores clássicos
  - 0..1
  - 1 (equivalente a 1..1 default)
  - * (equivalente a 0..*)
  - 1..*

- endereco : String[0..3]

A 1 centímetro de distância... das associações

- Associações
  - Guarda as mesmas informações dos atributos
  - Utiliza uma notação gráfica
  - Deve ser utilizado para propriedades que são relevantes ao diagrama
  - Determina o papel das classes na associação
  - Determina o sentido de navegação





A 1 centímetro de distância… das operações

- Operações são descritas via
  - Visibilidade
  - Nome
  - Lista de parâmetros
  - Tipo de retorno

+ finaliza(data : Date) : Money

Operações (visibilidade)

- Valem as mesmas regras de visibilidade de atributos
- Privado (-)
  - Funcionalidades de apoio à própria classe
- Pacote (~)
  - Funcionalidades de apoio a outras classes do pacote (ex. construção de um componente)
- Protegido (#)
  - Funcionalidades que precisam ser estendidas por outras classes (ex. construção de um *framework*)
- Publico (+)
  - Funcionalidades visíveis por todas as classes do sistema

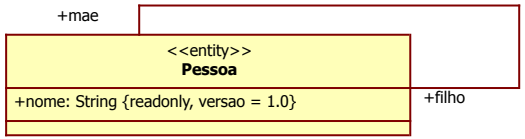
+ finaliza(data : Date) : Money

Em análise...

- Não se atenha aos detalhes
  - Visibilidade
  - Navegabilidade
  - Tipo
- Visibilidade pública em propriedades
  - Assume campo privado e métodos de acesso (get e set)
- Operações
  - Somente as responsabilidades obvias das classes

Palavras-chave, propriedades e restrições

- Apóiam a linguagem gráfica com informações textuais
- Permitem dar mais semântica aos elementos do modelo
- Notação de palavra-chave
  - Textual: <<palavra>> (ex.: <<interface>>)
  - Icônica: imagem representando a palavra-chave
- Notação de propriedades e restrições
  - {propriedade} (ex.: {readonly})
  - {nome = valor} (ex.: {versão = 1.0}



Propriedades de atributos e associações

- Alguns exemplos...
- {readonly}
  - Somente oferece operações de leitura
- {ordered}, {unordered}
  - Indica se o atributo ou associação multivalorado mantém a sequência dos itens inseridos

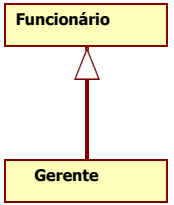
- endereco : String = “Sem Endereço” {readonly}

Outros relacionamentos entre classes

- Além das associações, alguns outros tipos de relacionamentos são importantes
  - Generalização
  - Composição
  - Agregação
  - Dependência
  - Classes de associação

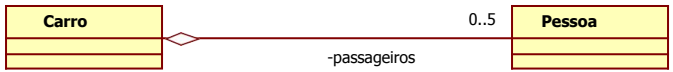
Generalização

- Visa estabelecer relações entre tipos
- Leitura: “é um”
- Se Gerente “é um” Funcionário
  - Todas as operações e propriedades (não privadas) de Funcionário vão estar disponíveis em Gerente
  - Toda instância de Gerente pode ser utilizada onde se espera instâncias de Funcionário
  - Gera o efeito de herança e polimorfismo quando mapeado para código



Agregação

- É uma associação com a semântica de “contém”
- Serve como uma relação todo-parte fraca
- O todo existe sem as partes
- As partes existem sem o todo
- A parte pode ser agregada por vários todos



Composição

- É uma associação com a semântica de “é composto de”
- Serve como uma relação todo-parte forte
- O todo não existe sem as partes
  - As partes pertencem a somente um todo
  - A remoção do todo implica na remoção das partes

Dependência

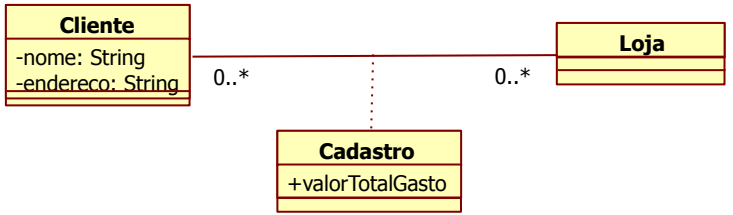
- Deixa explícito que mudanças em uma classe podem gerar consequências em outra classe
- Exemplos:
  - Uma classe chama métodos de outra
  - Uma classe tem operações que retornam outra classe
  - Uma classe tem operações que esperam como parâmetro outra classe
- Outros relacionamento (ex.: associação com navegação) implicitamente determinam dependência
- Não tente mostrar todas as dependências no seu diagrama!

Leitura: classe A depende da classe B

Classes de associação

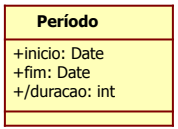
- Permitem a adição de informações em uma associação
- Devem ser transformadas em classes comuns posteriormente para viabilizar implementação

Qual o valor total gasto em cada loja?



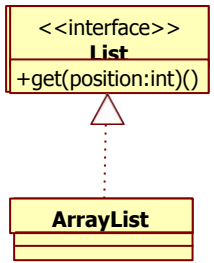
A 1 milímetro de distância... propriedades derivadas

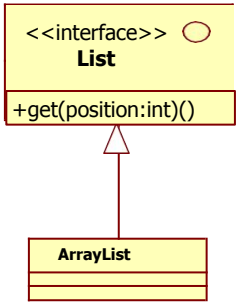
- São propriedades que na verdade não existem como atributos ou associações
- Podem ser inferidas por outras propriedades da classe
- É interessante explicitar através de nota ou restrição a fórmula de derivação
- São marcadas com o símbolo “/”



A 1 milímetro de distância… Interfaces

- Uma classe sem nenhuma implementação
  - Todas operações são abstratas
- Faz uso da palavra-chave <<interface>>
  - Pode ser representado também como um ícone
- O relacionamento de realização indica as classes que implementam a interface
  - Equivalente a generalização





Dicas

- Inicie com um diagrama simples
- O que normalmente tem em todo diagrama
  - Classes
  - Atributos
  - Operações
  - Associações
- Use os demais recursos da linguagem somente quando for realmente necessário

Dicas (possíveis candidatos)

- Classes
  - Entidades externas que produzem ou consomem informações (ex.: sistema de validação do cartão de crédito)
  - Coisas que são parte do problema e que são informações compostas (ex.: Produto)
  - Eventos que ocorrem durante a operação do sistema (ex.: Pedido)
  - Papeis que interagem com o sistema (ex.: Cliente)
  - Unidades organizacionais relevantes (ex.: Rede de lojas)
  - Lugares que fornecem o contexto do problema ou do sistema (ex.: Loja)
  - Estruturas definidas no problema (ex.: Estoque)

Dicas (possíveis candidatos)

- Atributos
  - Informação primitiva que precisa ser memorizada (ex.: Preço)
- Associações
  - A classe A precisa se relacionar com a classe B para atender a operações específicas (ex.: Cliente – Pedido)
- Operações
  - Funcionalidades que devem ser providas por uma classe para viabilizar o uso do sistema (ex.: calculaTotal em Pedido)

DIAGRAMA DE ATIVIDADE

Diagrama de Atividades

- Uma **atividade** é um estado de realização de algo:
  - Um processo do negócio;
  - Uma rotina de software;
  - Fluxo e sequência das funcionalidades do sistema

- Um **Diagrama de Atividades**:
  - Representa aspectos dinâmicos;
  - Descreve uma sequência de atividades;
  - Suporta comportamento condicional e paralelo;
  - Similar a um Fluxograma, mas permite fluxos concorrentes.

O que é?

- Diagrama criado para modelagem de processos

- Ferramenta útil para modelar processos
  - Processos de negócio
  - Casos de uso
  - Passos de execução de uma rotina
  - Workflows

Diagrama de atividades

- No diagrama de atividades, o objetivo é modelar um processo
  - Pode envolver diversos objetos
  - Pode implicar diferentes estados no objeto

A 1 km de distância...

- Retângulos de canto arredondado representando atividades ou ações
- Caixas representando objetos
- Linhas representando fluxos de controle

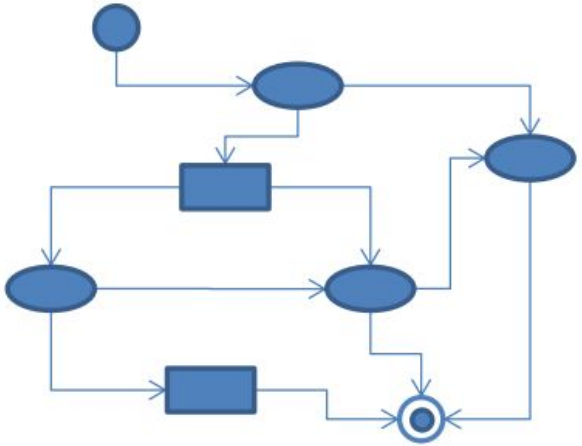


Diagrama de Atividades

Elementos do Diagrama:

- **Nó Inicial**
  - Podemos ter mais de um nó inicial;
- **Ação/Atividade**
  - Representa passo, rotina, transformação de dados por meio de um processamento
- **Fluxo de Controle**
  - São as setas que mostram a ordem de execução das ações
- **Nó de Decisão**
  - Equivalente a uma estrutura de decisão (IF then ELSE);
  - Podem ter várias decisões no nó;

Diagrama de Atividades

Nó de Decisão - comportamento condicional:

É delineado por desvios e intercalações;

- **Desvio**:
  - Uma transição de entrada;
  - Várias transições de saída guardadas;
  - Somente uma transição de saída pode ser tomada;
- **Intercalação**:
  - Múltiplas transições de entrada;
  - Uma transição de saída;
  - Marca o final de um desvio.

Diagrama de Atividades

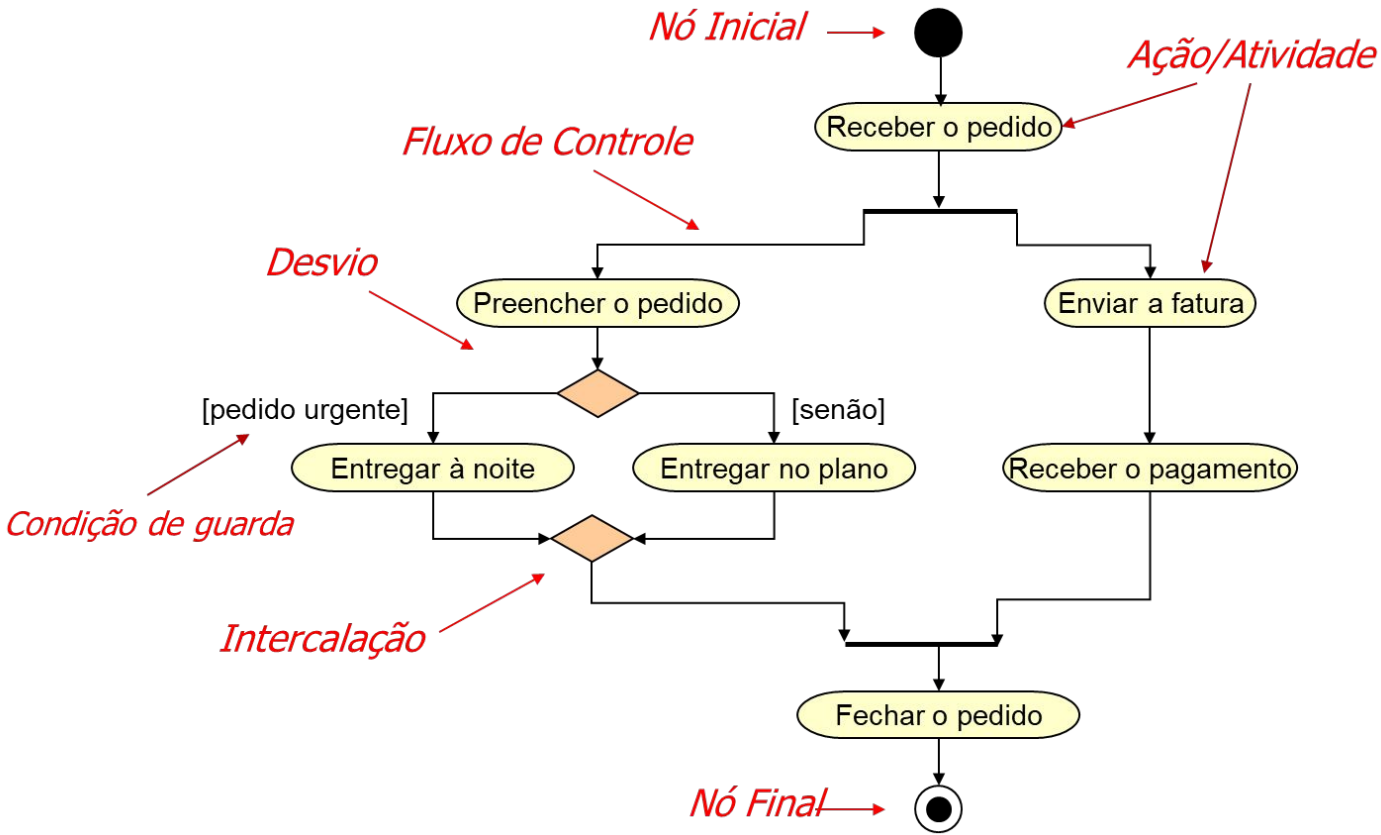


Diagrama de Atividades

Diagrama de Atividades de matrícula de um Sistema Acadêmico.

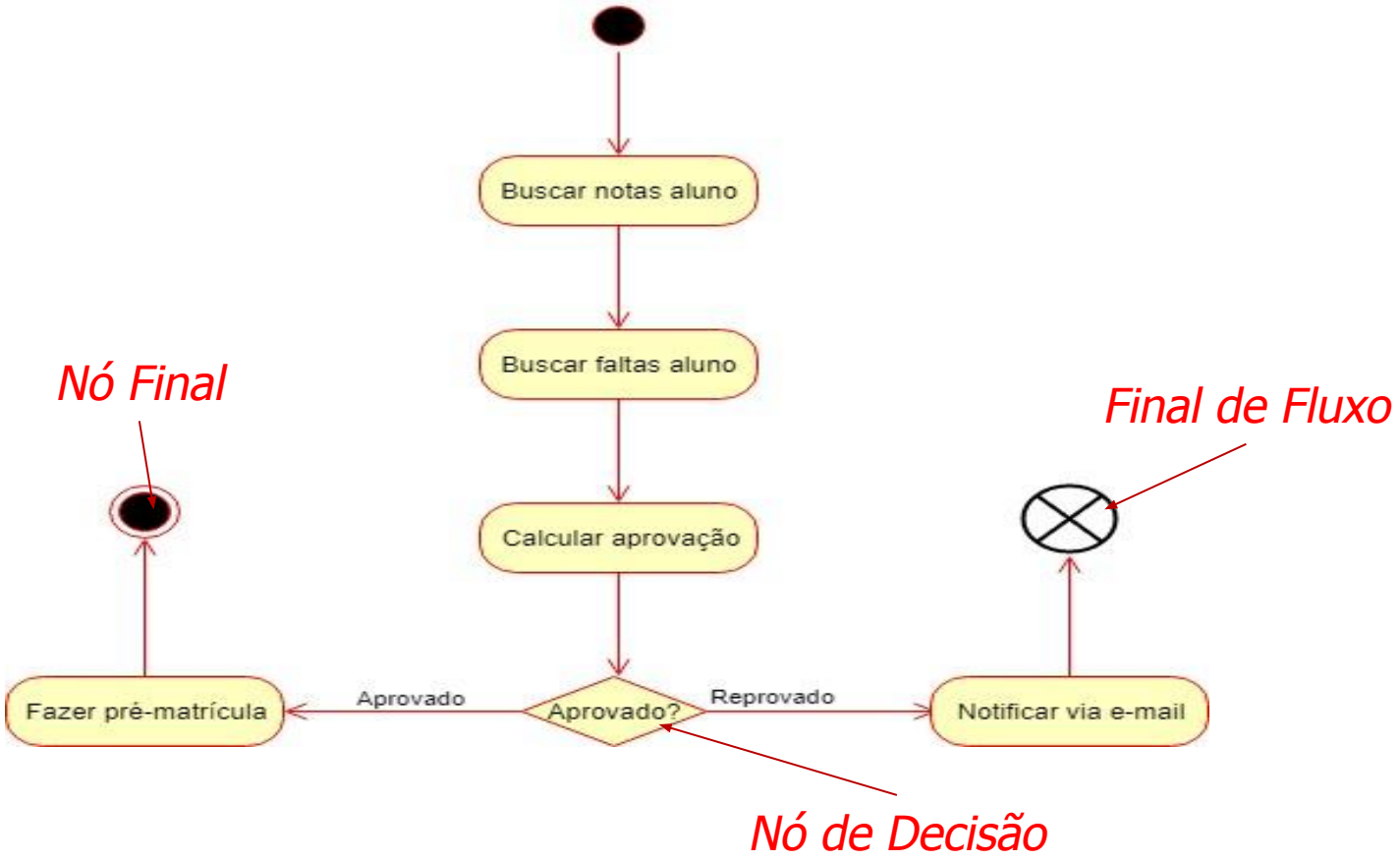


Diagrama de Atividades

Elementos do Diagrama:

- **Final de Fluxo**
  - Mostra o encerramento de uma rotina mas não todo fluxo;
- **Nó Final**
  - Ponto de saída de um processamento (todo fluxo). É permitido também mais de um nó final nos diagramas
- **Nó de Bifurcação (Fork)**
  - Quando um fluxo de controle se transforma em 2 ou mais;
- **Nó de União (Join)**
  - Quando 2 ou mais fluxos se transformam em um único fluxo de controle.

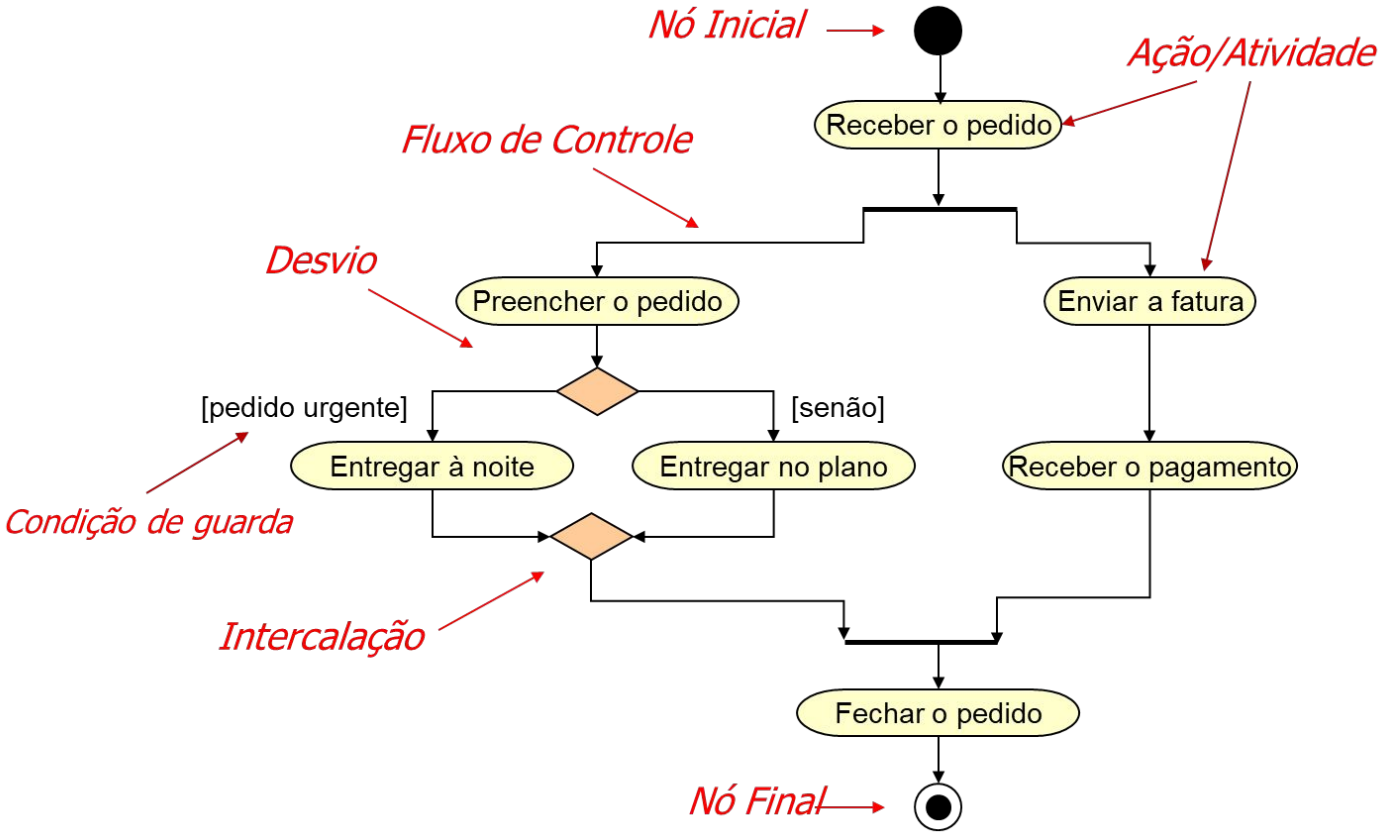
Diagrama de Atividades

Comportamento paralelo:

É indicado por separações e junções;

- **Separação/Bifurcação (Fork):**
  - Uma transição de entrada;
  - Várias transições de saída;
  - Uma transição de entrada dispara todas as transições de saída;
- **Junção/União (Join)**:
  - Múltiplas transições de entrada;
  - Sincroniza as atividades que acontecem em paralelo;
  - Separação e junção devem se completar.

Diagrama de Atividades



A 1 km de distância...

- As ações representam procedimentos atômicos
  - Não podem ser decompostos
- As atividades representam procedimentos compostos
  - Têm um sub-diagrama de atividades próprio
- Uma ação pode ser posteriormente detalhada, se tornando uma atividade

A 1 metro de distância... dos fluxos de controle

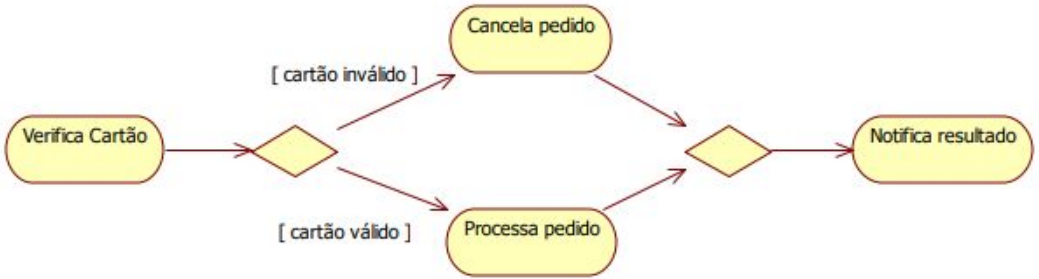
- Fluxos de controle determinam precedência entre atividades

- Uma atividade somente pode executar após a execução de todas as atividades com fluxos que desemboquem nela

Decisões

- O diagrama de atividades permite que decisões sejam tomadas durante o fluxo

- Os fluxos que saem das decisões têm condições de guarda, determinando qual caminho seguir



Paralelismo e Sicronismo

- Paralelismo permite que duas ou mais atividades sejam feitas em paralelo
- Sincronismo possibilita que sejam demarcados pontos de controle que obrigam a chegada dos fluxos para continuar a execução

Objetos

- Objetos são produtos requeridos ou gerados pela execução de atividades
- Um fluxo com objetos deixa implícito um fluxo de controle
- Um objeto é descrito por:
  - Nome
  - Estado (entre colchetes)

Objetos

- Caso se deseje representar os papéis responsáveis por cada atividade, pode ser utilizado o recurso de raias

- Raias podem ser horizontais ou verticais

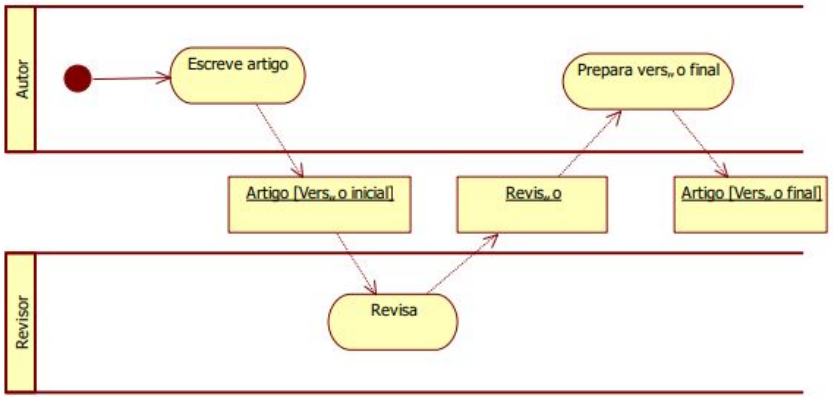


Diagrama de Atividades

Definição do Escopo

- **Nome do Produto:** Casa Segura

- **Missão do Produto:** o software a ser desenvolvido deverá fazer a gestão do dispositivo de segurança instalado nas casas dos clientes de uma seguradora, de forma que o proprietário da casa possa monitorar seu domicílio pela internet, ao acessar a interface do software desenvolvido.

Diagrama de Atividades

Especificação de Requisitos

Diagrama de Atividades para a função acessar a vigilância por câmeras via internet do sistema Casa Segura

Será necessário exibir visões das câmeras.

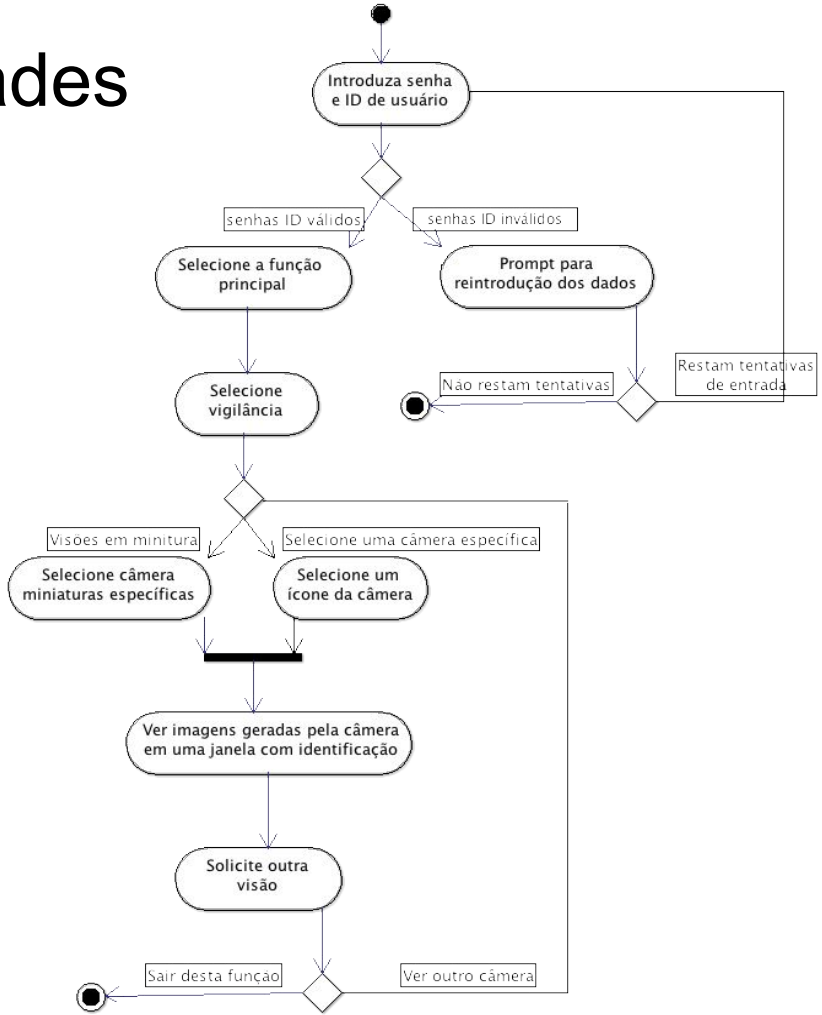
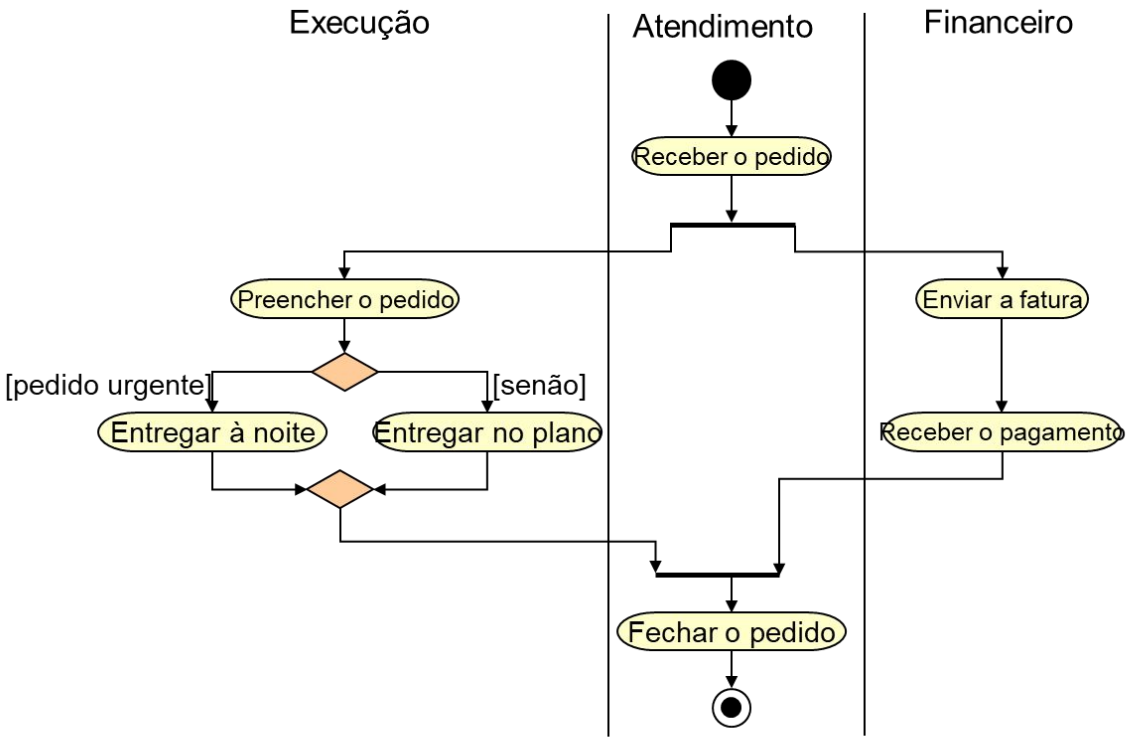


Diagrama de Atividades em Raias

Raias:

- Utilizadas para mostrar o fluxo de um processo que possui interação com 2 ou mais autores;
- Dizem quem faz o quê;
- Deve-se organizar o diagrama em zonas verticais separadas por linhas;
- Cada raia representa a responsabilidade de uma classe, ator, departamento, etc.
- Muito útil para o projeto de interfaces do sistema.

Diagrama de Atividades em Raias



Exemplo: Farmácia
Diagrama de Raias

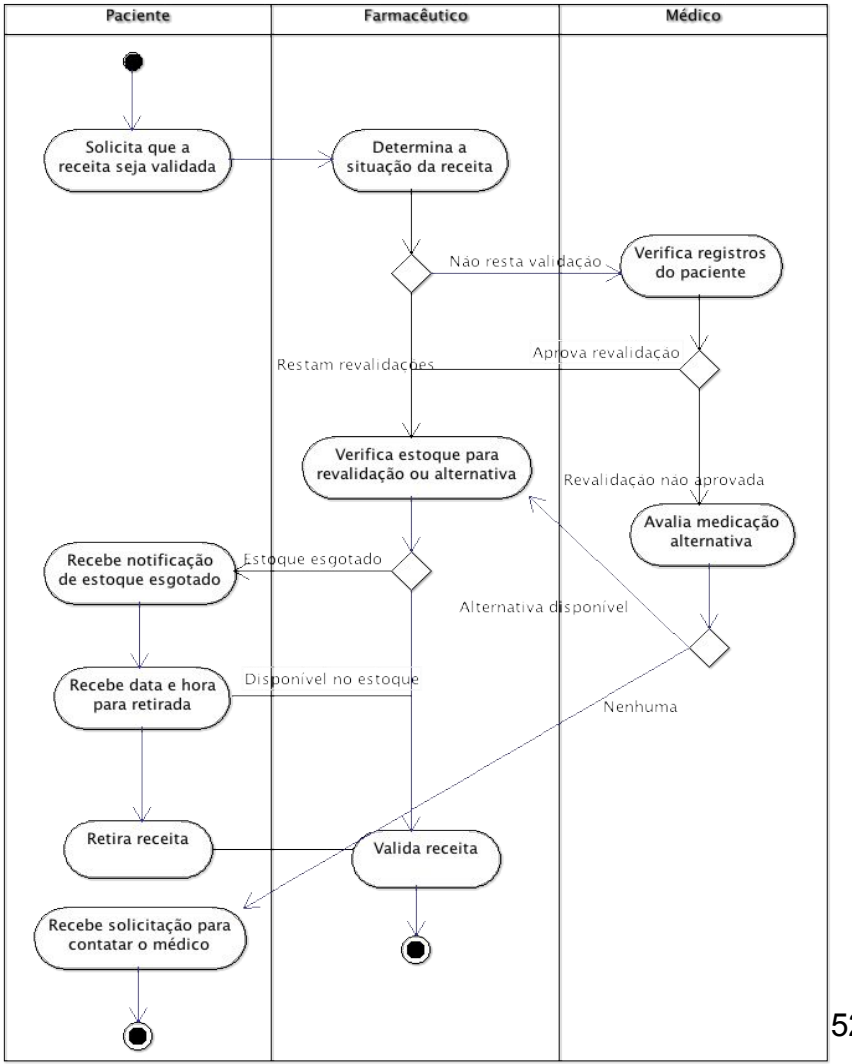


Diagrama de Atividades em Raias

- O fluxo de eventos no diagrama de raias nos permite reconhecer uma série de características-chave da interface;

- Cada usuário implementa tarefas distintas via interface, sendo necessárias interfaces diferente para médicos, farmacêuticos e pacientes;

- O projeto da interface para os farmacêuticos e médicos deve contemplar acesso e exibição de informações de fontes secundárias.

Diagrama de Atividades em Raias

- Como o Diagrama de Atividade representa uma sequência de ações dos atores no sistema, um uso bastante comum do Diagrama em Raias é mostrar a sequência em que ocorrem os Casos de Uso do sistema.

- Vamos analisar 2 exemplos:
  1. Sistema de Controle de uma Biblioteca
  2. Sistema de Controle de Emissões de Atas de Reunião

Diagrama de Casos de Uso - Biblioteca

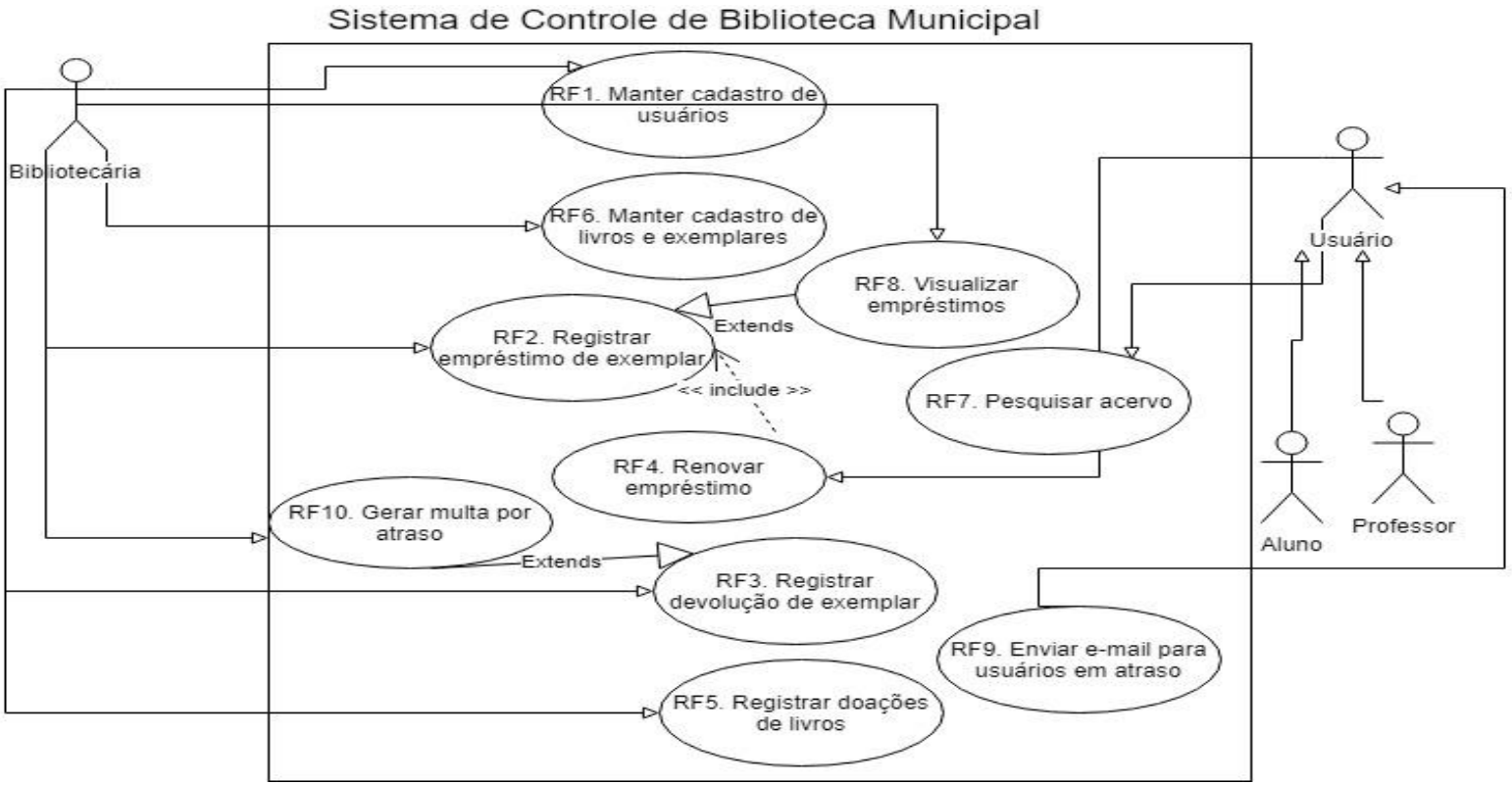


Diagrama de Atividades - Biblioteca

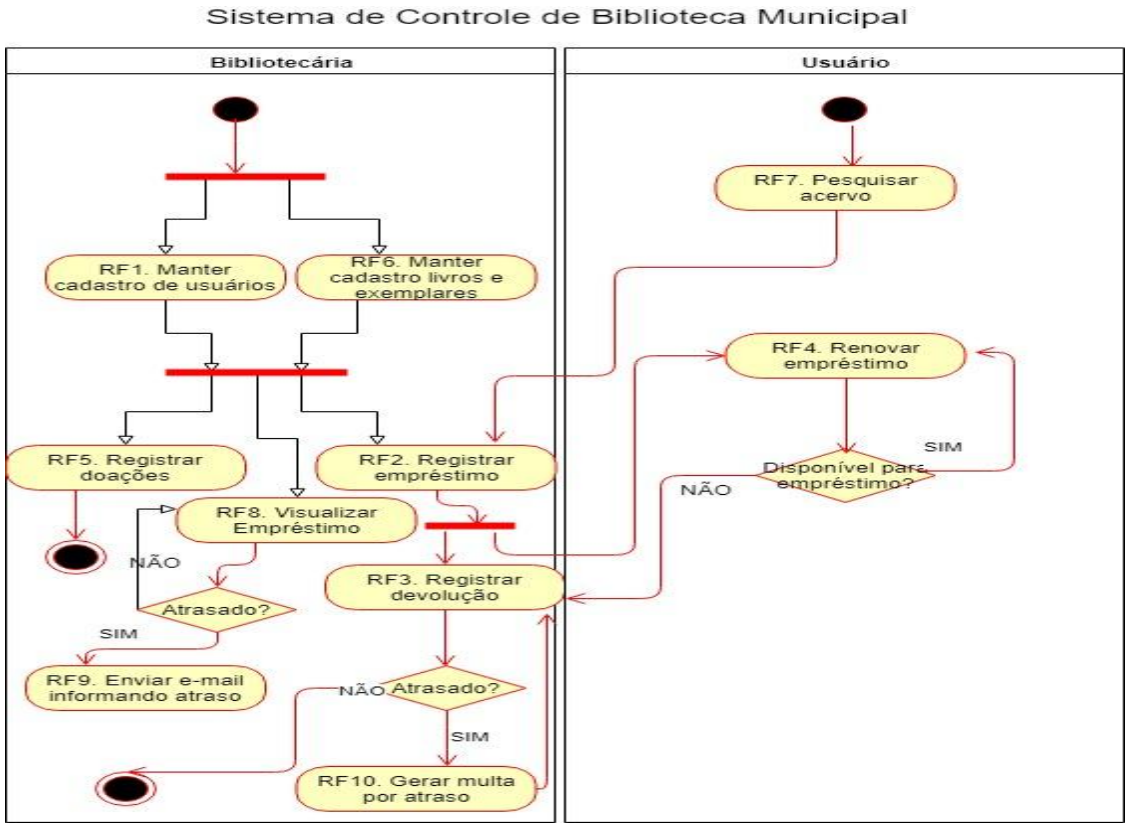


Diagrama de Casos de Uso – Atas de Reunião

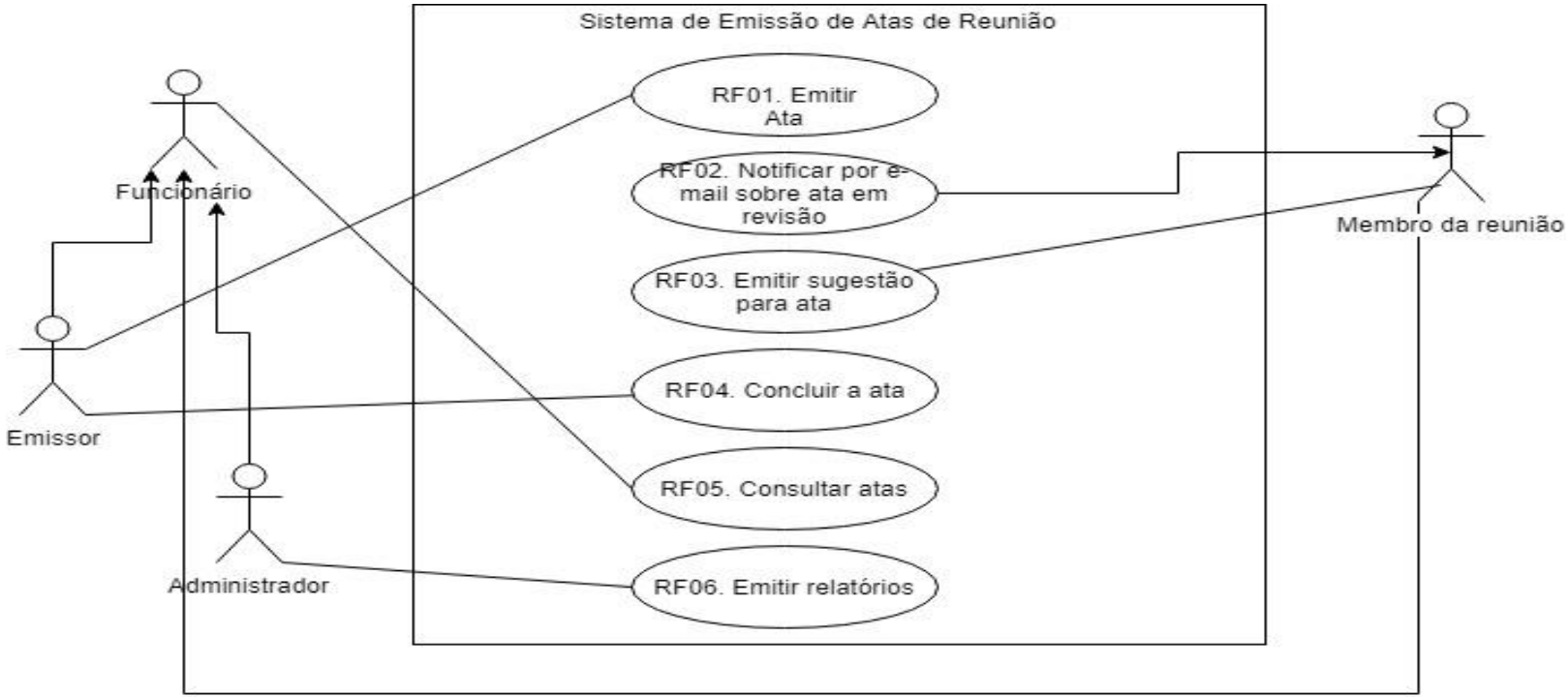
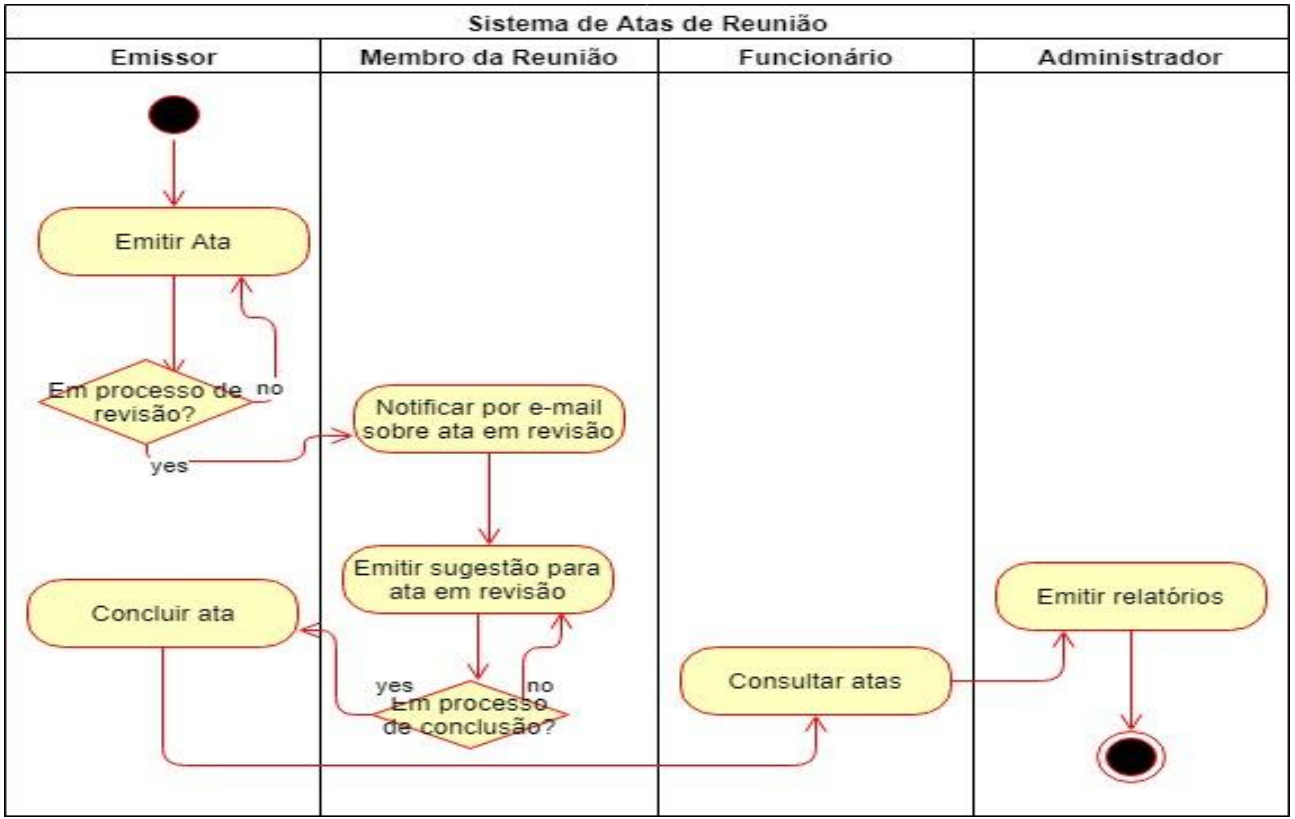


Diagrama de Atividades – Atas de Reunião



Modelagem de Software

Prof. Pedro Henrique Dias Valle

Adaptado de:
Prof. Fabrício Martins Mendonça

Exercício

- Elabore um diagrama de classes para um sistema de ponto de vendas
  - R01. O gerente deve poder fazer login com um ID e senha para iniciar e finalizar o sistema;
  - R02. O caixa (operador) deve poder fazer login com um ID e senha para poder utilizar o sistema;
  - R03. O sistema deve registrar a venda em andamento – os itens comprados;
  - R04. O sistema deve exibir a descrição e preço e do item registrado;
  - R05. O sistema deve calcular o total da venda corrente;
  - R06. O sistema deve tratar pagamento com dinheiro – capturar a quantidade recebida e calcular o troco;
  - R07. O sistema deve tratar pagamento com cartão de crédito – capturar a informação do cartão através de um leitor de cartões ou entrada manual e autorizar o pagamento utilizando o serviço de autorização de crédito (externo) via conexão por modem;
  - R08. O sistema deve tratar pagamento com cheque – capturar o número da carteira de identidade por entrada manual e autorizar o pagamento utilizando o serviço de autorização de cheque (externo) via conexão por modem;
  - R09. O sistema deve reduzir as quantidades em estoque quando a venda é confirmada;
  - R10. O sistema deve registrar as vendas completadas;
  - R11. O sistema deve controlar diversas lojas, com catálogo de produtos e preços unificado, porém estoques separados;