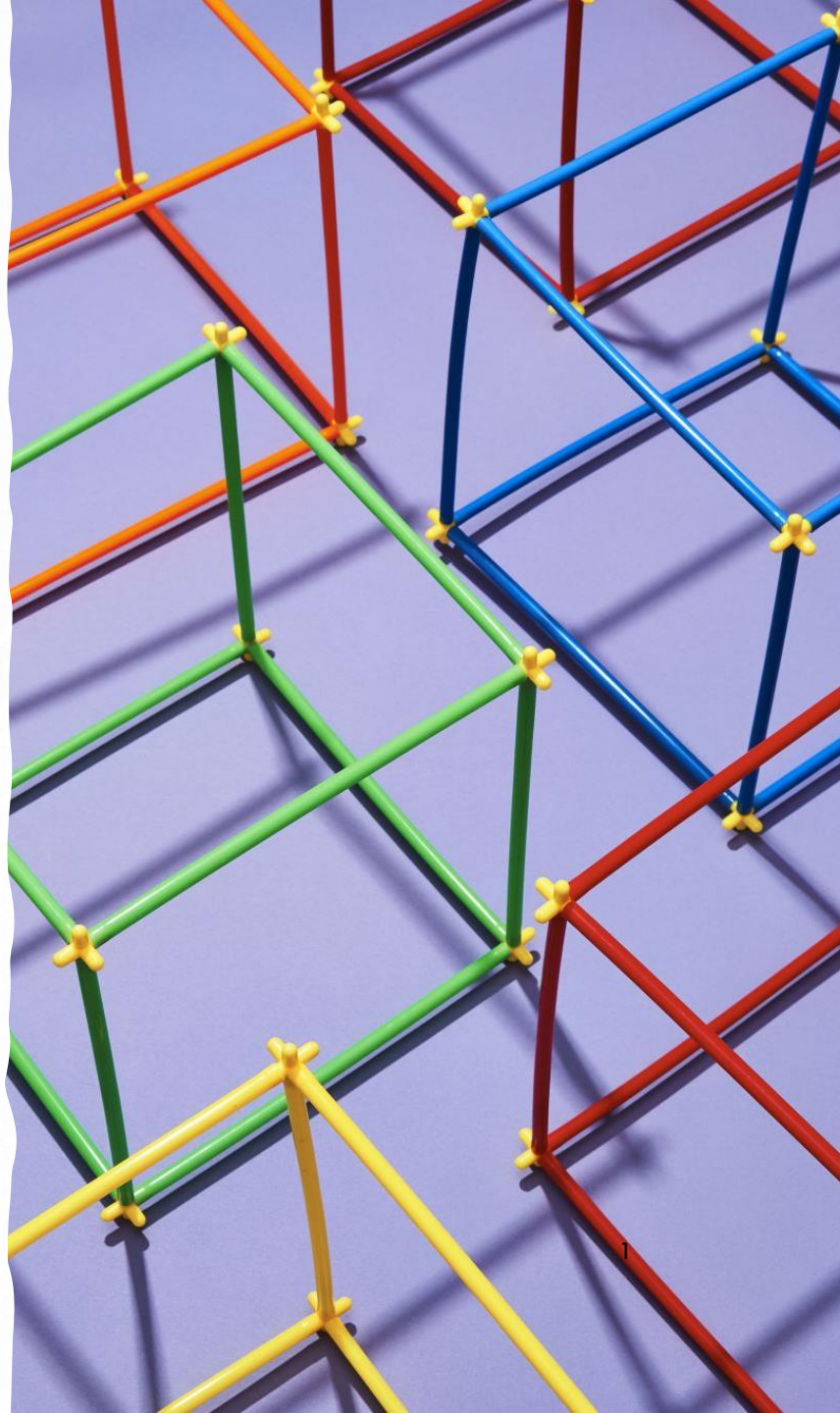
REVISÃO A ORIENTAÇÃO A OBJETOS

PROF. MA. MARINA GIROLIMETTO

MARINA.GIROLIMETTO@UFFS.EDU.BR



ORIENTAÇÃO A OBJETOS - CLASSE

- **A UML está totalmente inserida no paradigma de orientação a objetos.**

- Conceitos simples que aprendemos desde a infância, como pessoa, carro e casa, por exemplo, são **classes**, ou seja, grupos de objetos, **tendo características e comportamentos** de qualquer objeto do grupo em questão.

ORIENTAÇÃO A OBJETOS - CLASSE

- Um **objeto amarelo e o objeto vermelho podem ter, ambos, a mesma classificação: carro**. “Carro” é um termo geral que se refere a muitos objetos.
- Cada **um dos objetos-carro tem características semelhantes entre si:**
  - todos têm quatro rodas;
  - tem no mínimo, duas portas;
  - tem luzes de farol e freio;
  - tem vidros frontais e laterais;
  - transportar pessoas de um lugar para outro.

ORIENTAÇÃO A OBJETOS – INSTÂNCIA DA CLASSE

- O **objeto é um exemplo do grupo carro**, ou seja, uma instância da classe carro. Assim, **instanciação constitui-se simplesmente em criar um exemplo de um tipo, um grupo, uma classe.**

- Todos os objetos da classe carro possuem o atributo placa, mas cada um dos objetos possui um valor diferente para sua placa específica.

CLASSES DE OBJETOS

- Uma classe representa uma categoria e os objetos são os membros ou exemplos dessa categoria.
- Na UML, uma classe é representada por um retângulo que pode ter até três divisões.
- **A primeira divisão armazena o nome pelo qual a classe é identificada (e essa é a única divisão obrigatória), a segunda enuncia os possíveis atributos pertencentes à classe e a terceira lista as possíveis operações (métodos) que a classe contém.**

Figura 2.1 – Exemplo de uma classe.

ATRIBUTOS OU PROPRIEDADES

- **Os atributos representam as características**, são apresentados na segunda divisão da classe e contêm o nome do atributo e o tipo (integer, float, string...). Exemplos: o nome, o cpf ou a idade em um objeto da classe pessoa ou a placa, a cor em um objeto da classe carro.

- Os atributos podem assumir valores diversos em cada instância. Ex:
  pessoa1 – nome: “João”;
  pessoa2 – nome: “Paulo”.

Figura 2.2 – Exemplo de classe com atributos.

OPERAÇÕES, MÉTODOS OU COMPORTAMENTOS

- Classes costumam ter métodos. **Um método representa uma atividade que um objeto de uma classe pode executar.** Este, pode receber ou não parâmetros e tende a retornar valores.
- Os métodos são armazenados na terceira divisão de uma classe.

Figura 2.3 – Exemplo de classe com métodos.

VISIBILIDADE

- **A visibilidade é utilizada para indicar o nível de acessibilidade de um determinado atributo ou método.**
- Modos de visibilidade:
  - Público (+): pode ser utilizado por qualquer objeto;
  - Protegido (#): objetos da classe detentora e os de suas subclasses poderão ter acesso a este;
  - Privado (-): somente os objetos da classe detentora poderão enxergá-lo; e
  - Pacote (~): é visível por qualquer objeto dentro do pacote.

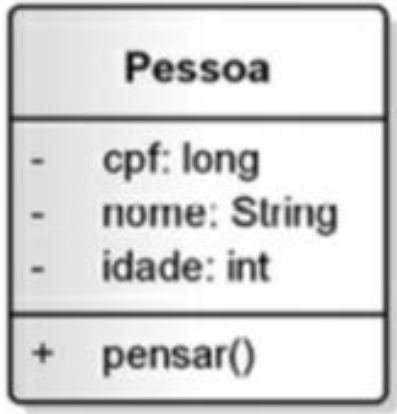


Figura 2.4 – Exemplo de Visibilidade.

VISIBILIDADE

- **Normalmente os atributos costumam ser privados ou protegidos, enquanto os métodos costumam ser públicos.**
- Um atributo privado, além de só ser visível por objetos de sua classe, só poderá ser acessado por meio de métodos. Assim, objetos de outras classes não terão conhecimento sobre quais atributos estão contidos na classe em questão e nem poderão acessá-los.

HERANÇA

- **A herança permite o reaproveitamento de atributos e métodos.**
- Trabalha com os conceitos de superclasses e subclasses.
- Uma superclasse (classe mãe), contém classes derivadas dela, chamadas subclasses (classes-filha).
- **A vantagem do uso da herança é que não precisamos redeclarar os atributos e métodos previamente definidos**: a subclasse herda-os automaticamente, permitindo reutilização do código já pronto.

HERANÇA

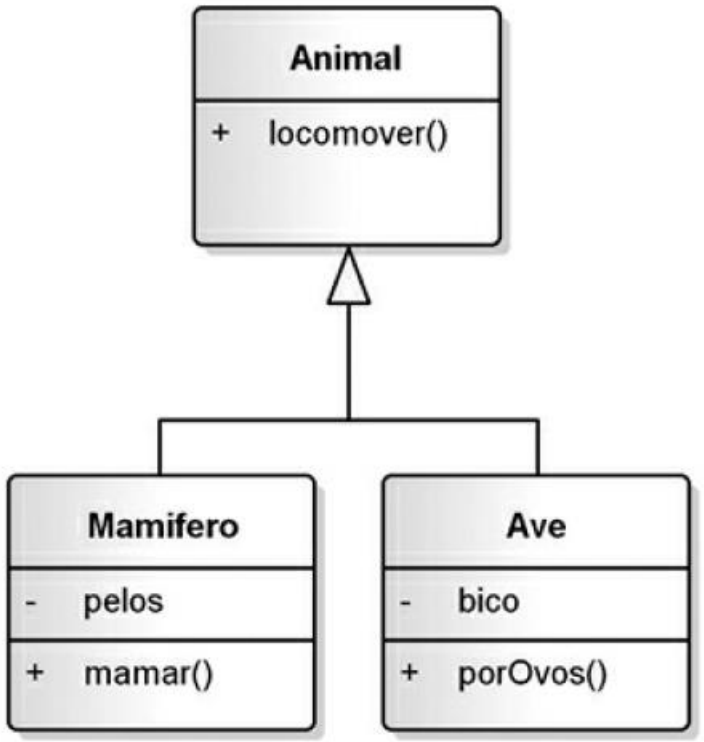


Figura 2.5 – Exemplo de Herança.

- A herança permite trabalhar com especializações. Pode-se criar classes gerais, com características compartilhadas por muitas classes, mas que tenham pequenas diferenças entre si.
- Uma subclasse pode se tornar uma superclasse a qualquer momento, bastando para tanto que se derive uma subclasse dela.

HERANÇA MÚLTIPLA

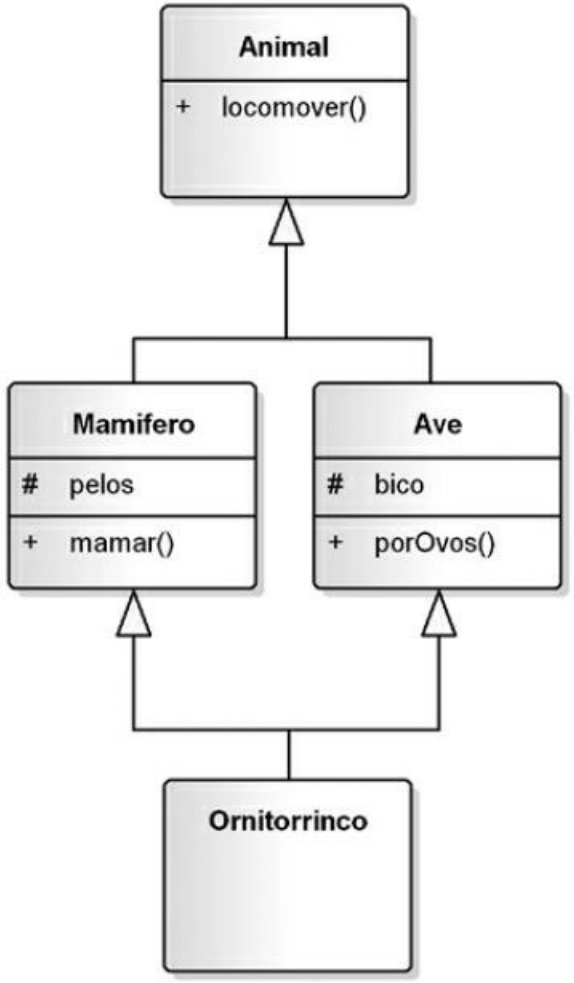


Figura 2.6 – Exemplo de Herança Múltipla.

- A herança múltipla ocorre quando uma subclasse herda características de duas ou mais superclasses.
- No caso, uma subclasse pode herdar atributos e métodos de diversas superclasses.

POLIMORFISMO

- **O polimorfismo trabalha com a redeclaração de métodos previamente herdados por uma classe**.
- Esses métodos, embora semelhantes, diferem de alguma forma da implementação utilizada na superclasse, sendo necessário, portanto, reimplementá-los na subclasse.

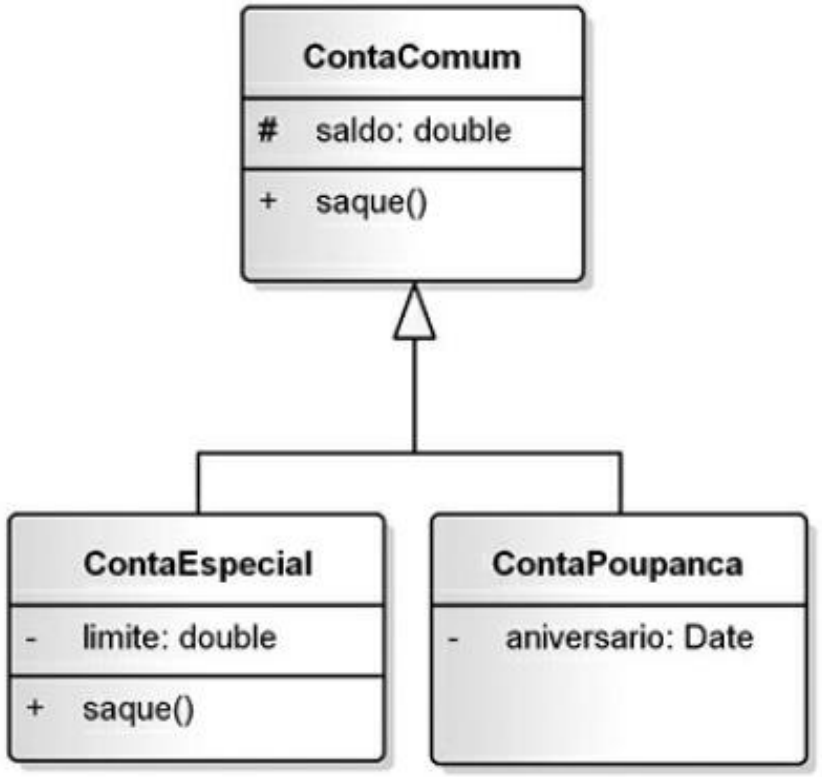


Figura 2.7 – Exemplo de Polimorfismo.

DIAGRAMAS DA UML

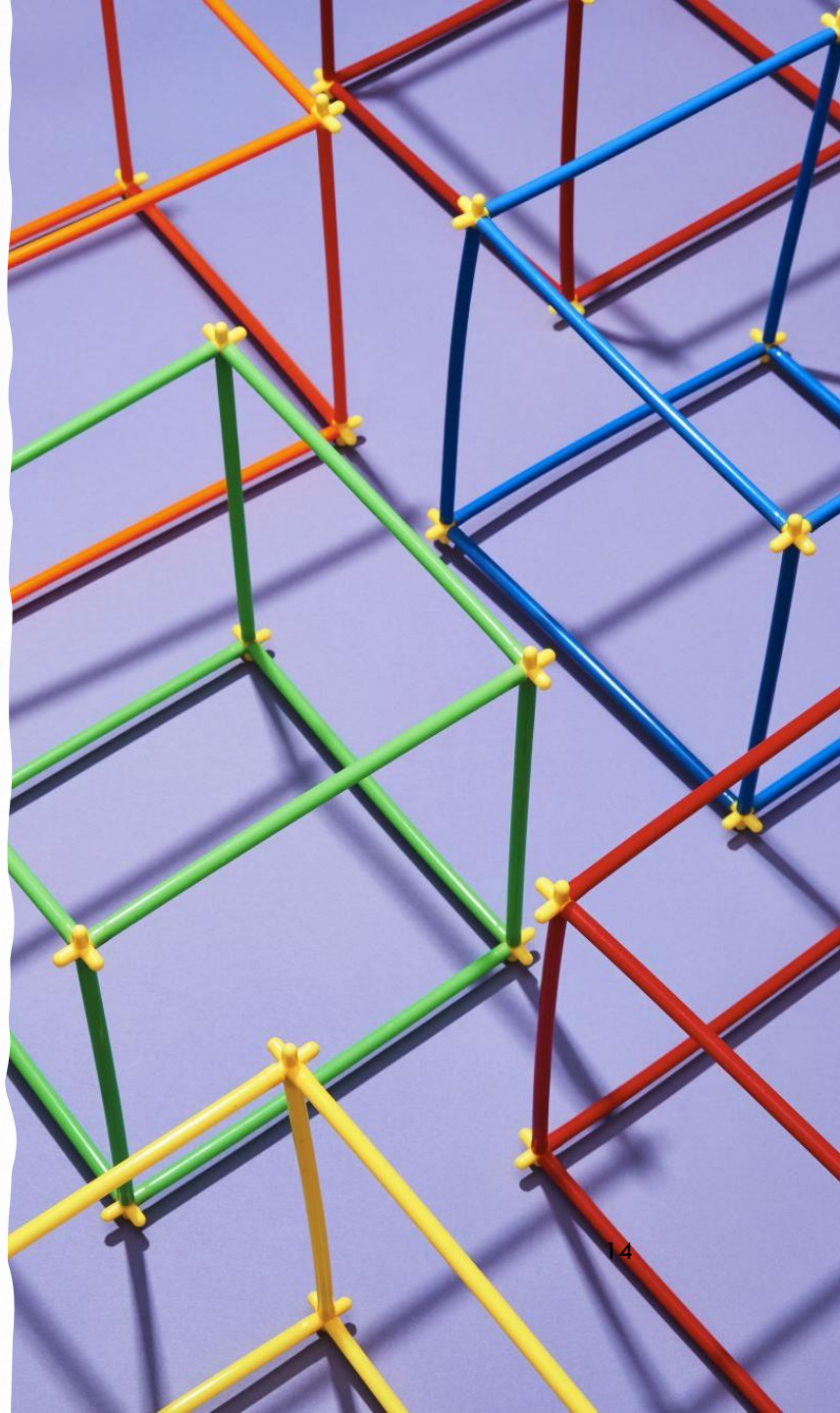


Diagrama de Casos de Uso

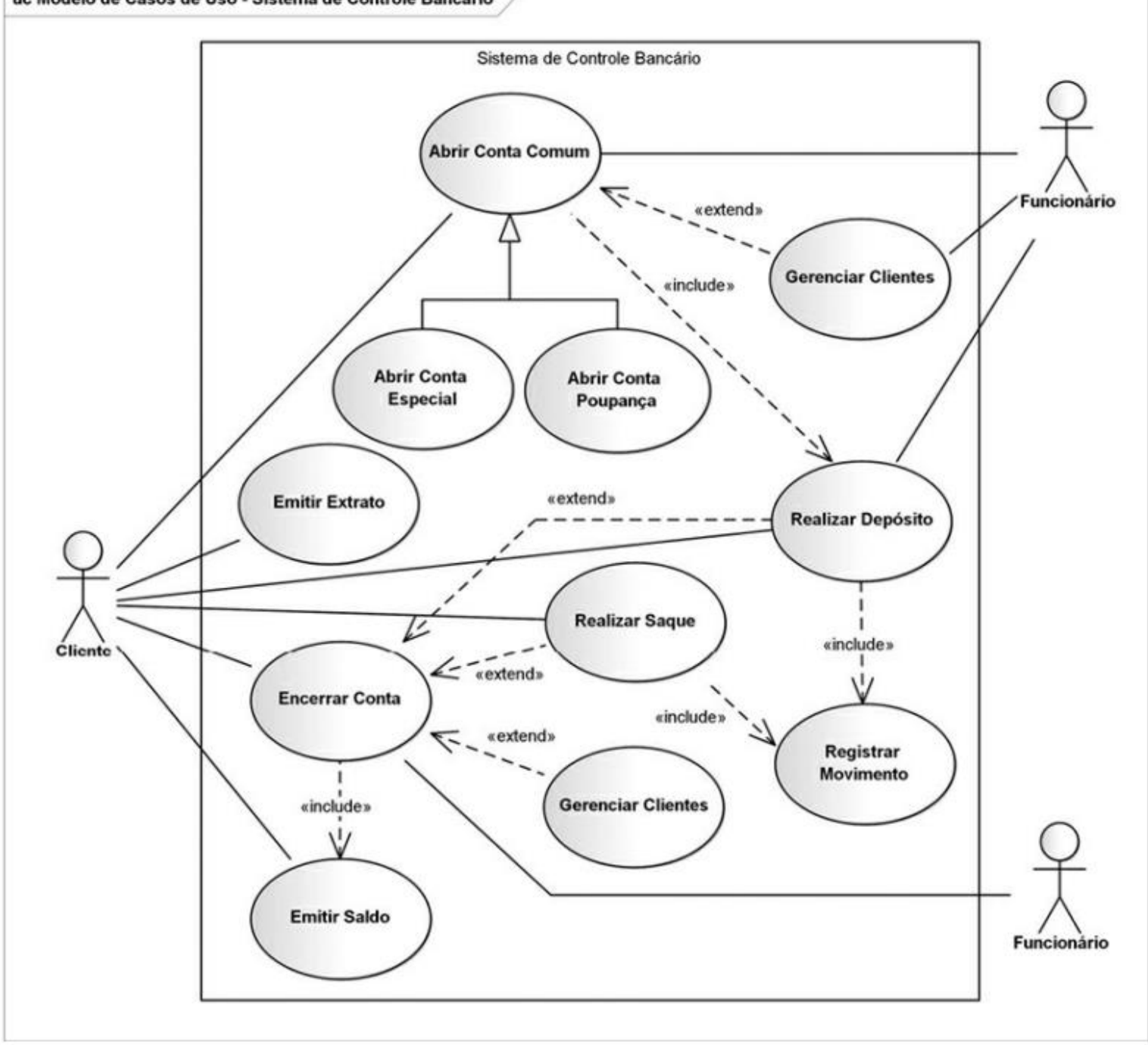


Figura 1.1 – Exemplo de Diagrama de Casos de Uso.

- Apresenta uma visão externa geral das funcionalidades que o sistema deverá oferecer aos usuários.
- Não se preocupa muito como tais funcionalidades serão implementadas.
- Serve de base para diversos outros diagramas.

Diagrama de Classes

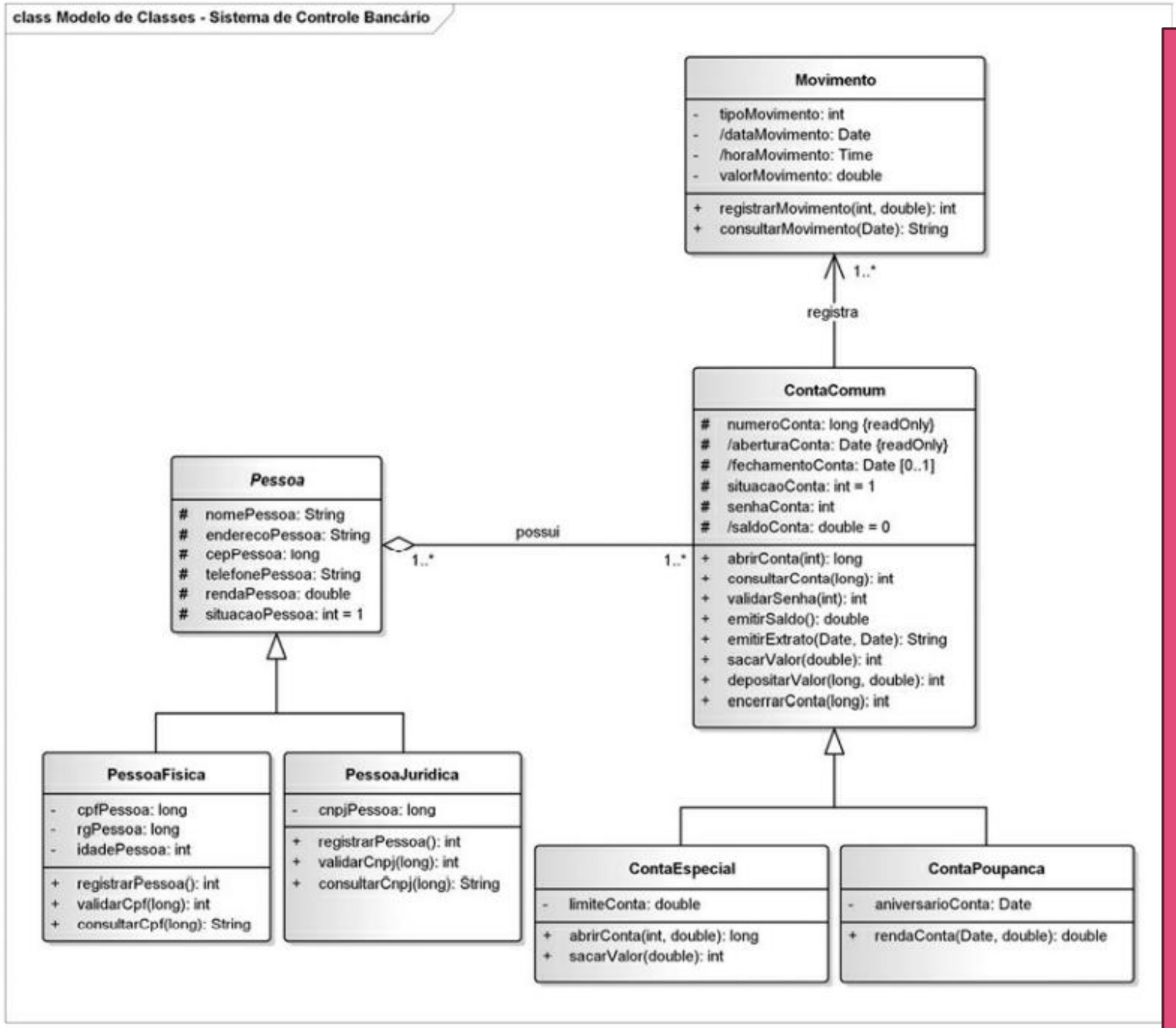


Figura 1.2 – Exemplo de Diagrama de Classes.

- Permite a visualização das classes que comporão o sistema com seus respectivos atributos e métodos.
- Demonstra como as classes se relacionam, complementam e transmitem informações entre si.
- Visão estática de como as classes estão organizadas.
- Apoio para a construção da maioria dos outros diagramas.

Diagrama de Objetos

- Complemento do diagrama de classes.
- Fornece visão dos valores dos objetos.

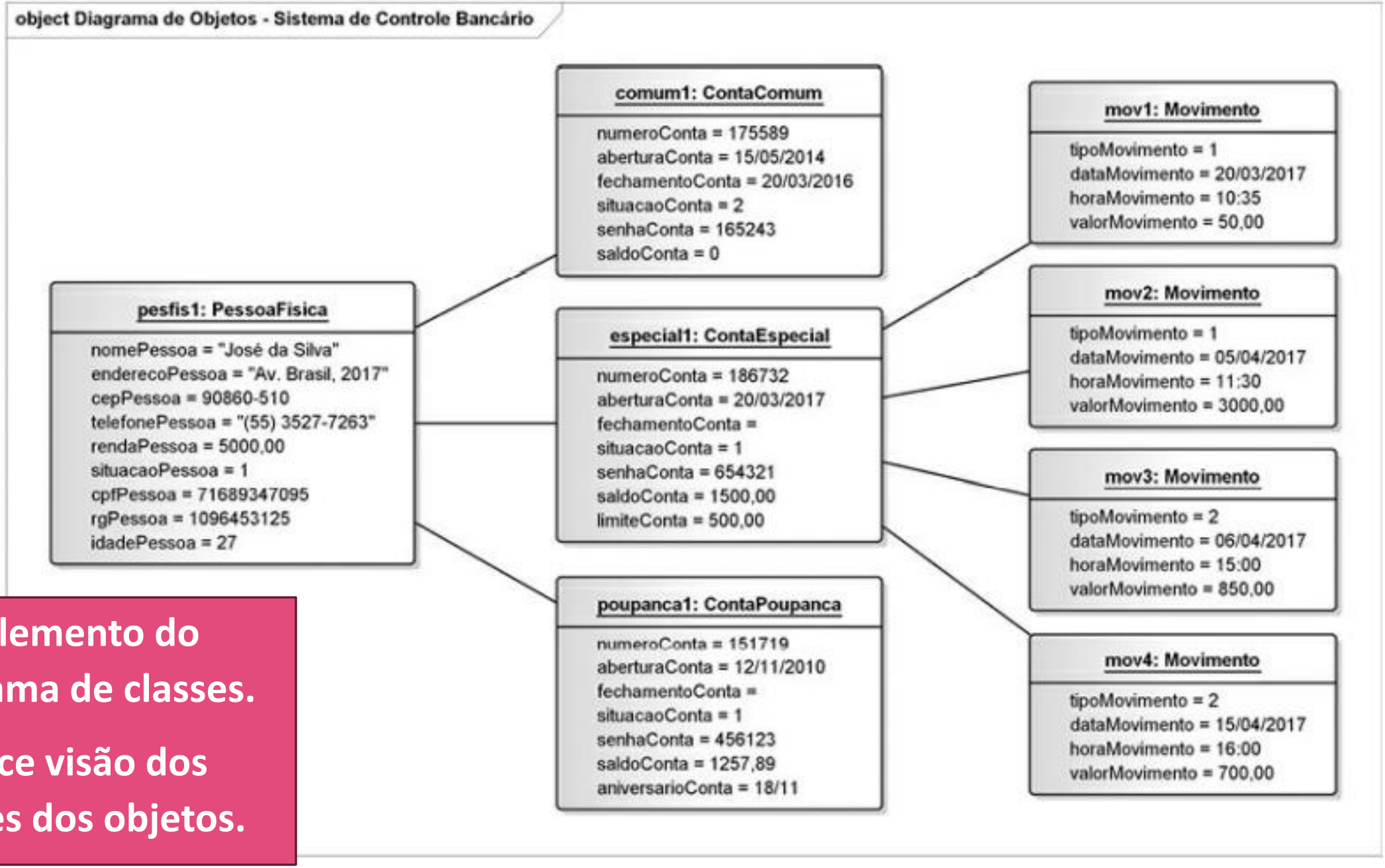
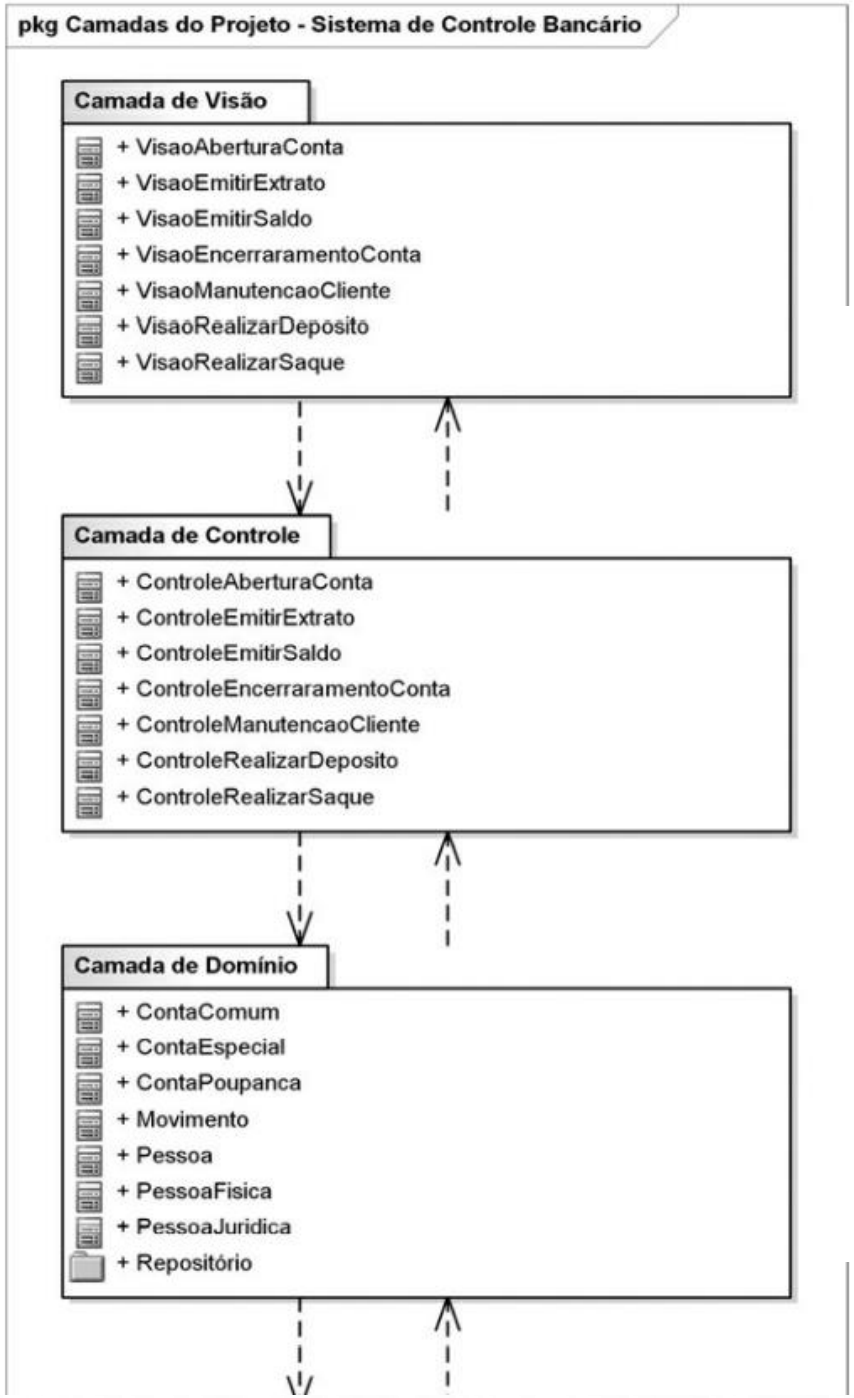


Figura 1.3 – Exemplo de Diagrama de Objetos.

Diagrama de Pacotes



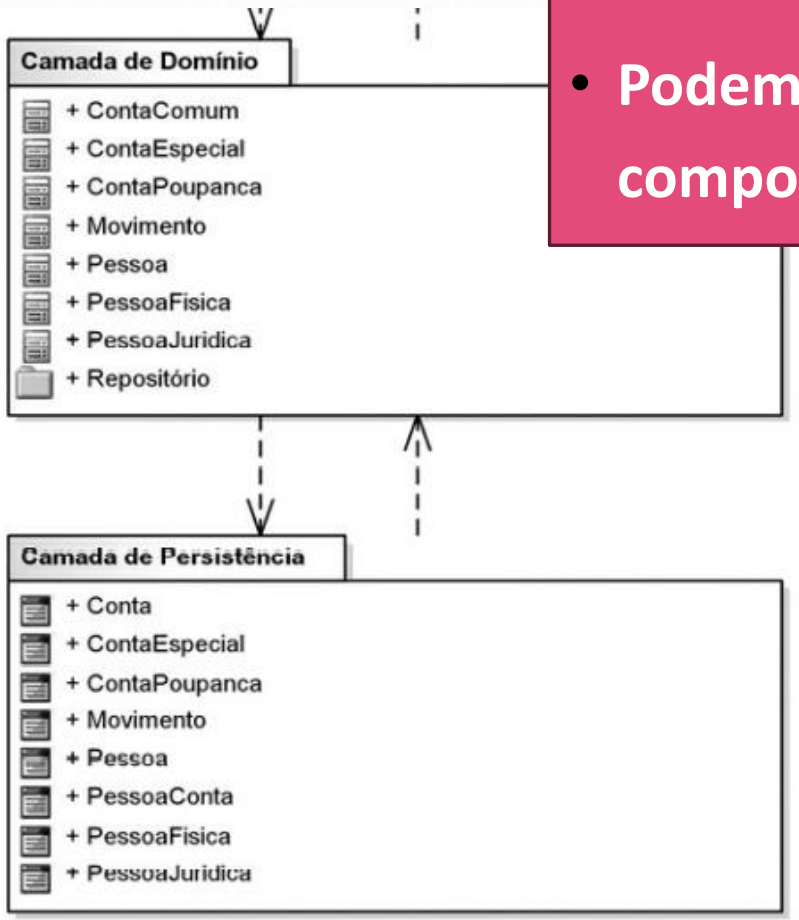


Figura 1.4 – Exemplo de Diagrama de Pacotes.

- **Representa como os elementos estão divididos logicamente.**
- **Podem ser subsistemas, componentes, camadas...**

Diagrama de Sequência

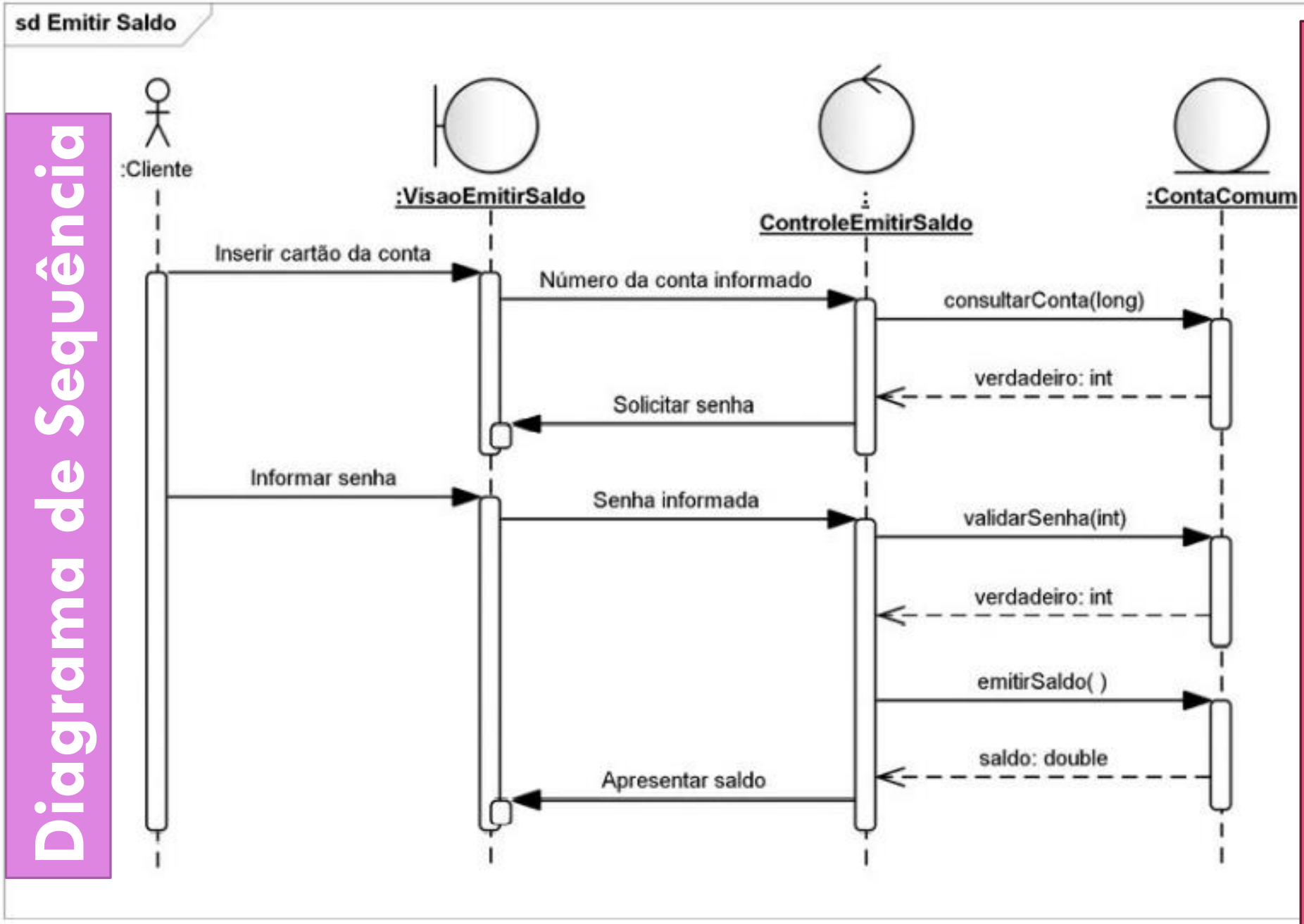
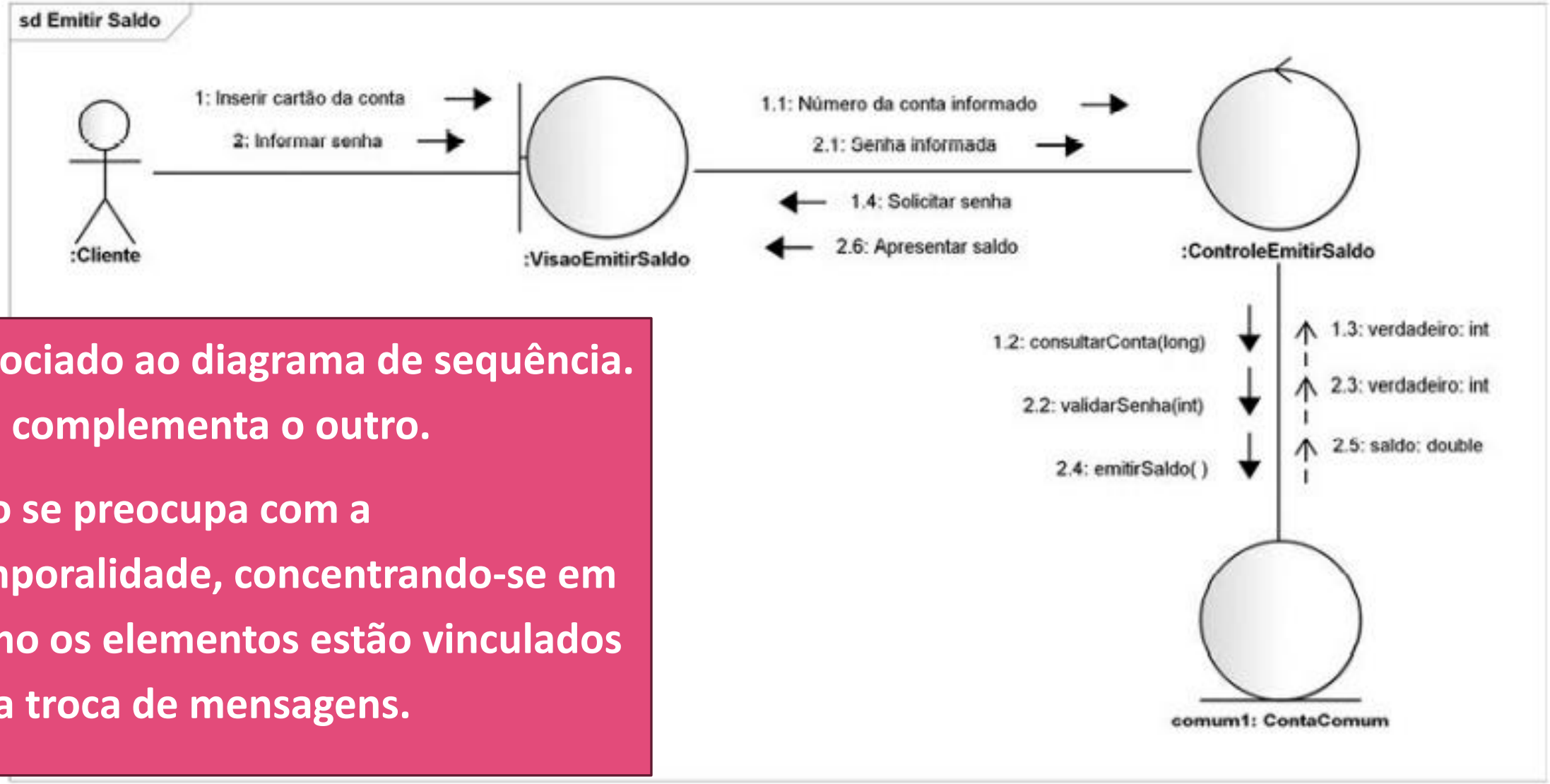


Figura 1.5 – Exemplo de Diagrama de Sequência.

- É comportamental e se preocupa com a ordem temporal em que as mensagens são trocadas entre os objetos envolvidos.
- Pode se basear em um caso de uso e se apoia no diagrama de classes.
- Identifica evento, ator, processo por meio de métodos disparados por mensagens entre objetos.

Diagrama de Comunicação



- Associado ao diagrama de sequência. Um complementa o outro.
- Não se preocupa com a temporalidade, concentrando-se em como os elementos estão vinculados e na troca de mensagens.

Figura 1.6 – Exemplo de Diagrama de Comunicação.

Diag. de Máquina de Estados

- Demonstra o comportamento de um elemento por meio de conjunto finito de transições de estado.
- Expressa comportamento de parte de sistema.
- Elemento modelado pode ser instância de uma classe ou caso de uso.

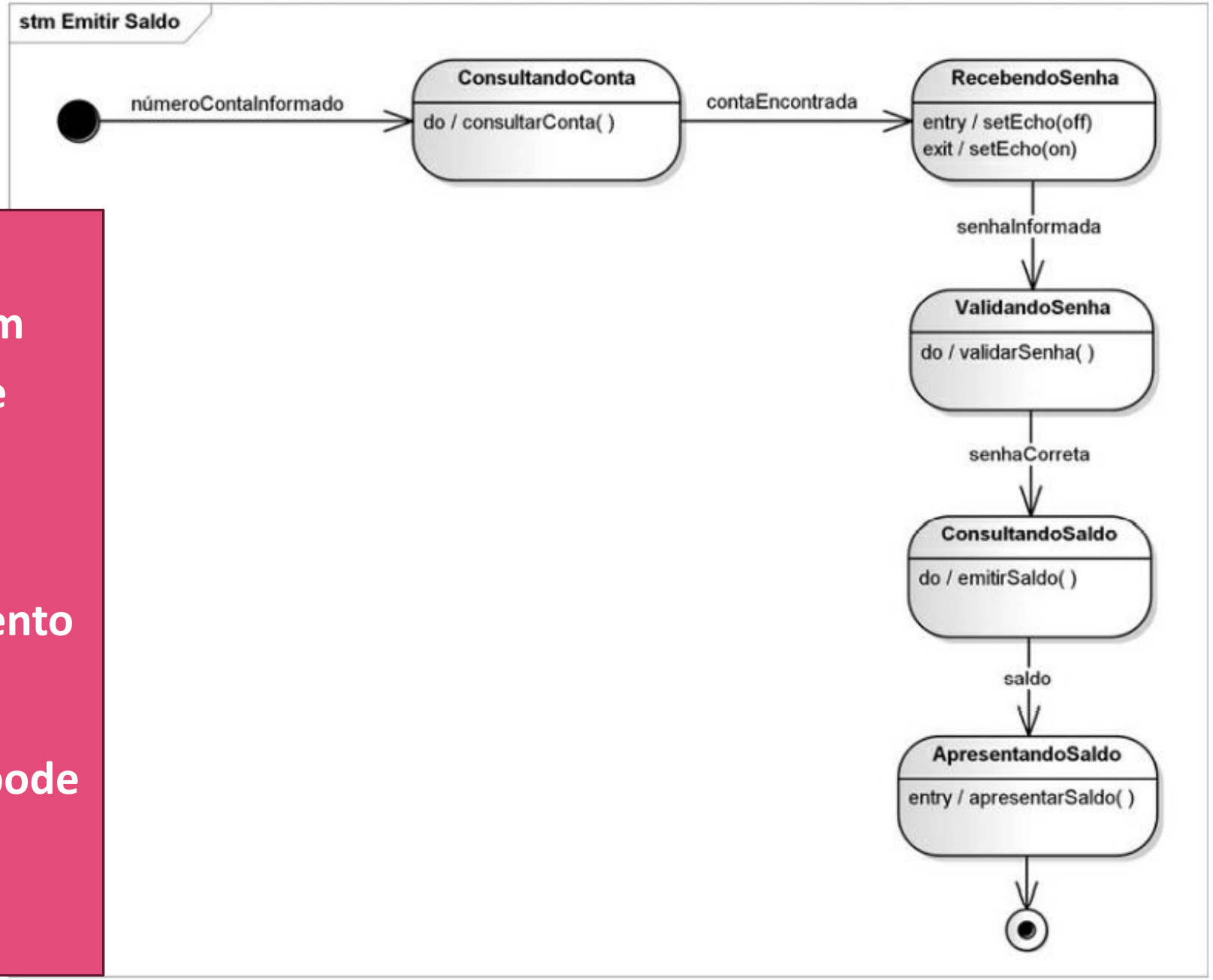


Figura 1.7 – Exemplo de Diagrama de Máquina de Estados.

Diagrama de Atividades

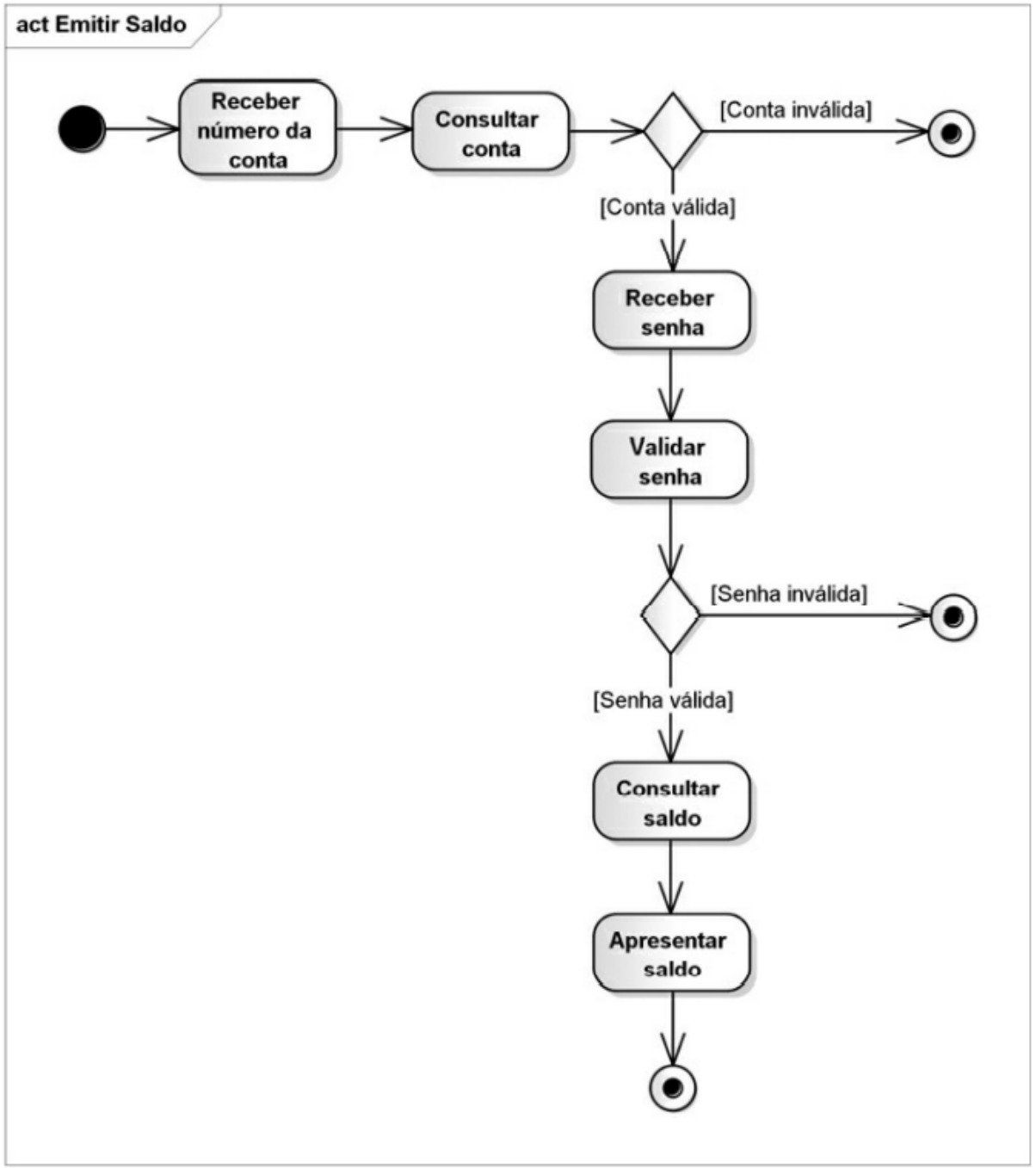
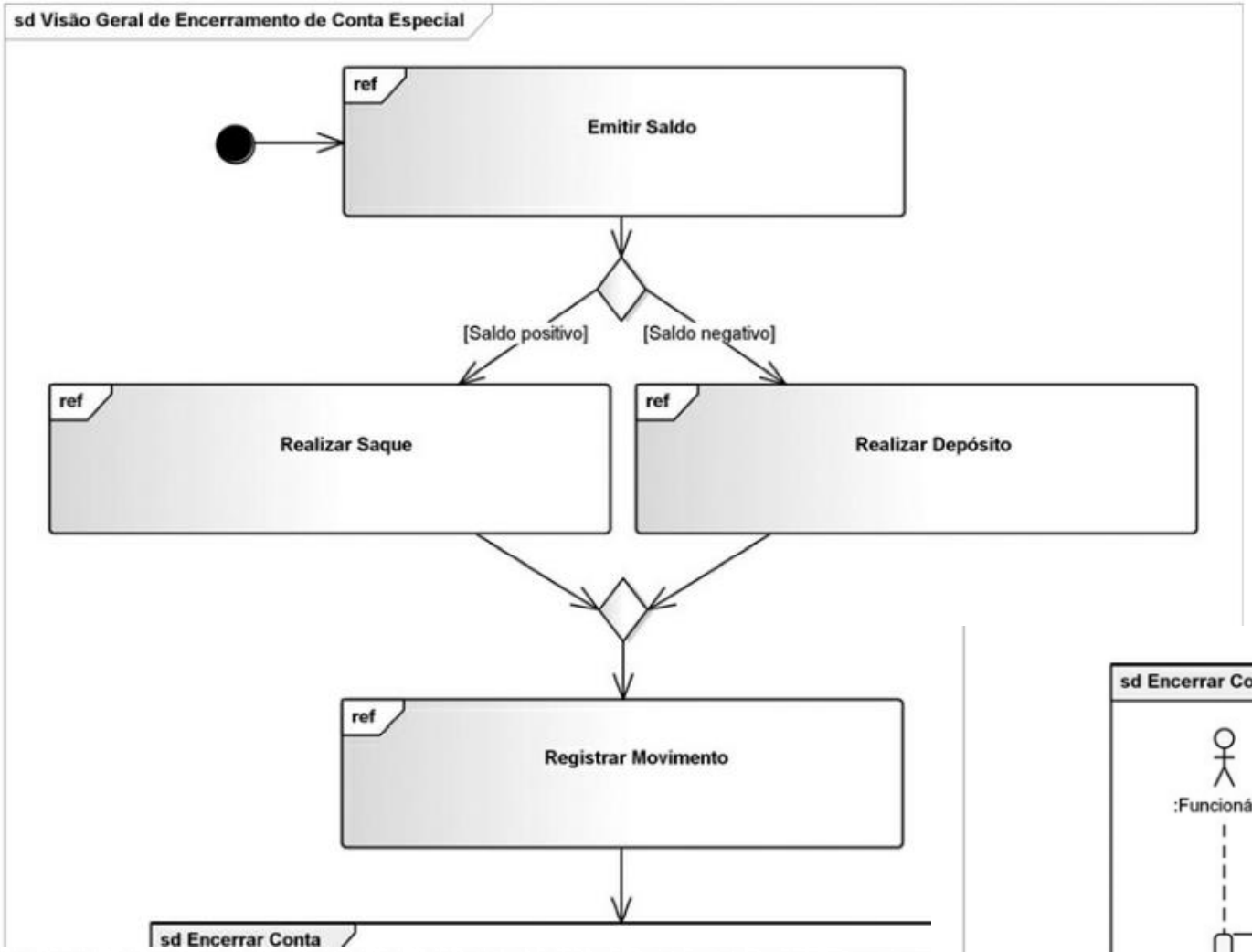


Figura 1.8 – Exemplo de Diagrama de Atividade.

- **Descreve os passos/fluxo para a conclusão de uma atividade específica (método, algoritmo, processo completo).**

D. de Visão Geral de Interação

- Variação do diagrama de atividade.
- Fornece visão geral de um sistema ou processo de negócio e seus subprocessos.



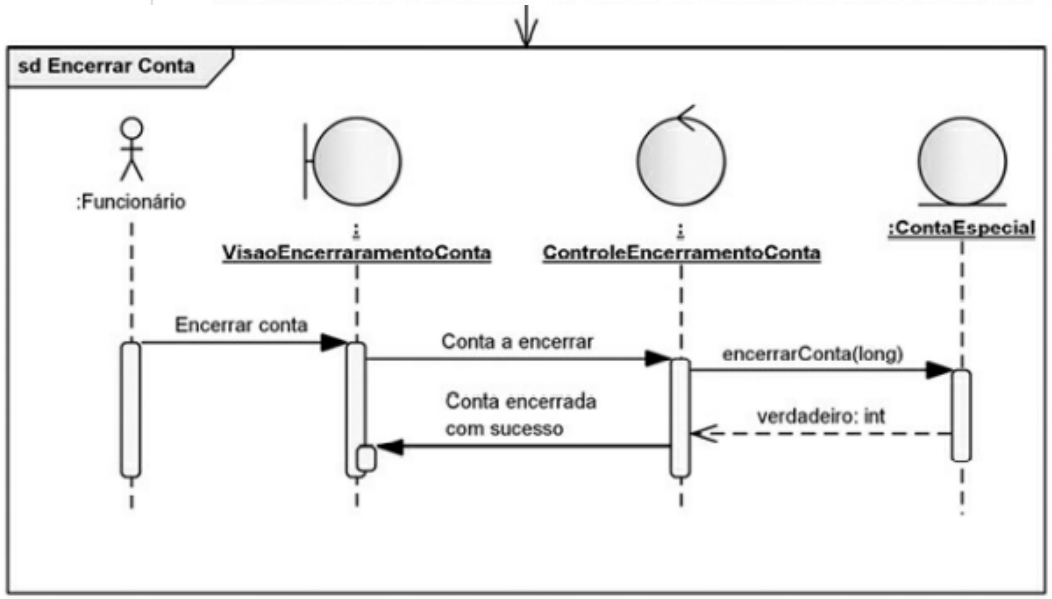


Figura 1.9 – Exemplo de Diagrama de Visão Geral de Interação.

Diagrama de Componentes

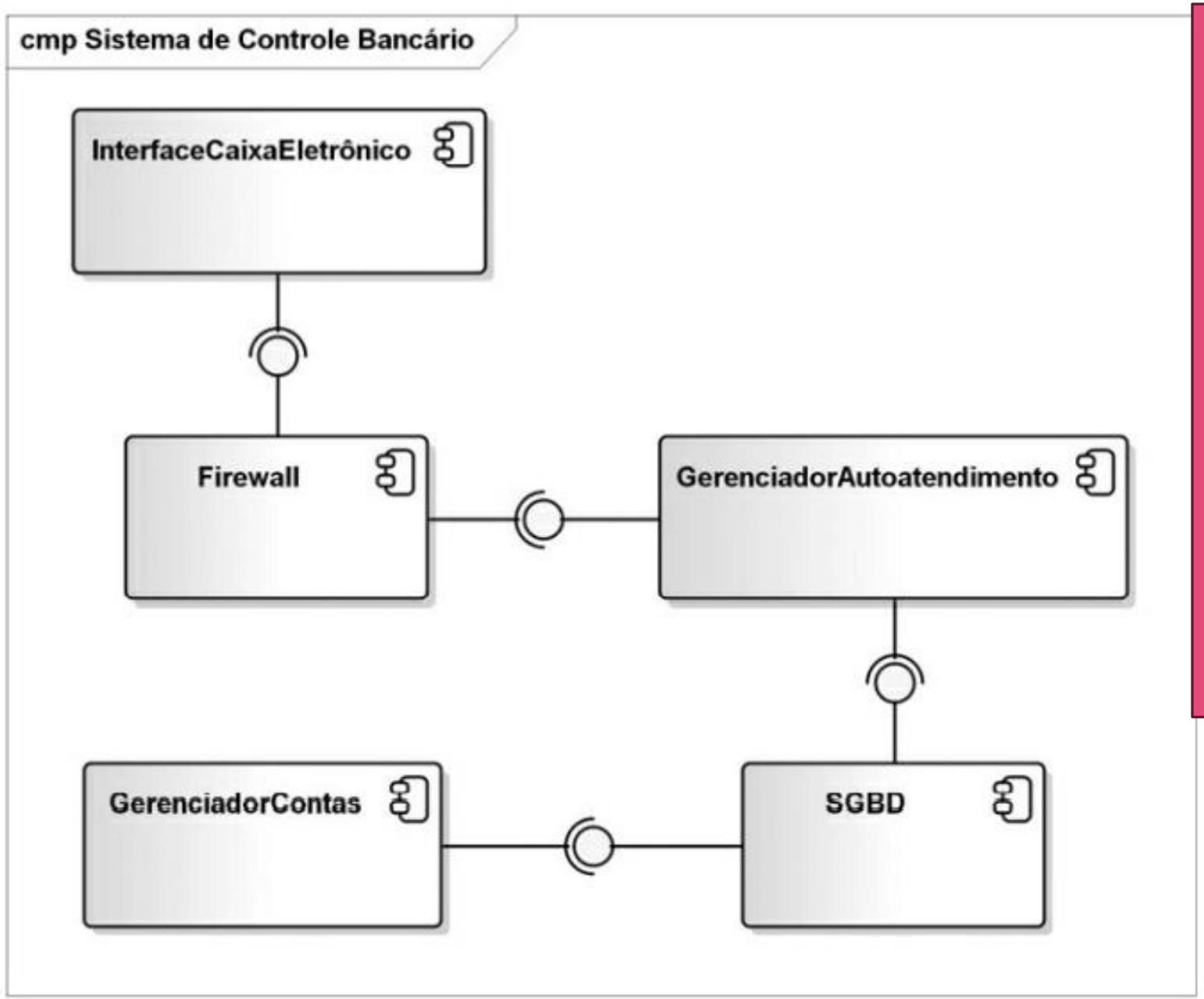


Figura 1.10 – Exemplo de Diagrama de Componentes.

- Identifica componentes que fazem parte do sistema, subsistema ou classes internas de componente individual.
- Componente pode ser lógico, físico (código-fonte, bibliotecas, arquivos executáveis...).

Diagrama de Implantação

- **Determina as necessidades de hardware do sistema (servidores, estações, topologia, protocolos, servidores...).**

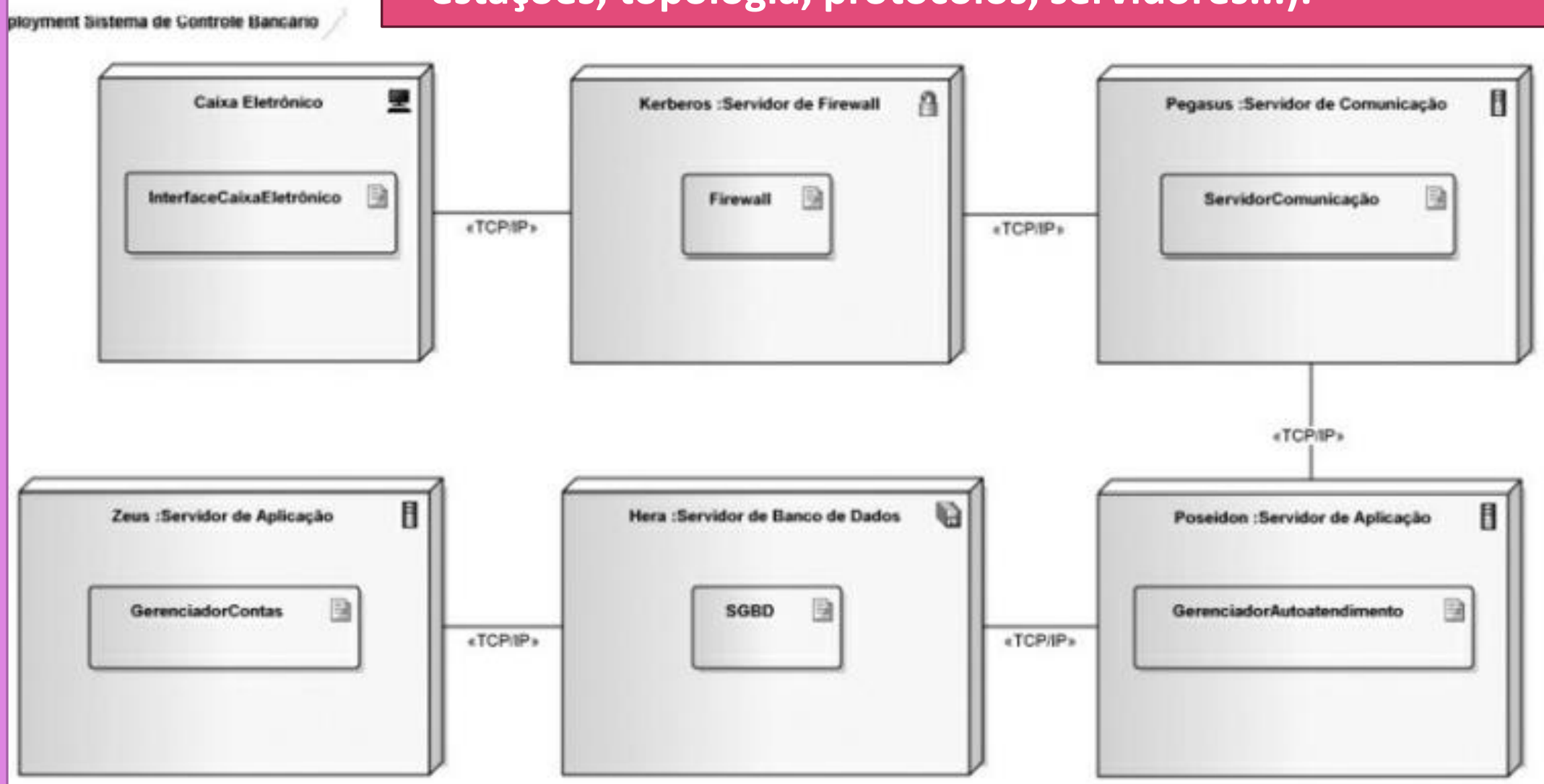


Figura 1.11 – Exemplo de Diagrama de Implantação.

Diag. de Estrrutura Composta

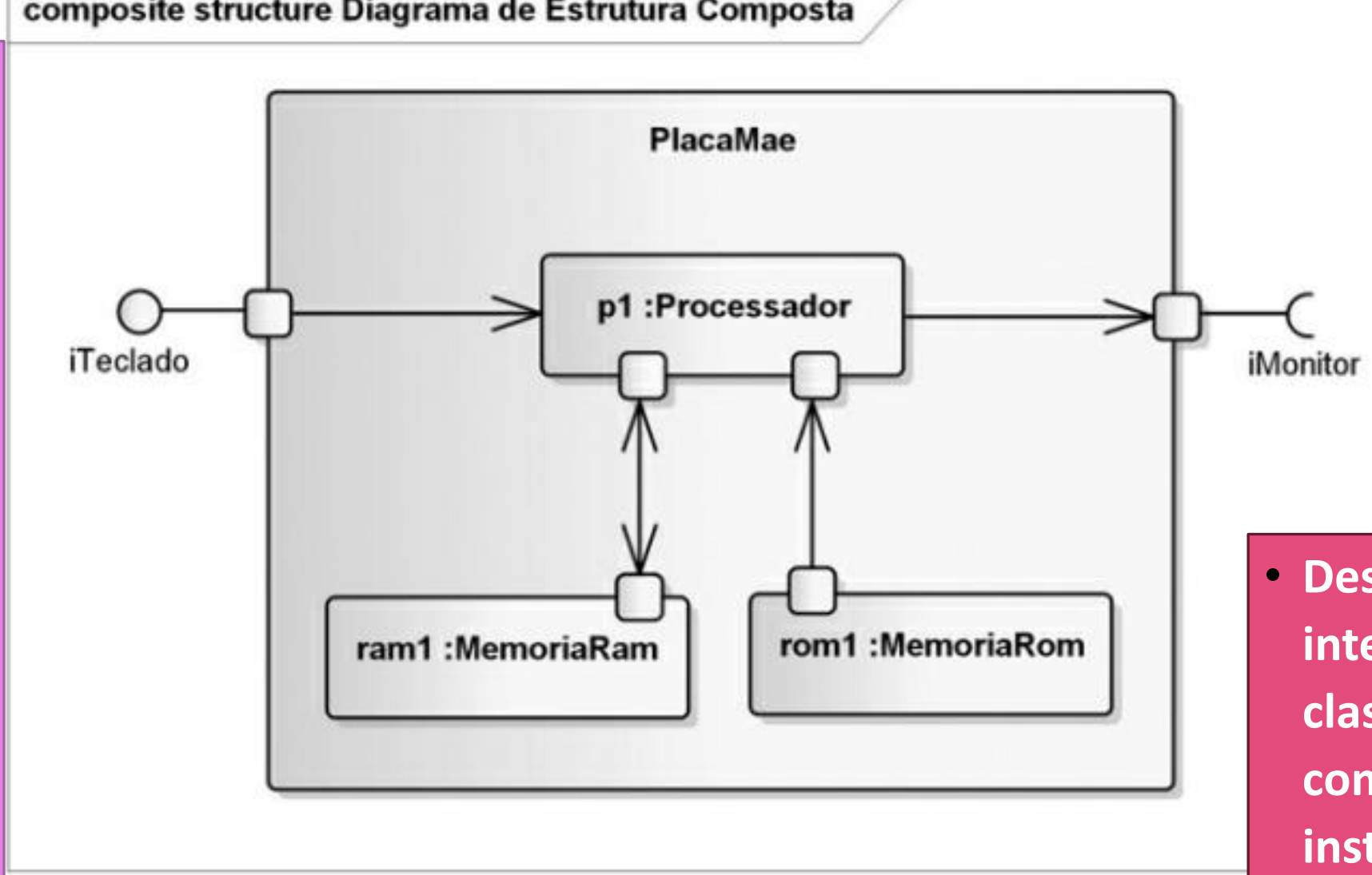


Figura 1.12 – Exemplo de Diagrama de Estrutura Com

- **Descreve a estrutura interna de um classificador (classe ou componente) ou instância, detalhando partes internas e como se comunicam e colaboram entre si.**

D. de Tempo ou Temporização

- Descreve a mudança no estado ou condição de instância ou papel que assume em determinado momento.
- Pode ser usado em sistemas de tempo real, de multimídia, processos de rede em que o tempo ou sincronismo de eventos é importante.

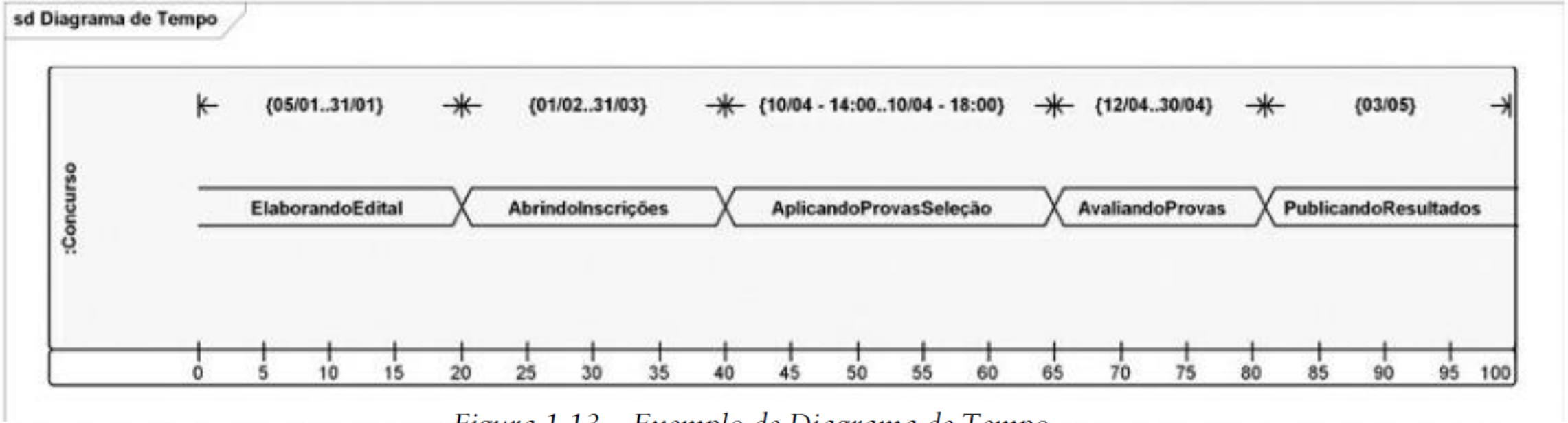


Figura 1.13 – Exemplo de Diagrama de Tempo.

Diagrama de Perfil

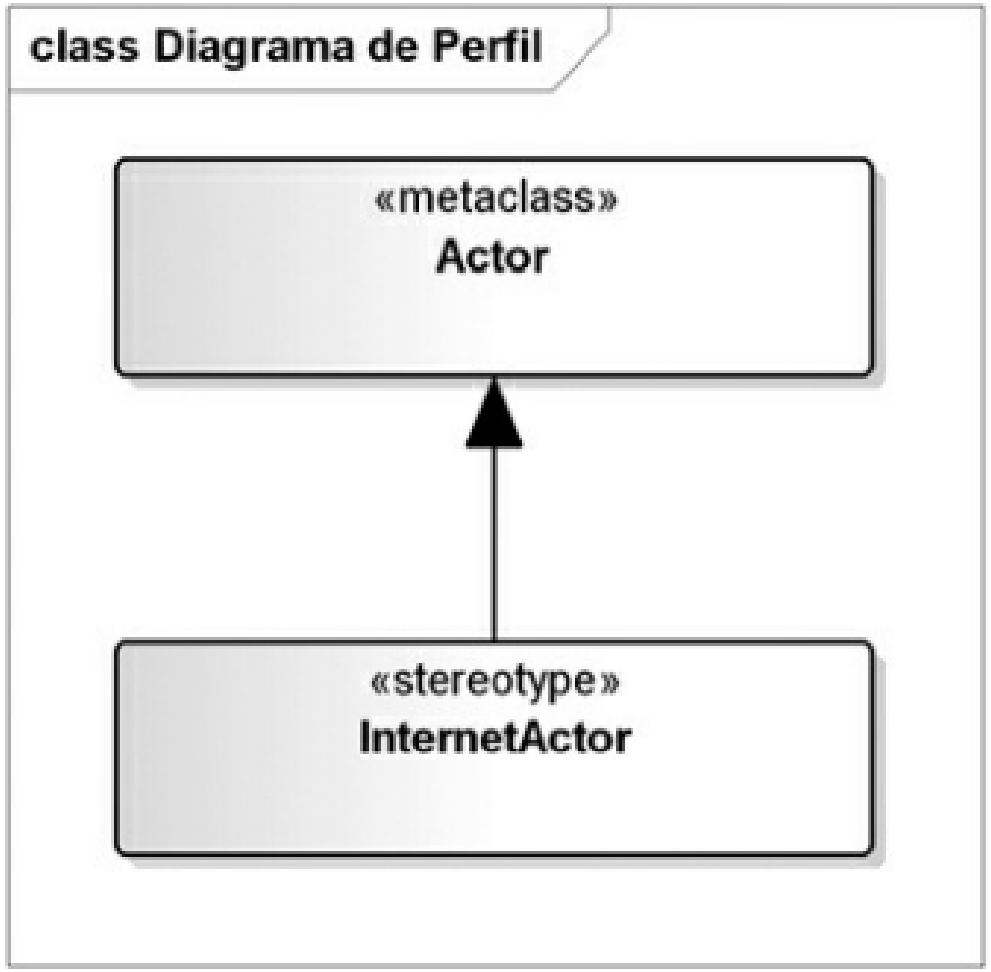


Figura 1.14 – Exemplo de Diagrama de Perfil.

- Abstrato.
- Cria-se modelagem de novos domínios.
- Adapta algo que o UML não consegue modelar.

Síntese Geral dos Diagramas

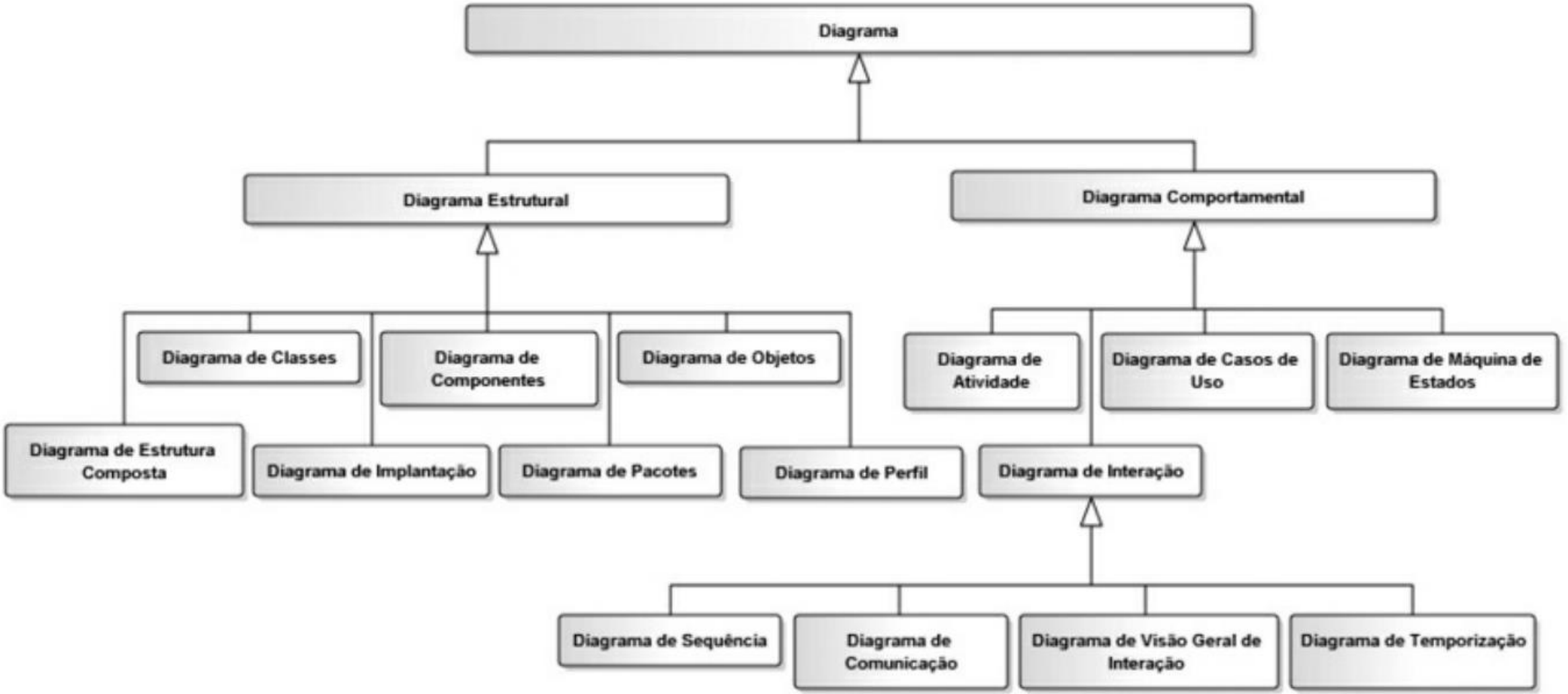


Figura 1.15 – Diagramas da UML.