PROJETO PROCEDIMENTAL

Projeto de Programas – PPR0001

Introdução

- Primeiro estudamos e determinamos a estrutura estática
- Agora vamos completar a trípode de modelagem:
  - Modelo de Objetos: especifica a estrutura dos objetos
  - Modelo Funcional: especifica o que acontece aos objetos
  - Modelo Dinâmico: especifica quando acontecem eventos aos objetos

- O modelo de objetos é importante para qualquer sistema que envolva dados não triviais
- Os modelos dinâmico e funcional são mais importantes para um tipo específico de sistema computacional:
  - Em sistemas não-interativos, e.g. compiladores, o modelo dinâmico é trivial, ao passo que o modelo funcional é fundamental
  - Em um banco de dados geralmente o modelo funcional é trivial pois o propósito é armazenar e organizar dados e não transformá-los

Modelagem Dinâmica

- Inicialmente vamos estudar e escrever o relacionamento entre as entidades em relação ao tempo (modelo dinâmico)
  - Diagrama de Eventos
  - Diagrama de Estados

  - Descreve sequência de operações que ocorrem em resposta a um evento
  - Descreve os estados em que um objeto pode estar
  - Não há preocupação em como as operações são implementadas

- Devemos escrever apenas os diagrama de acordo com as necessidades e a complexidade do sistema

Evento e Estados

- Estado: representa um estado do sistema; um conjunto característico de valores dos atributos

- Evento: estímulo individual de um objeto para outro
  - Eventos relacionados de forma causal (e.g. voo: saída → chegada)
  - Eventos não relacionados ou concorrentes (e.g. dois voos distintos)

Diagrama de Eventos

- Representam cenários: uma sequência de eventos que ocorrem durante uma determinada execução do sistema
  - A abrangência do cenário pode variar
- Eventos podem transmitir informações de um objeto para outro
- Representamos o tempo de cima para baixo
- Distâncias são irrelevantes
- Foco na sequência de eventos
- Podemos usar a ferramenta ASTAH para gerar tal diagrama (usar diagrama de sequencia com mensagens assíncronas)
  - ASTAH Community é a versão gratuita da ferramenta

Diagrama de Eventos

Exemplo – Chamada Telefônica

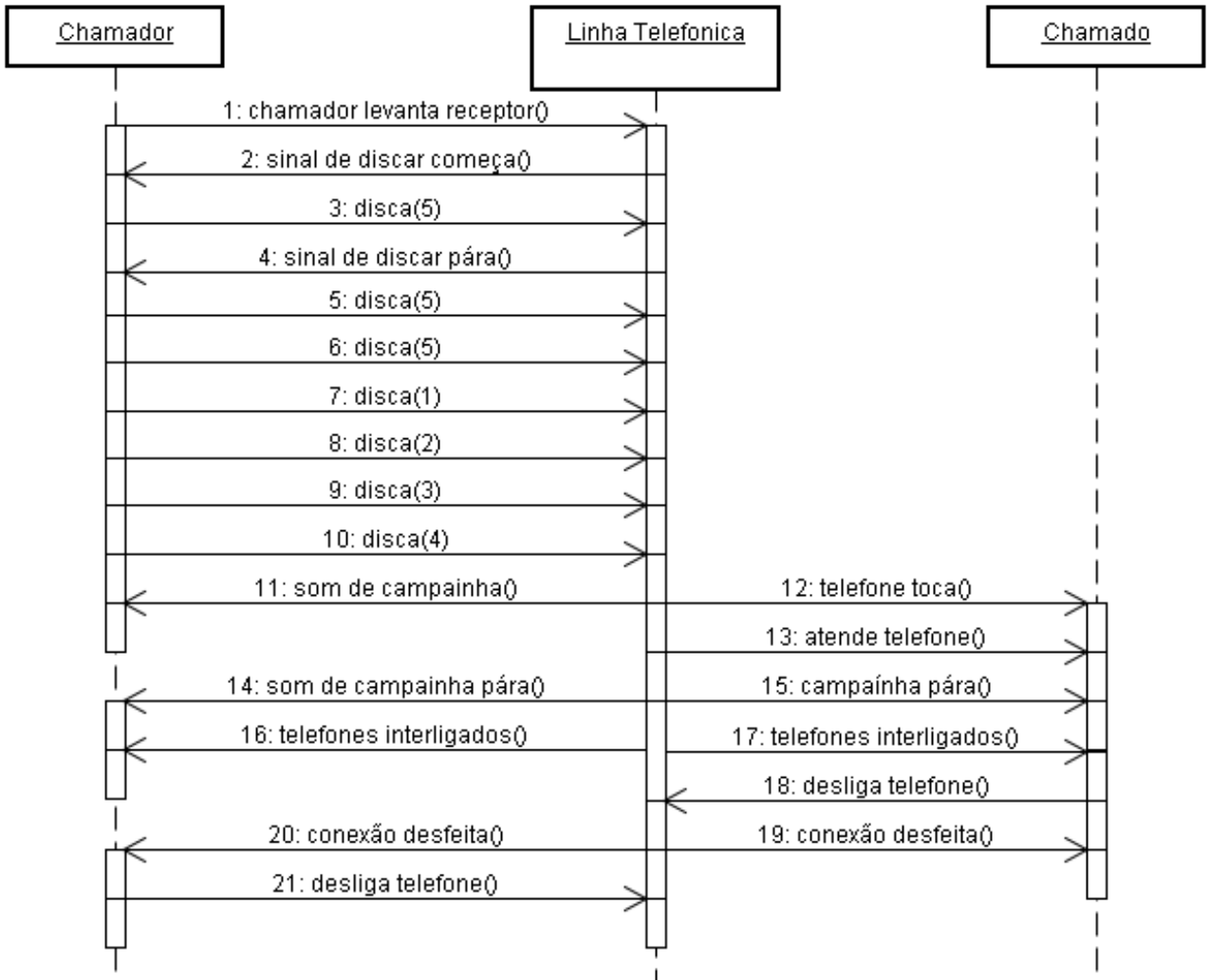


Diagrama de Estados

- É um diagrama que representa os possíveis estados dos objetos e os eventos que levam de um estado a outro

- Um estado pode ser duradouro (intervalo de tempo)
  - e.g. viajar de uma cidade a outra, alarme soando, ou inatividade (telefone).
- Um estado está relacionado aos valores de objetos
  - e.g água líquida → temperatura maior que 0°C e menor que 100°C;
    transmissão de um automóvel → ré || ponto morto || primeira || ...;

- Deve-se concentrar nas qualidades (das entidades) que são relevantes ao sistema e abstrair as irrelevantes

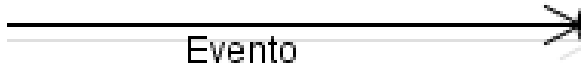
Diagrama de Estados

- Diagramas de Estados podem representar:
  - Ciclos de vida (e.g. jogo de xadrez)
    Marcados por um início e um fim
  - Laços contínuos (e.g. linha telefônica)
    Estado inicial e final não existe, ou não se conhece, ou não é de interesse para a representação do sistema.
- Podemos utilizar o JUDE para desenhar estes diagramas também:
  - Diagrama de Máquina de Estados

Diagrama de Estados - Representação

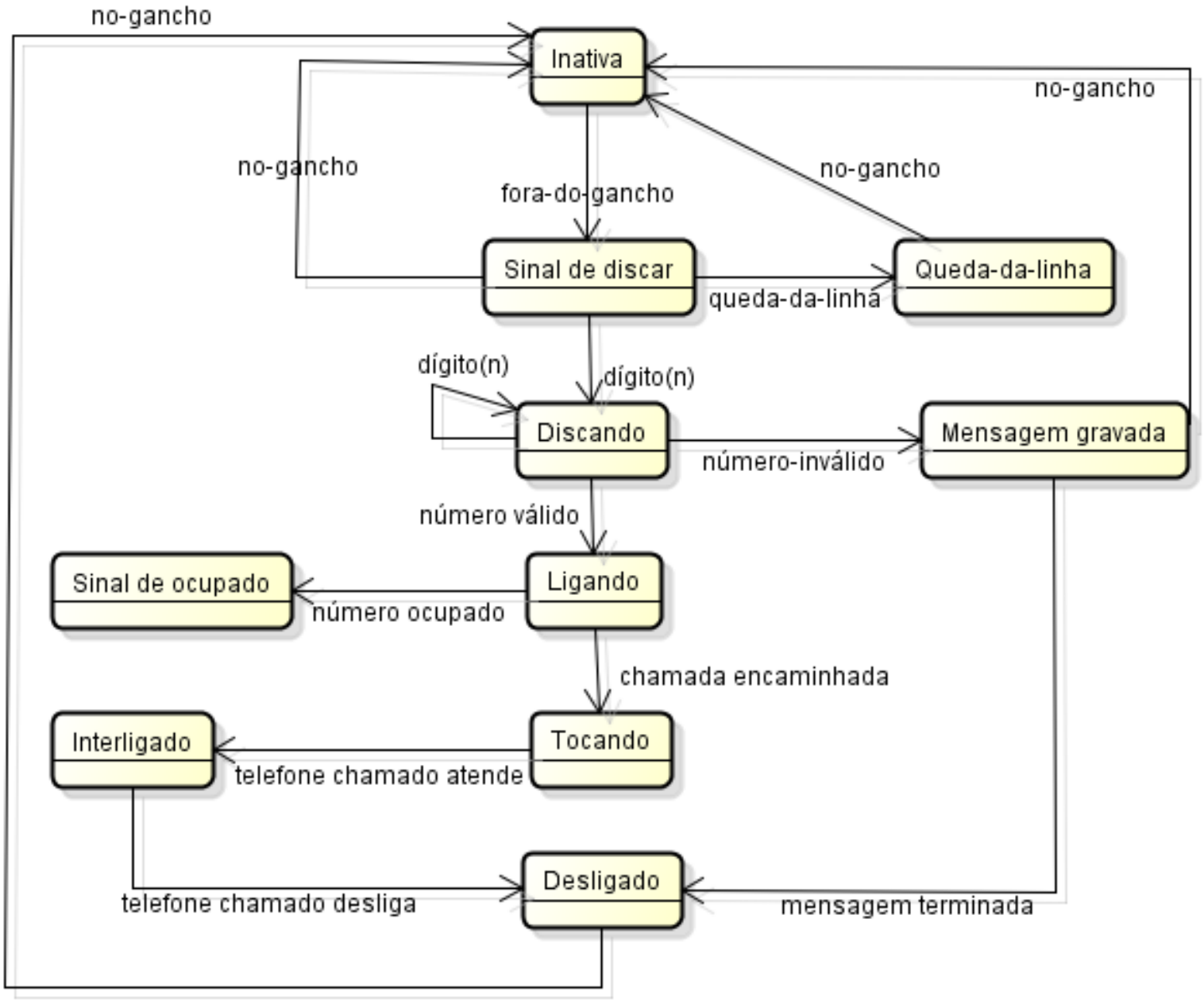
- Estado:
  - Representado por um retângulo arredondado com nome que o identifica;
  - Cada estado pode ter um ou mais eventos associados a ele;

- Transição:
  - Passagem de um estado para outro por meio de um evento;
  - Representada por uma seta direcionada e rotulada.



- Estado Inicial:
  - Indica um inicio e pode representar a criação de um objeto
  - Representado por um círculo cheio, pode ser rotulado.

- Estado Final:
  - Indica uma possível terminação e pode representar a destruição de um objeto
  - Representado por um “olho-de-boi”, pode ser rotulado.


no-gancho
Inativa
no-gancho
no-gancho
no-gancho
fora-do-gancho
Sinal de discar
Queda-da-linha
queda-da-linha
dígito(n)
dígito(n)
Discando
Mensagem gravada
número-inválido
número válido
Sinal de ocupado
Ligando
número ocupado
chamada encaminhada
Interligado
Tocando
telefone chamado atende
Desligado
telefone chamado desliga
mensagem terminada

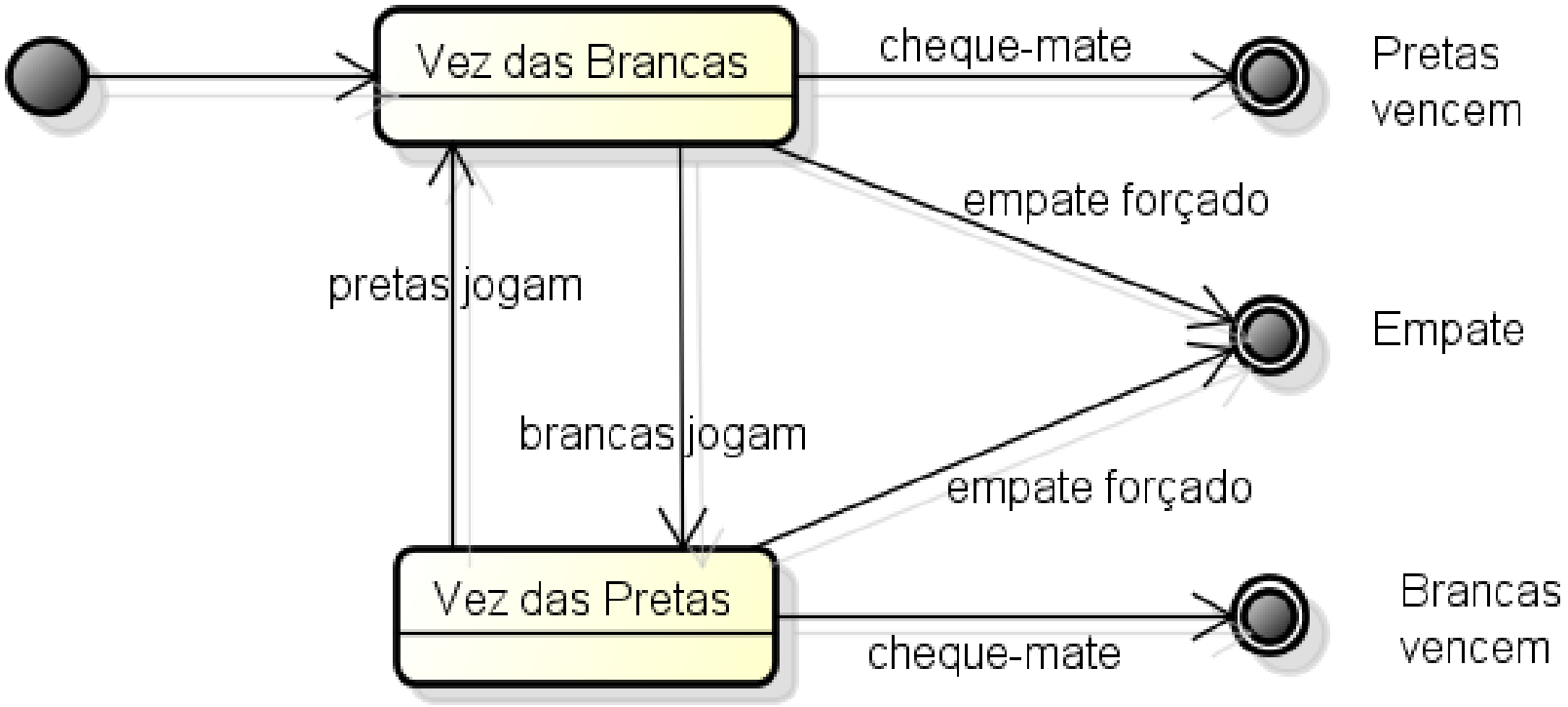

Vez das Brancas
cheque-mate
Pretas
vencem
empate forçado
pretas jogam
Empate
brancas jogam
empate forçado
Vez das Pretas
Brancas
vencem
cheque-mate

Diagrama de Estados

- Condições:
  - Podem ser utilizadas como guardas nas transições, garantindo que uma transição só ocorre sob certa condição;
  - Representadas de forma textual entre colchetes;
  - No JUDE utilize o campo “guard” das setas de transição;

  (ver próximo slide)

- Operações:
  - Descrição comportamental que especifica o que um objeto faz em determinado estado ou em resposta a eventos;
  - Representação textual após uma barra (/) em transições;
  - Representação textual após a expressão “faça /” (ou “*do /*”) em estados;

Diagrama de Estados – Condições

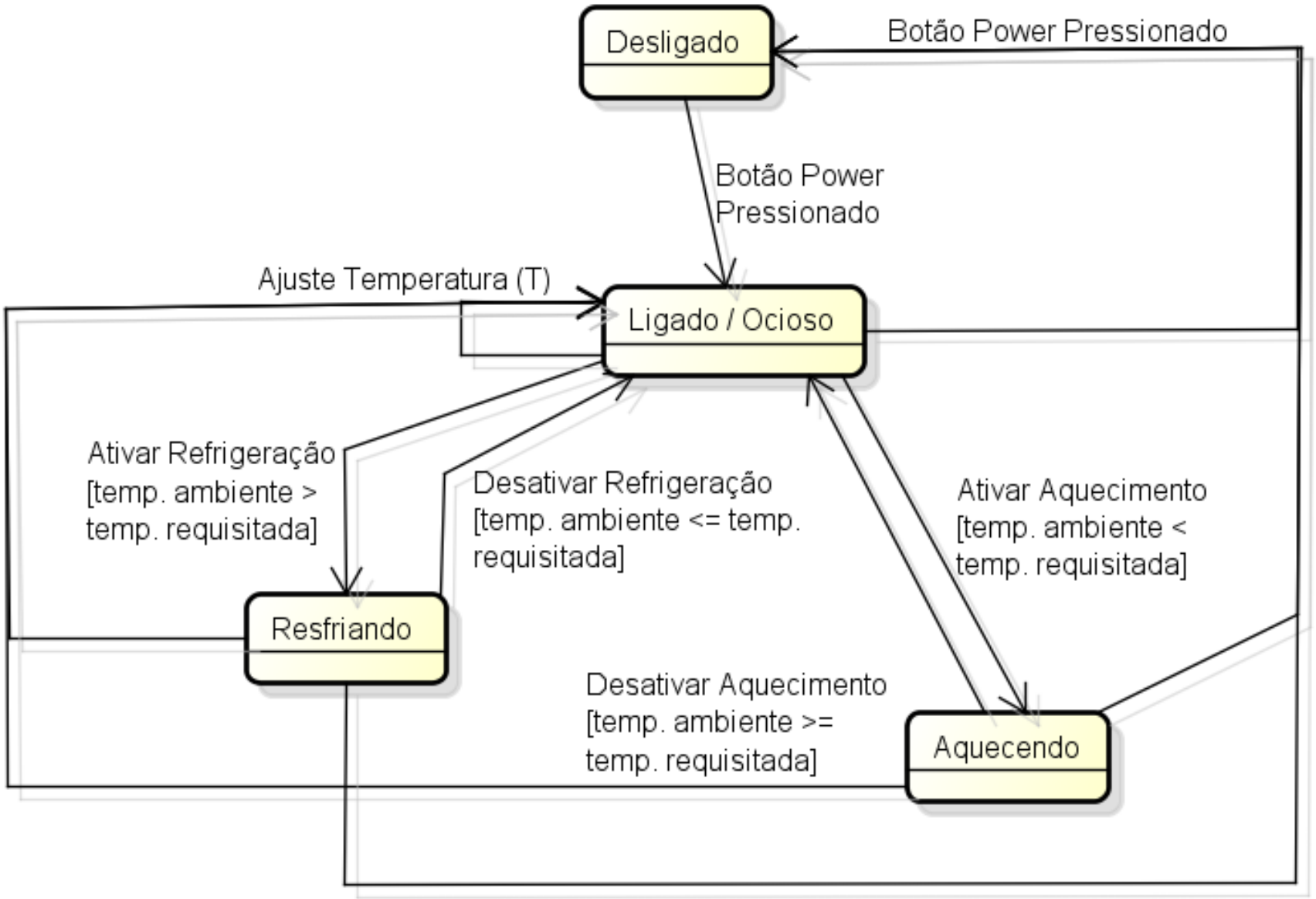


Diagrama de Estados – Operações

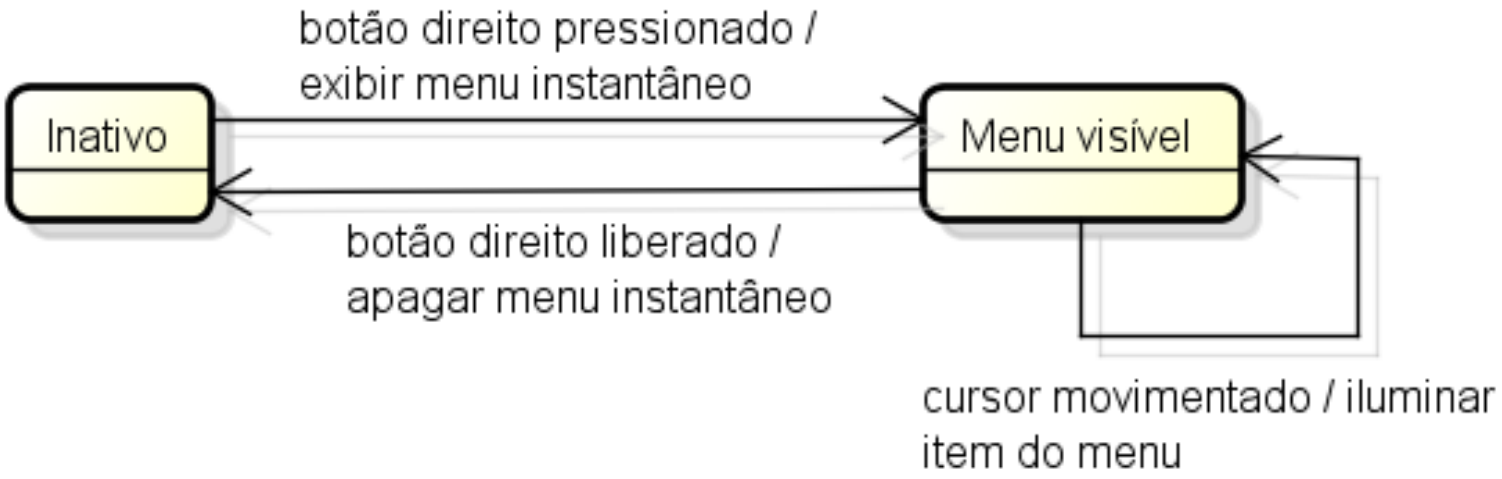


Diagrama de Estados – Operações

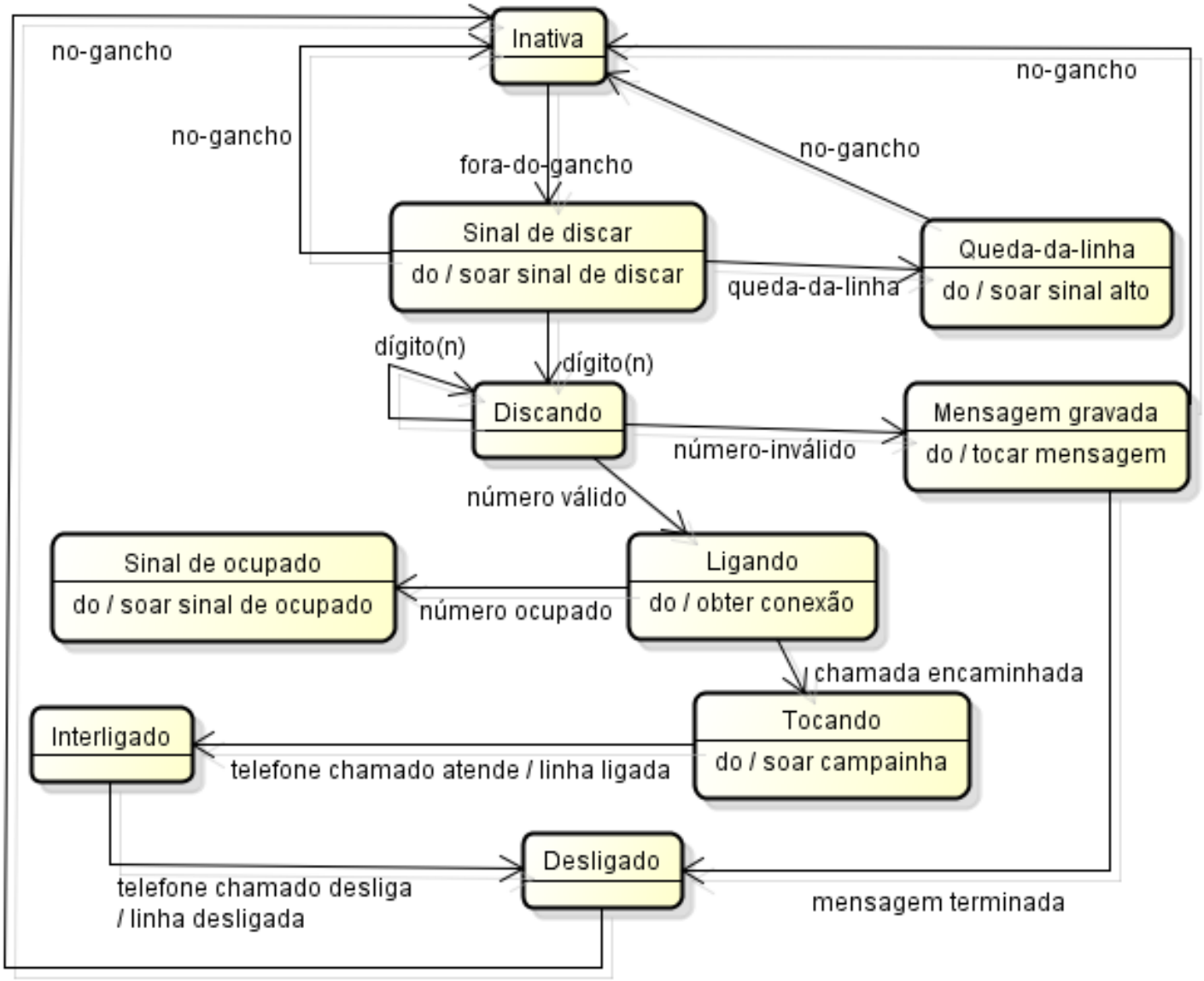


Diagrama de Estados Multinivelados

- Sistemas complexos podem requerer diagramas com muitos estados e eventos → falta de legibilidade

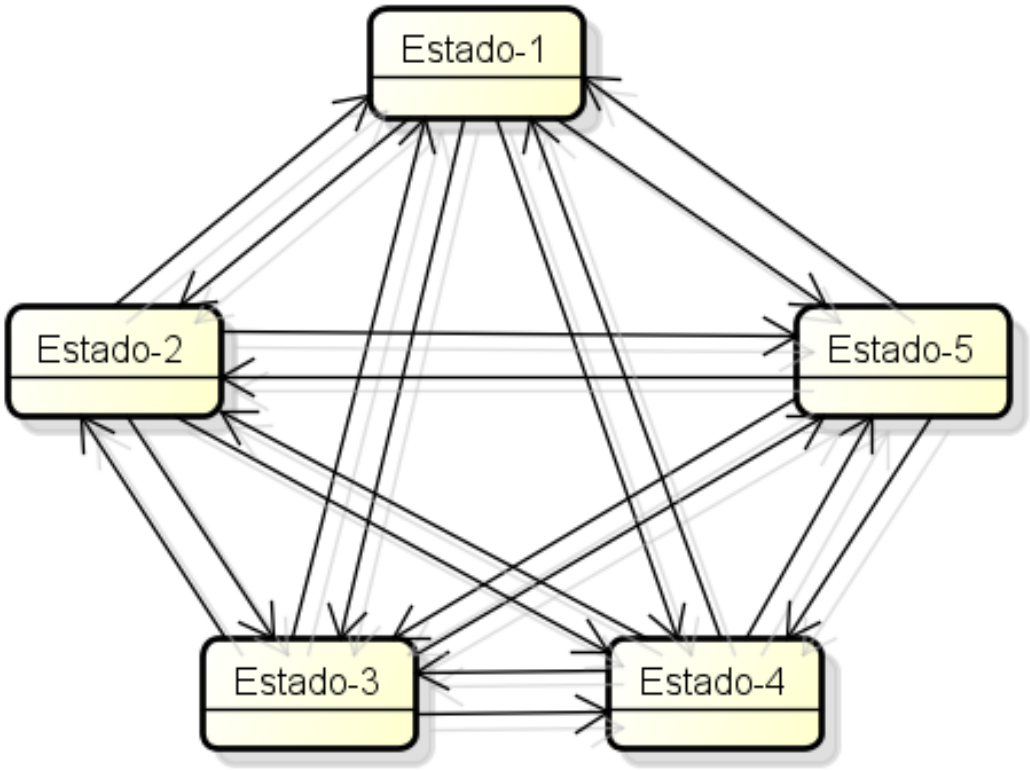
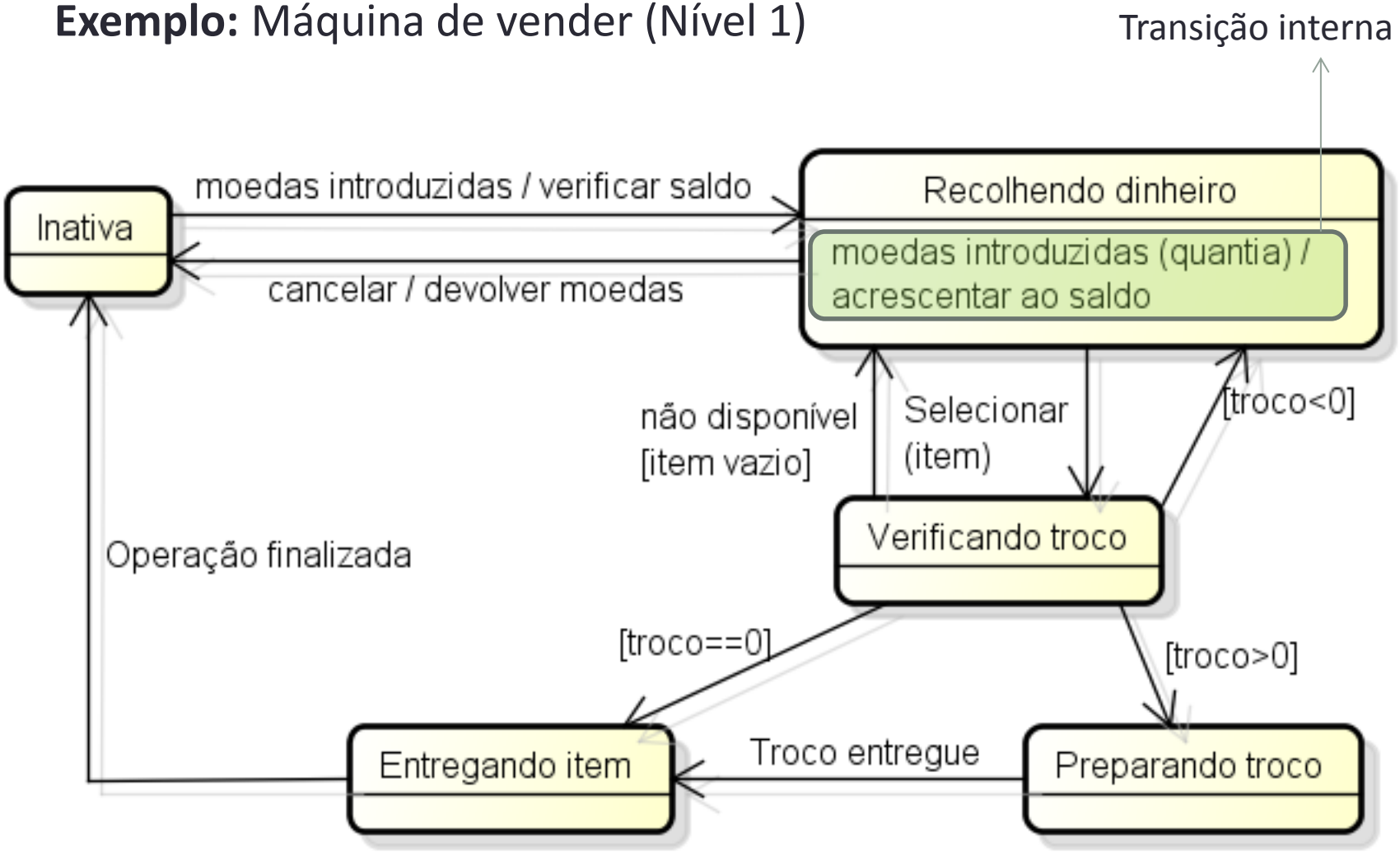


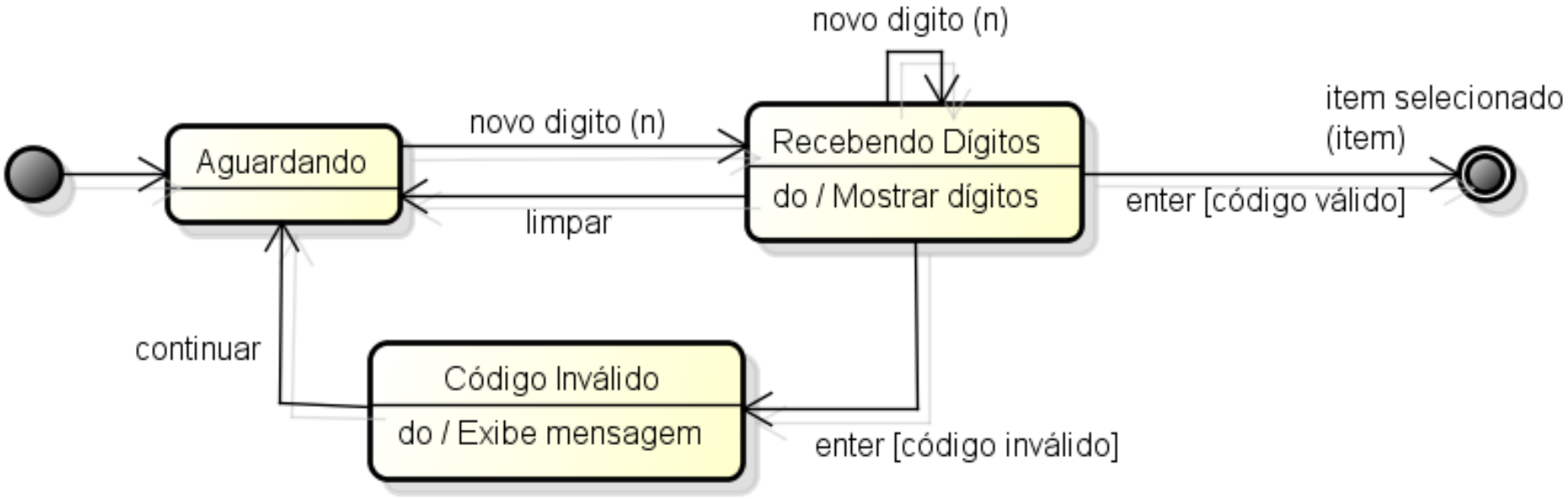
Diagrama de Estados Multinivelados

- Para manter a qualidade de expressão e melhorar a legibilidade os diagramas são desenhados em vários níveis de detalhamento
  - A quantidade de níveis pode variar de sistema para sistema;
  - A relação entre os diagramas deve ficar clara;
  - Vários diagramas podem fazer parte de um mesmo nível de detalhamento;

Exemplo: Máquina de vender (Nível 1)



Exemplo: Máquina de vender (Nível 2)
Selecionando item



Exemplo: Máquina de vender (Nível 2)
Entregando item

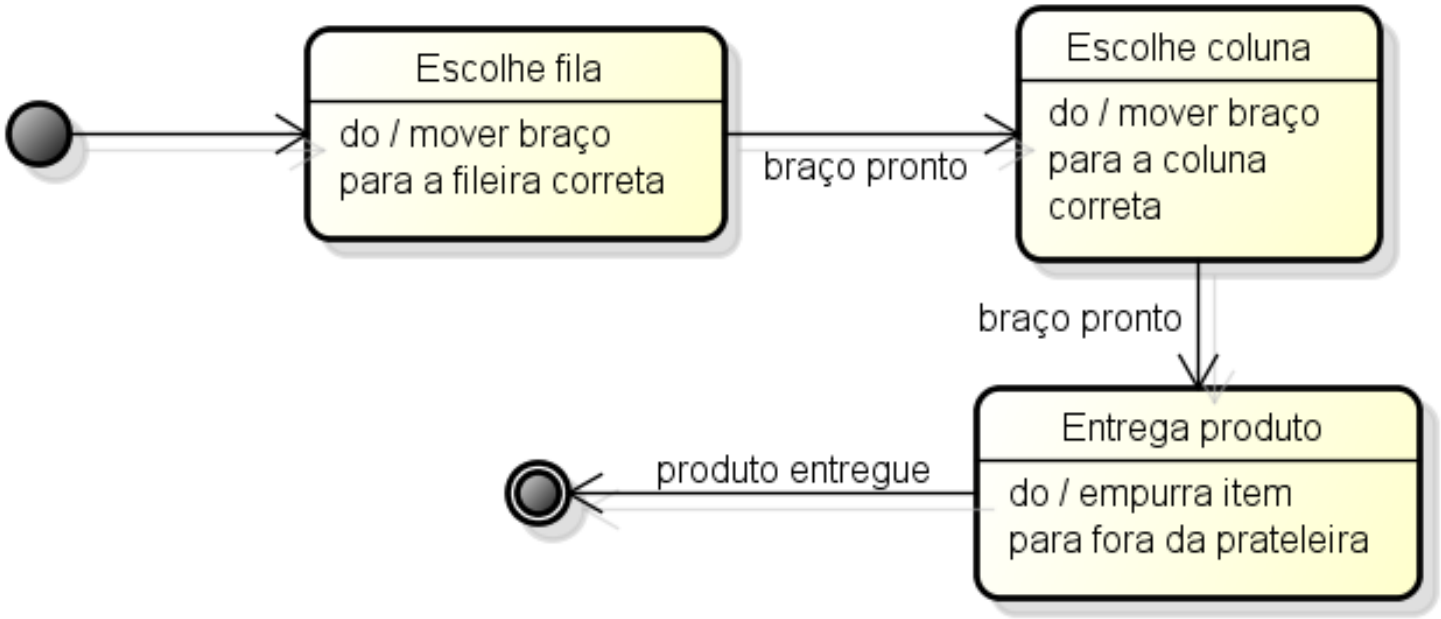


Diagrama de Estados

- Ações de Entrada:
  - Descrição de operações que devem ser realizadas assim que o fluxo alcança um dado estado;
  - Representação textual após a expressão “entrada /” (ou “*entry /*”)
- Ações de Saída:
  - Descrição de operações que devem ser realizadas assim que o fluxo for sair de um dado estado para outro;
  - Representação textual após a expressão “saída /” (ou “*exit /*”);
- Ordem execução:
  - Ações da transição que chega // Ação de entrada (*entry*) // Operações (*do*) // Ação de saída (*exit*) // ações da transição de saída

Diagrama de Estados

- Ações Internas:
  - Descrição de transição que ocorre sem causar uma mudança de estado;
  - Representação textual que aparece na parte inferior de um estado;
  - Parecido com uma auto transição, mas não igual!

Qual a diferença?

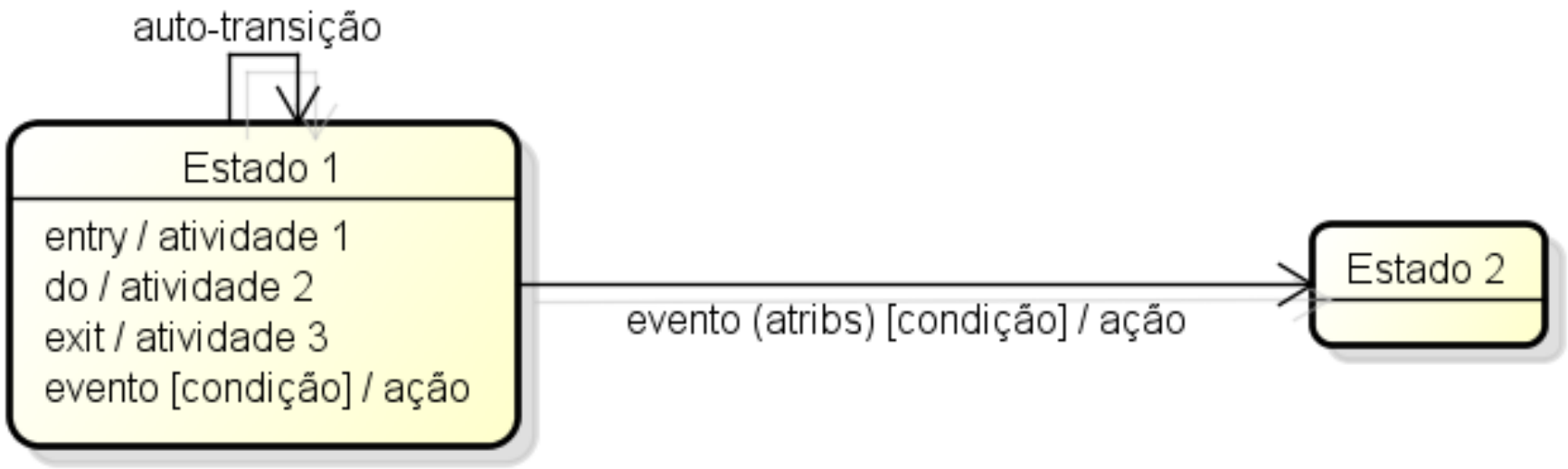
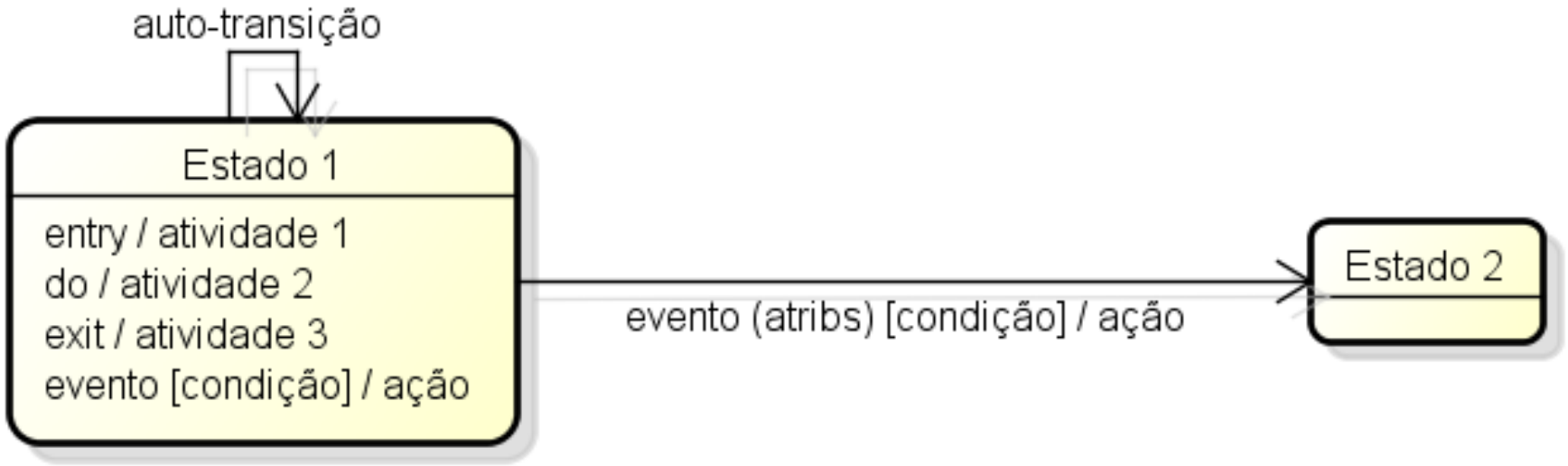


Diagrama de Estados

- Ações Internas:
  - Descrição de transição que ocorre sem causar uma mudança de estado;
  - Representação textual que aparece na parte inferior de um estado;
  - Parecido com uma auto transição, mas não igual!
    - Em ações internas as operações de entrada e saída não são realizadas
    - Em auto transições as operações de entrada e saída são realizadas



Modelagem Funcional

- O modelo funcional especifica como os valores de saída de um processamento se transformam em valores de entrada
- Para a representação, utilizamos múltiplos Diagramas de Fluxo de Dados (DFD)

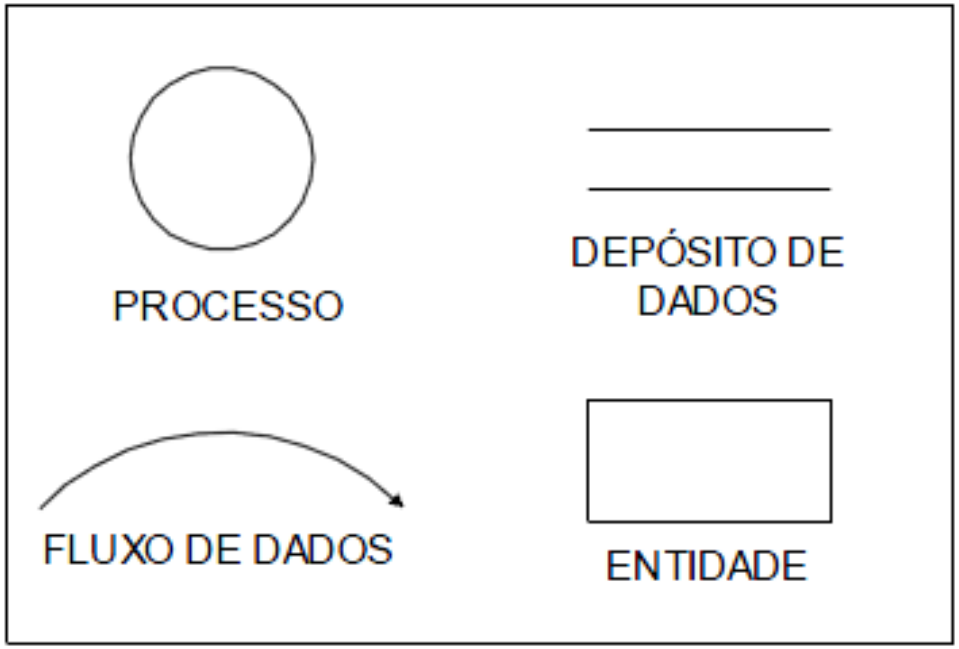
- DFD é um gráfico que mostra o fluxo dos valores de dados desde suas origens nos objetos, através dos processos que os transformam, até seus destinos em outros objetos.

Diagrama de Fluxo de Dados

- Um DFD pode ser visto como uma rede que ilustra a circulação dos dados no interior do sistema;

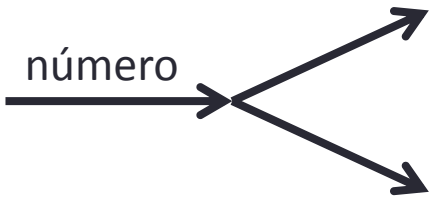
- Símbolos utilizados:

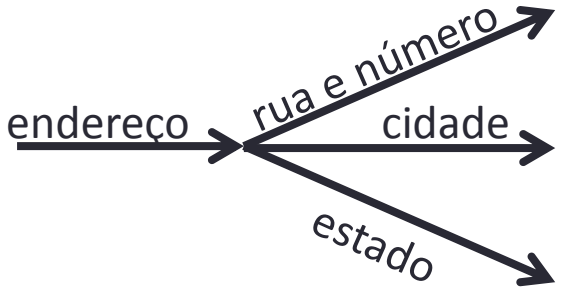
- Podemos criar DFDs apenas na versão paga do JUDE ☹



Fluxo de dados

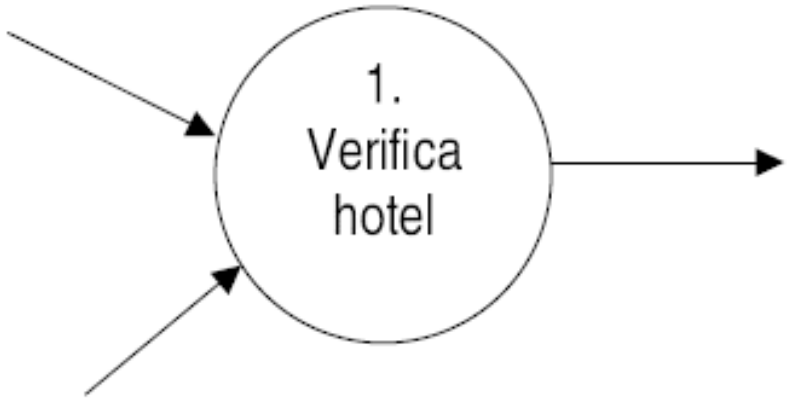
- Representado por setas direcionadas, indicam o fluxo de um determinado conjunto de dados.
- Pode-se representar a cópia ou a subdivisão dos componentes de um dado através de um “garfo”





Processos (bolhas ou bolas)

- Transformam fluxos de dados: entrada → saída
- São identificados com um nome (e opcionalmente um número)
- Os fluxos de dados envolvidos indicam os caminhos possíveis

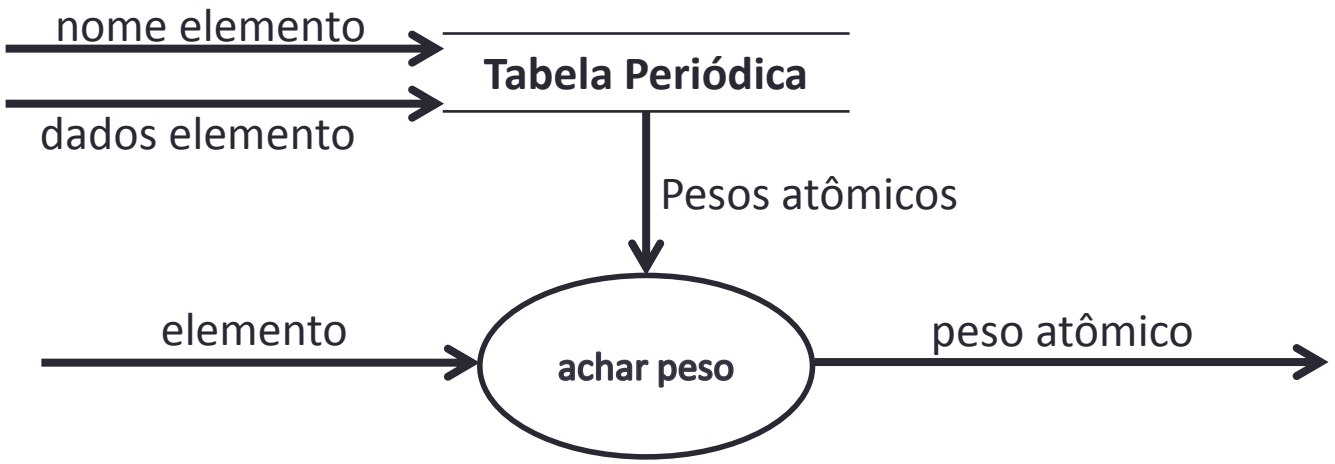


Depósito de dados (arquivo)

- São “reservatórios” para os dados existentes no sistema
  - Variável em memória, arquivos, banco de dados
- Representados por duas linhas horizontais paralelas com o nome do depósito (nome único) no meio
- Setas direcionadas a um depósito indicam a inclusão ou modificação de dados
- Setas que saem de um depósito indicam a consulta/recuperação de informação

Depósito de dados (arquivo)

- Exemplo:

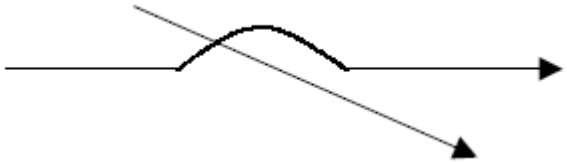


Entidades exteriores - atores

- Elementos que fornecem as entradas ao sistema
  - Fontes
- Elementos que recebem saídas do sistema
  - Terminadores, terminais
- Representados por retângulos
- Existem fora da fronteira do sistema (externos ao sistema)

Convenções adicionais

- Deve-se **minimizar** o cruzamento de fluxos
  - Se for inevitável utilizar a notação:



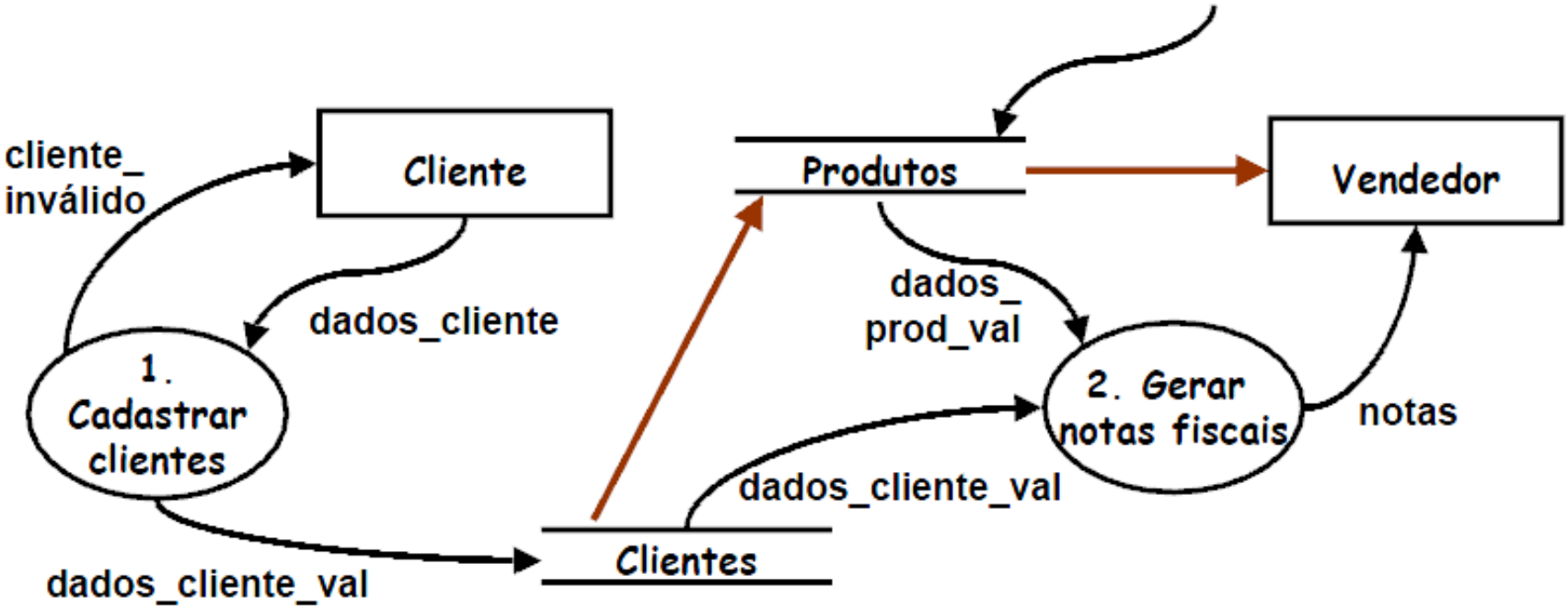
- Repositórios e Atores podem ser desenhados mais de uma vez
  - Continuam sendo o mesmo item (usar nome idêntico)

Elementos

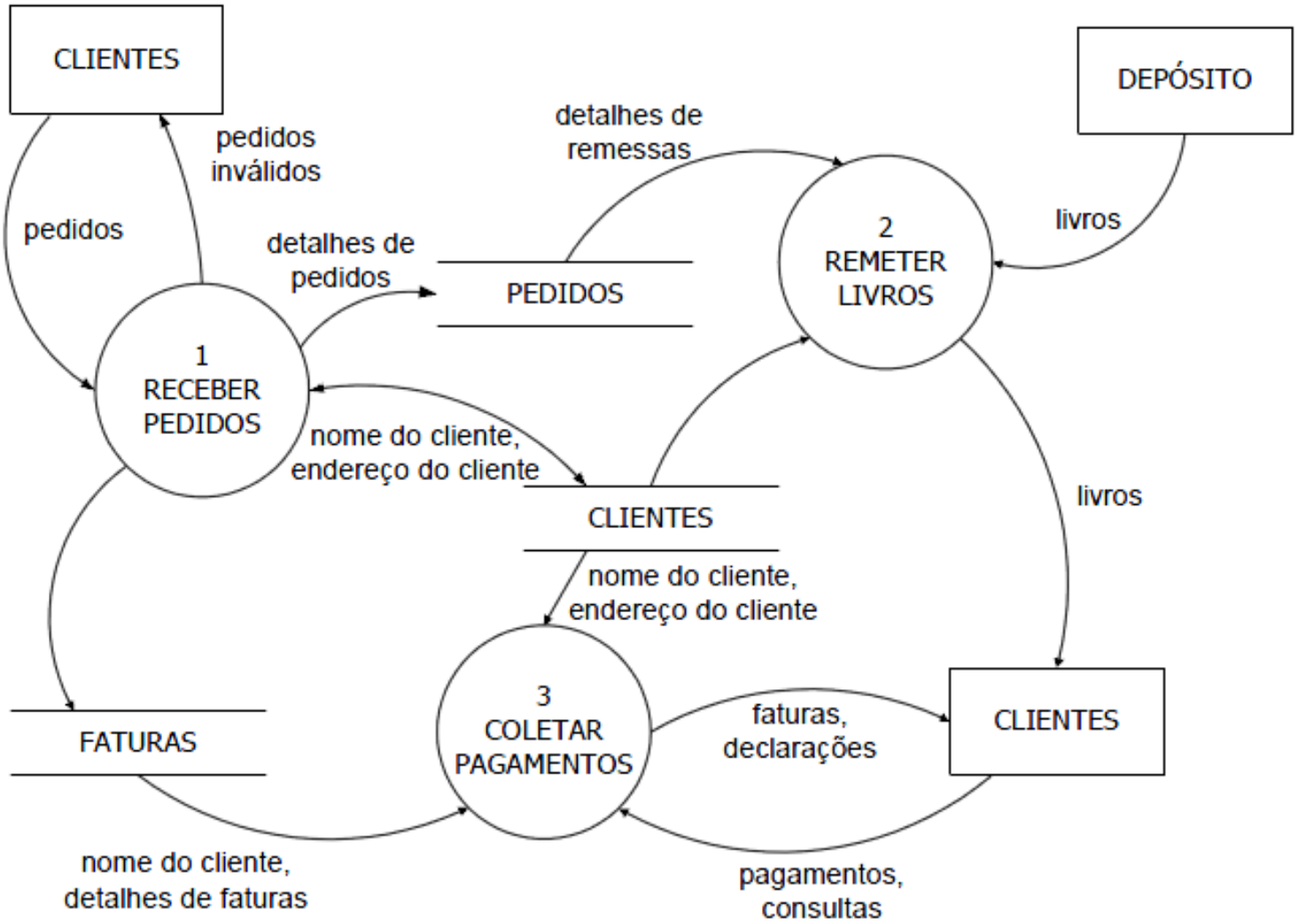
- Sistemas de reservas em um Hotel:
  - Um cliente efetua uma reserva num hotel através de uma agência;
  - Verifica-se a disponibilidade do hotel escolhido e é atribuído um quarto;
  - Calcula-se a conta;
  - É emitido um voucher ao cliente e informado o hotel da reserva.

Exemplos

- Sistema de vendas



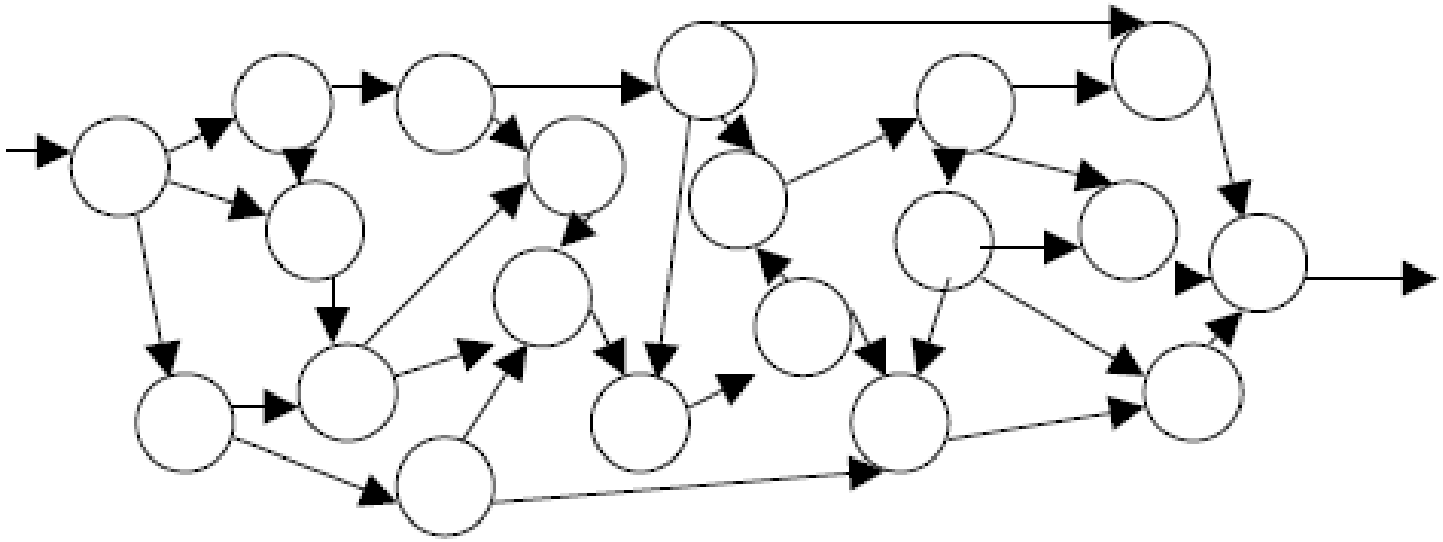
Exemplos


CLIENTES
DEPÓSITO
pedidos inválidos
detalhes de remessas
pedidos
detalhes de pedidos
PEDIDOS
2 REMETER LIVROS
livros
1 RECEBER PEDIDOS
nome do cliente, endereço do cliente
CLIENTES
livros
nome do cliente, endereço do cliente
FATURAS
3 COLETAR PAGAMENTOS
faturas, declarações
CLIENTES
nome do cliente, detalhes de faturas
pagamentos, consultas

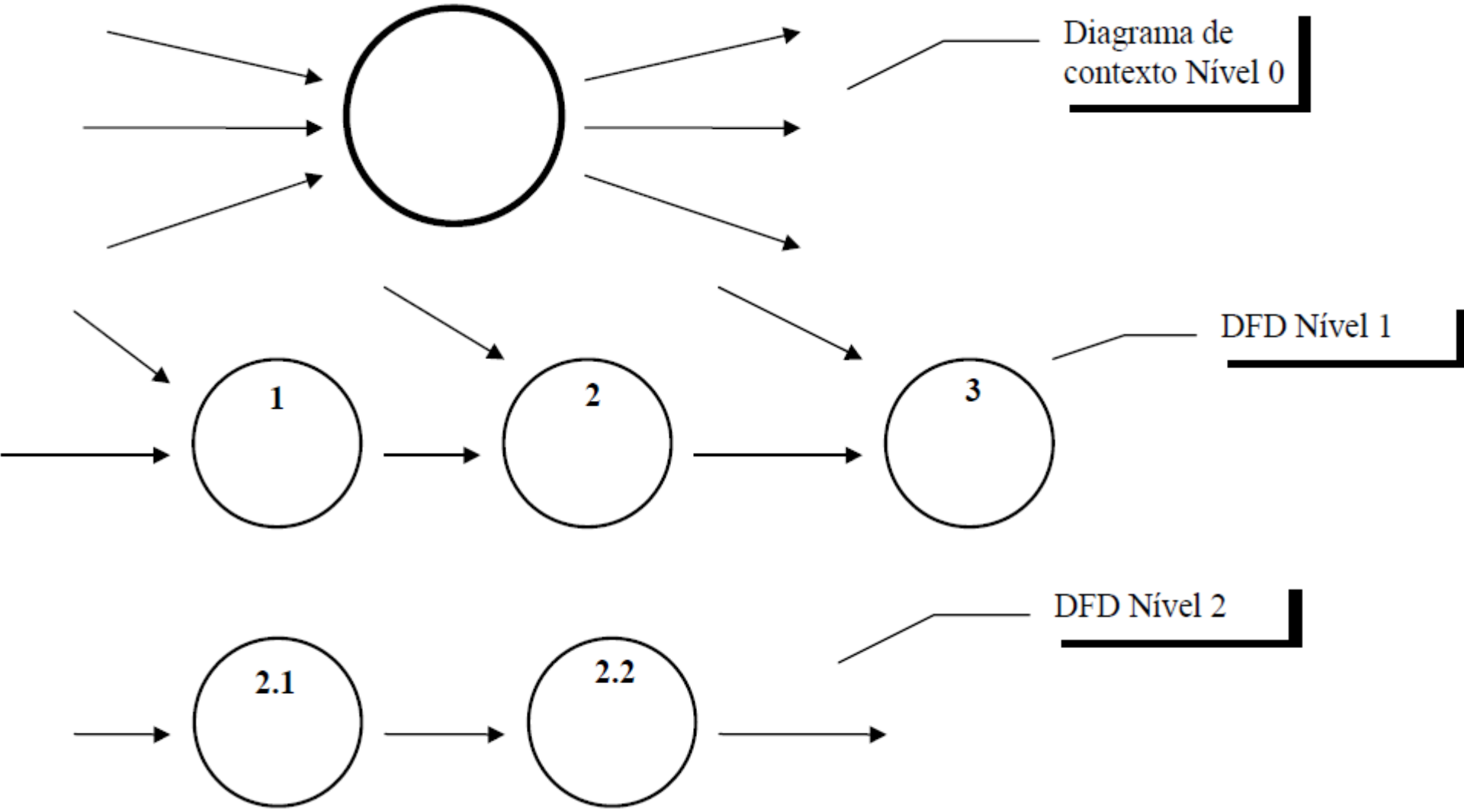
Decomposição

- Um DFD de um sistema pequeno é fácil de construir e é facilmente interpretado e entendido.

- Quanto mais complexo for o sistema modelado mais complexo o DFD poderá se tornar.



Decomposição de DFDs - Níveis



Convenções de decomposição

- Cada processo em um nível de DFD pode ser expandido para se tornar um novo DFD
- Cada processo de um nível inferior está relacionado com o nível superior e é identificado por um número composto (ex.: 2.1.3)
- Todos os fluxos de dados que entram e saem do nível superior devem aparecer no nível inferior (validação vertical)

Convenções de decomposição

- Em cada DFD, o limite superior recomendado para o número de processos é de aproximadamente 7
  - Diminui a complexidade no entendimento
- Processos que não são expandidos são denominados processos primitivos ou primitivas funcionais
  - Somente processos muito simples não serão expandidos

Erros comuns

- Não são permitidos processos apenas com entradas;
- Processos só com saídas são suspeitos e a maior parte das vezes incorretos.
  - Uma exceção: gerador de números aleatórios;

Recomendações

- Processos com nomes ambíguos ou muito genéricos revelam falta de conhecimento sobre o sistema (ex. manipulação de entrada, gera saída);
- Fluxos de dados com nomes como “itens de entrada”, ou “vários dados” revelam um conhecimento pobre do sistema;
- Cruzamento de fluxos indicam que é potencialmente necessário decompor o DFD;
- Primitivas funcionais com grande número de entradas e saídas indicam a necessidade de decompor o DFD;

Regras e Heurísticas de projeto

1. Estabelecer o contexto do DFD indicando todas as entidades externas do sistema;

2. Identificar todas as saídas e entradas do sistema. Desenhar o diagrama de contexto (abstração geral);

3. Selecionar um ponto de partido para o projeto. Desenhar os fluxos que são necessários para ir de um ponto a outro:
   - De entrada para saída;
   - De saída para entrada;
   - Partindo do centro para as pontas.

Regras e Heurísticas de projeto

4. Identificar os fluxos de dados e depósitos de dados

5. Rotular todos os processos com verbos operacionais que relacionem as entradas com os fluxos de saída;

6. Omitir detalhes de programação como verificação de erros, inicializações e finalizações.

7. Evitar cruzar fluxos de dados
   - Se necessário utilizar entidades ou arquivos duplicados, ou a notação especial.

Regras e Heurísticas de projeto

8. Reavaliar a consistência do DFD. Se necessário, refazer
   - Validação vertical: os fluxos que entram e saem de um processo, devem entrar e sair do DFD resultante da explosão
   - Validação horizontal: Todo o que entra num processo é utilizado e tudo o que sai desse processo foi produzido.

10. Verificar, preferencialmente com o utilizador, se o DFD representa o sistema

11. Depois de estabelecido o DFD, explodir cada processo. Repetir a decomposição até obter o detalhe suficiente

Exemplo – Sistema de Hotelaria

Requisitos funcionais

1. O sistema deve permitir que o Cliente faça reserva de quarto(s) em determinado(s) período(s). Neste momento, é averiguado se existe quarto disponível no período solicitado. Caso positivo, é feita a reserva do quarto e enviada a confirmação para o Cliente; para isto, são necessários os seguintes itens de informação: nome do Cliente, telefone e tipo de quarto (solteiro, casal). Caso negativo, é informado ao Cliente a não disponibilidade do quarto;

2. O sistema deve permitir o cancelamento da reserva, disponibilizando o quarto, caso o Cliente solicite;

3. O sistema deve cancelar automaticamente a reserva, caso o Cliente não compareça no hotel para hospedar-se até às 12 horas do dia da reserva, disponibilizando o quarto;

Exemplo – Sistema de Hotelaria

4. O sistema deve permitir o registro do cliente ao ocupar um quarto, reservado previamente. Caso o quarto não esteja reservado, uma mensagem de rejeição será emitida. Caso contrário, a confirmação será fornecida ao Cliente;

5. O sistema deve permitir a emissão da conta ao Cliente e a disponibilização do quarto para limpeza, no momento em que ele solicitar a sua saída;

6. O sistema deve permitir o registro do pagamento da conta. Ao efetivar o pagamento é gerado um recibo para o cliente;

7. O sistema deve permitir a disponibilização do quarto, por parte do Gerente, quando este estiver limpo.

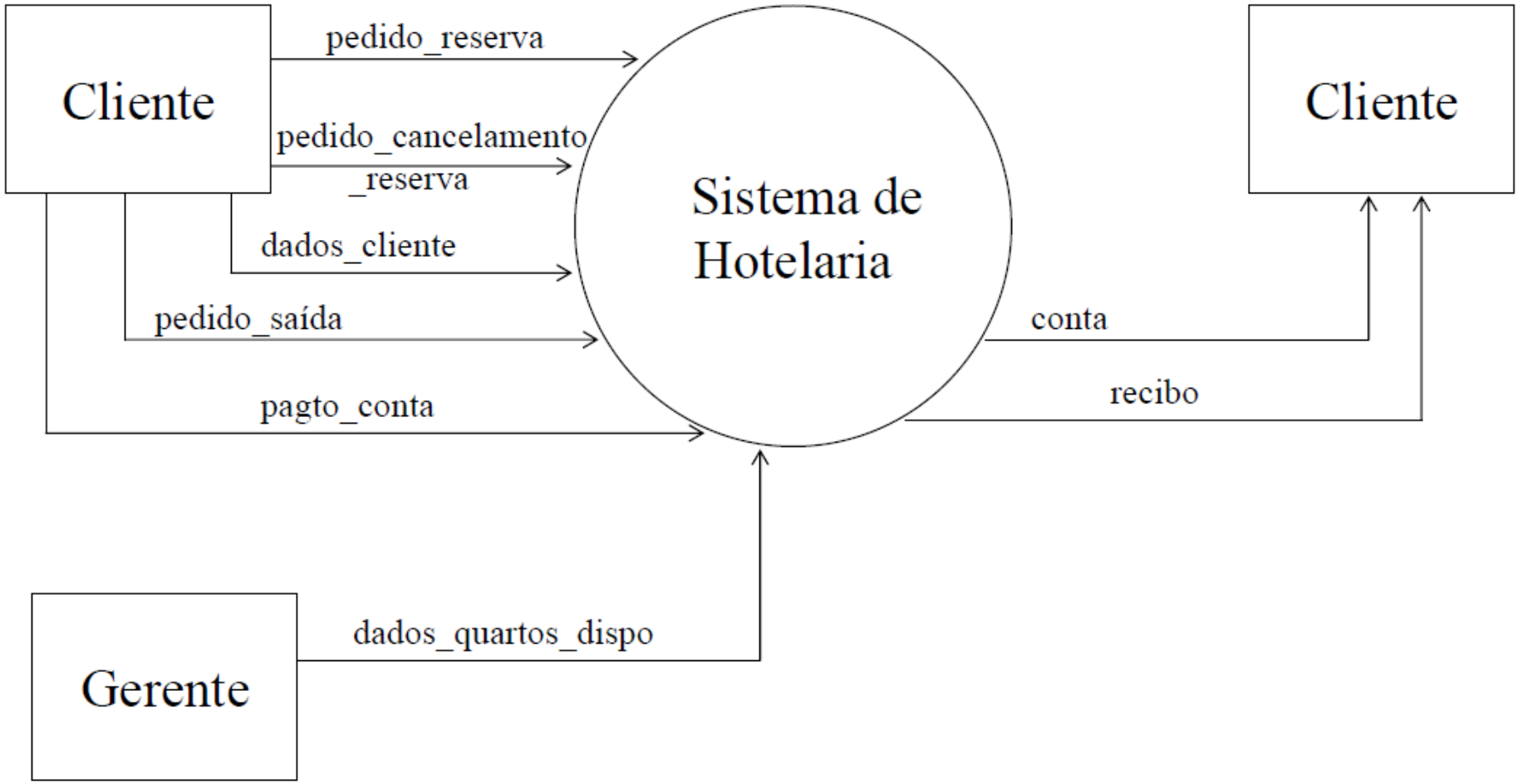
Exemplo – Sistema de Hotelaria

Eventos do Sistema

1. Cliente reserva quarto
2. Cliente cancela reserva
3. É hora de cancelar reserva
4. Cliente registra-se no hotel
5. Cliente solicita saída do hotel
6. Cliente paga a conta
7. Gerente disponibiliza o quarto

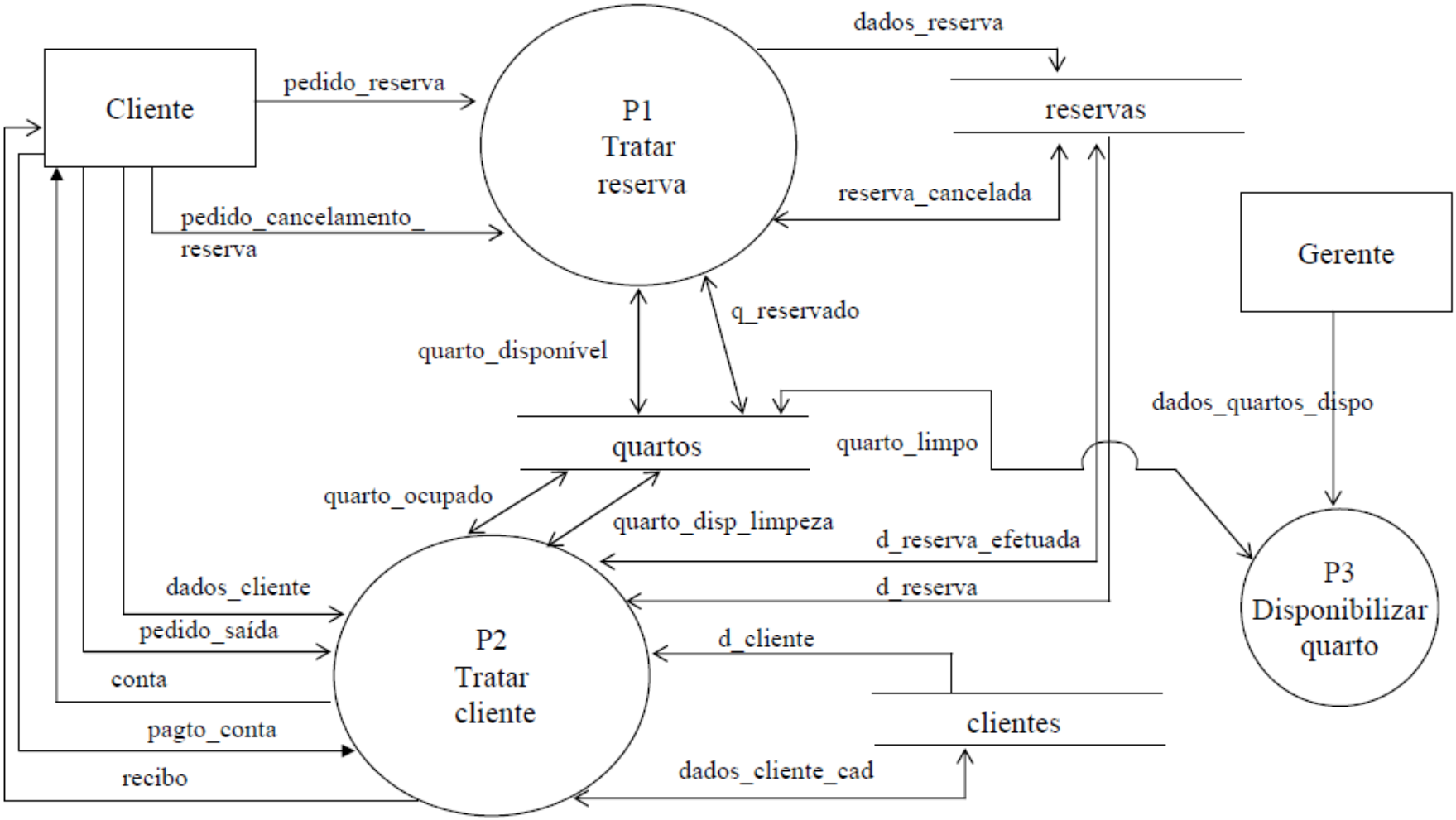
Diagrama de contexto

- Definir em uma abstração geral as entradas e saídas.



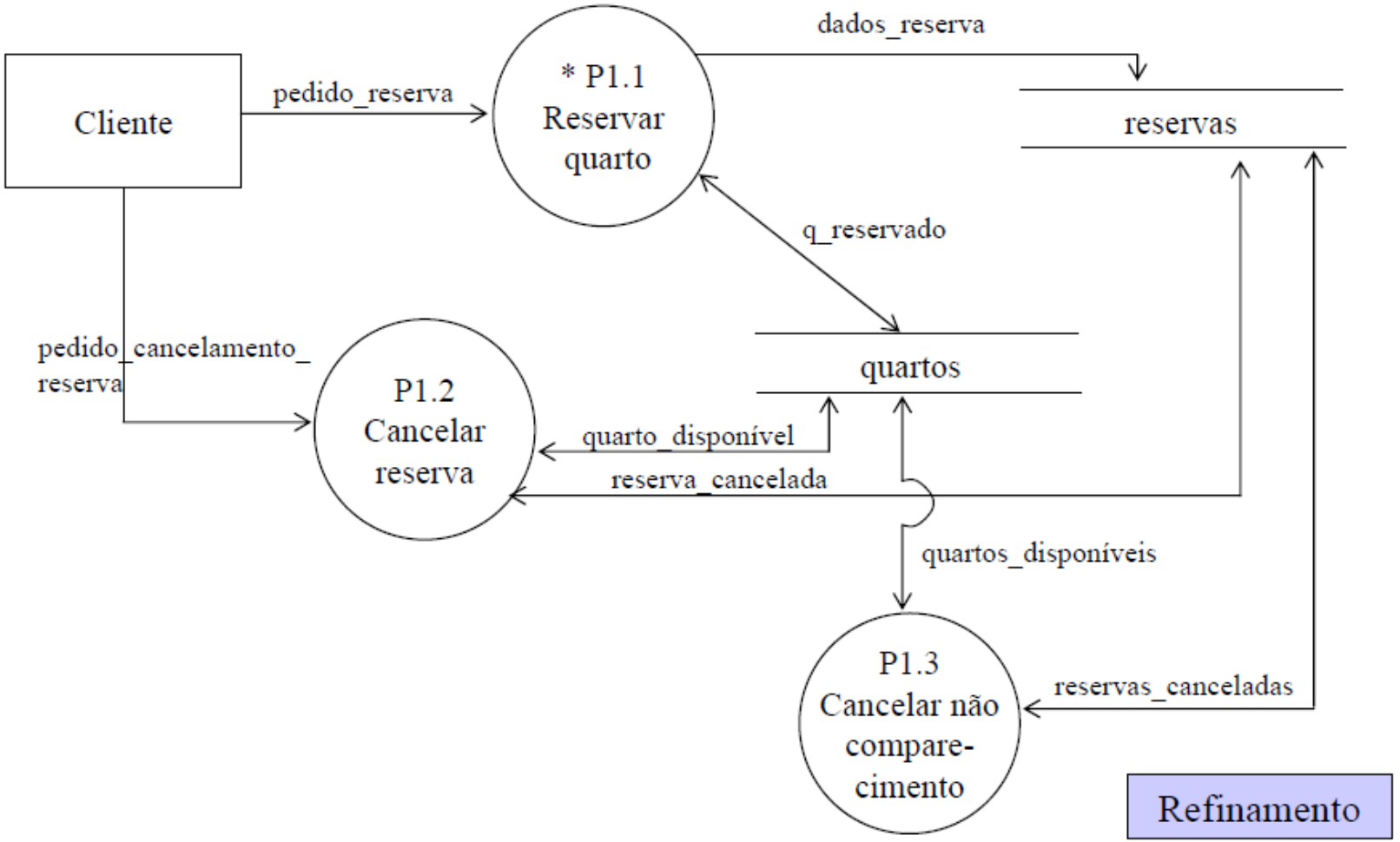
DFD – nível 0

- Visão mais específica das funcionalidades



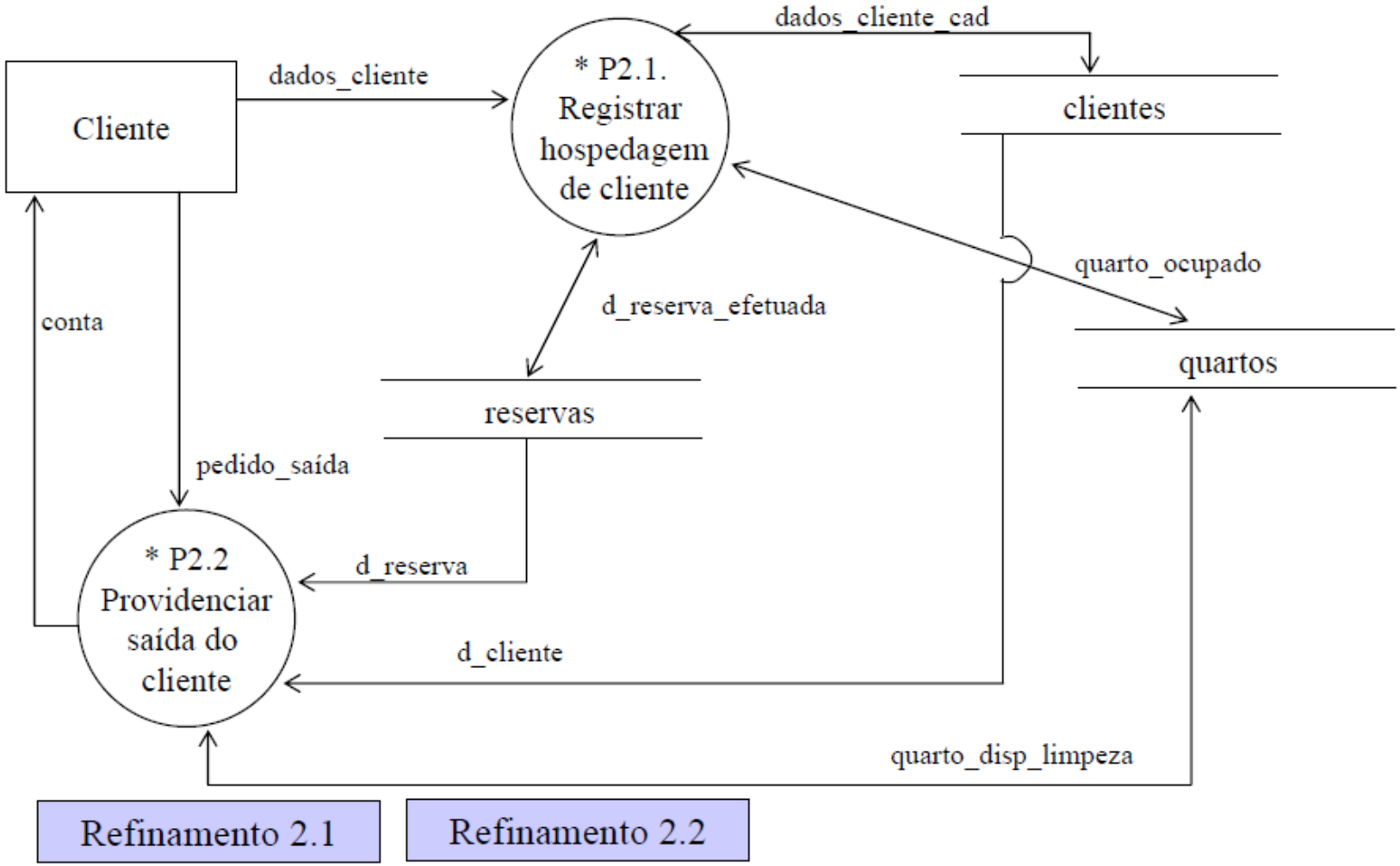
DFD – Nível 1

- Processo 1 detalhado



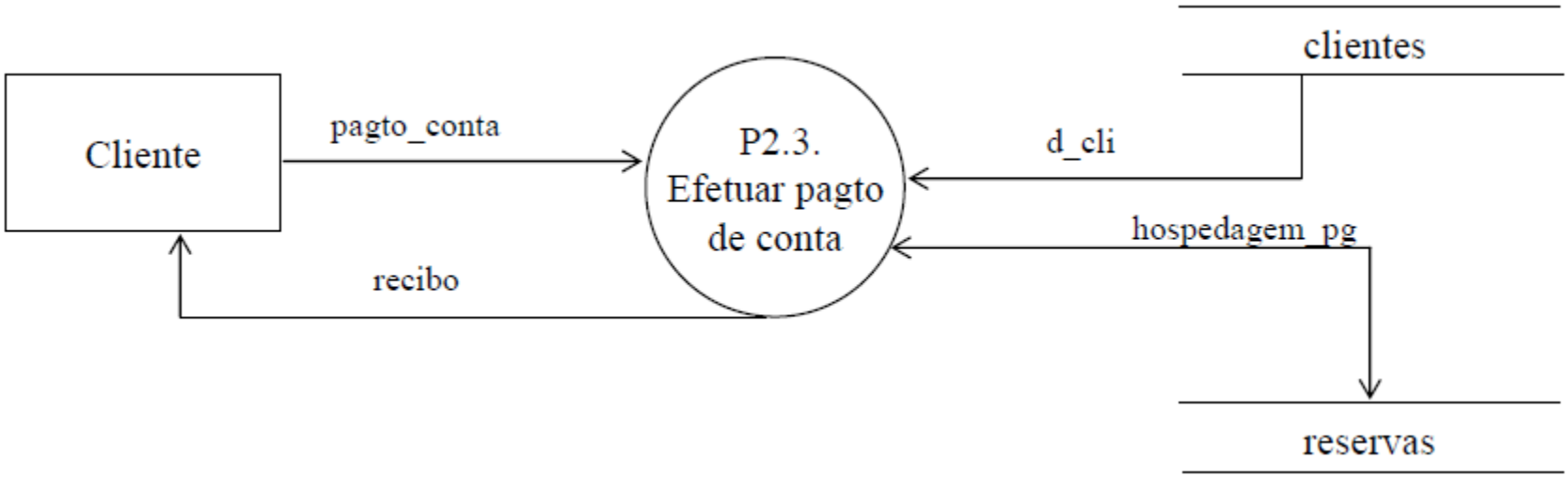
DFD – Nível 1

- Processo 2 detalhado

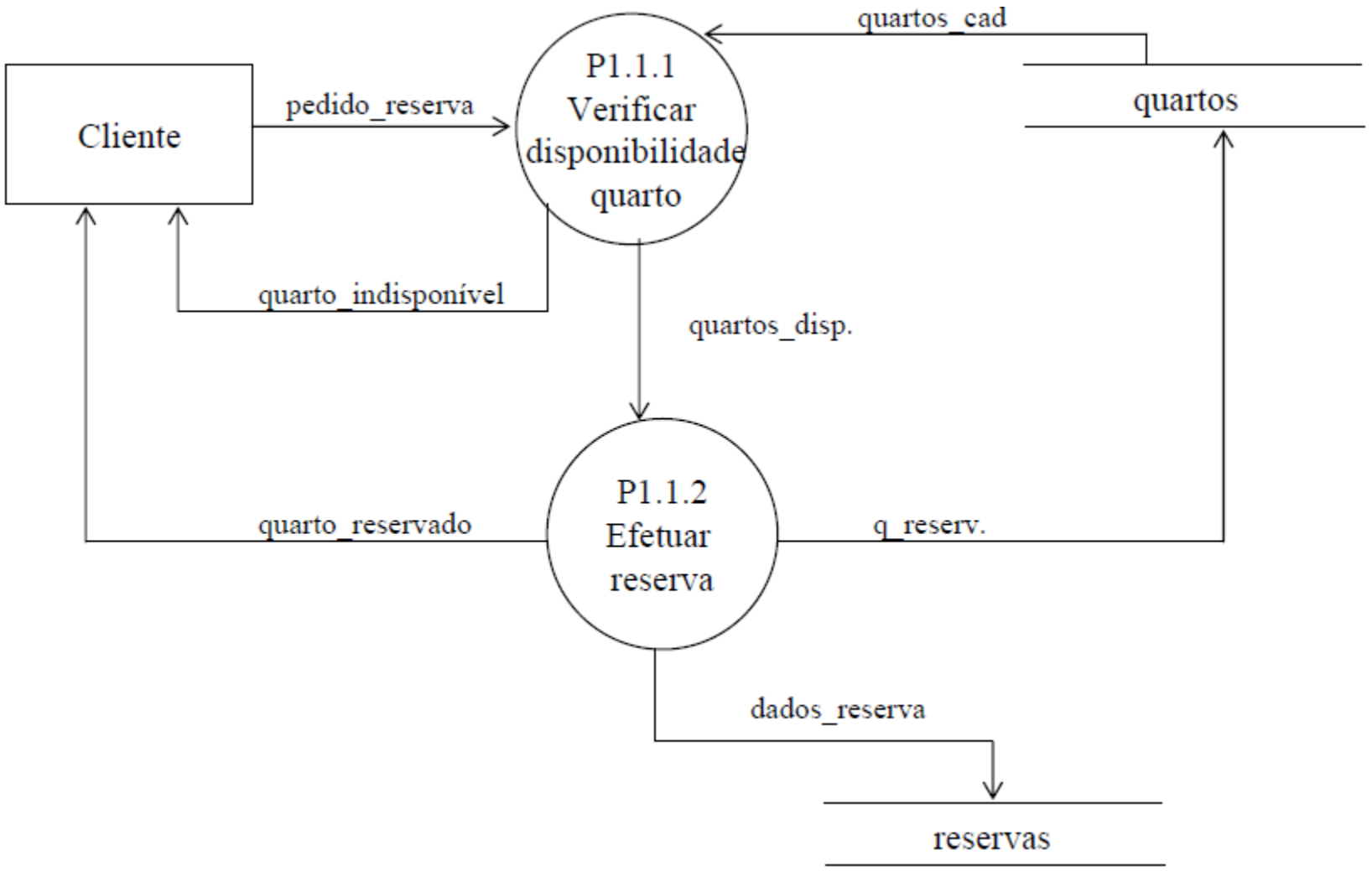


DFD – Nível 1

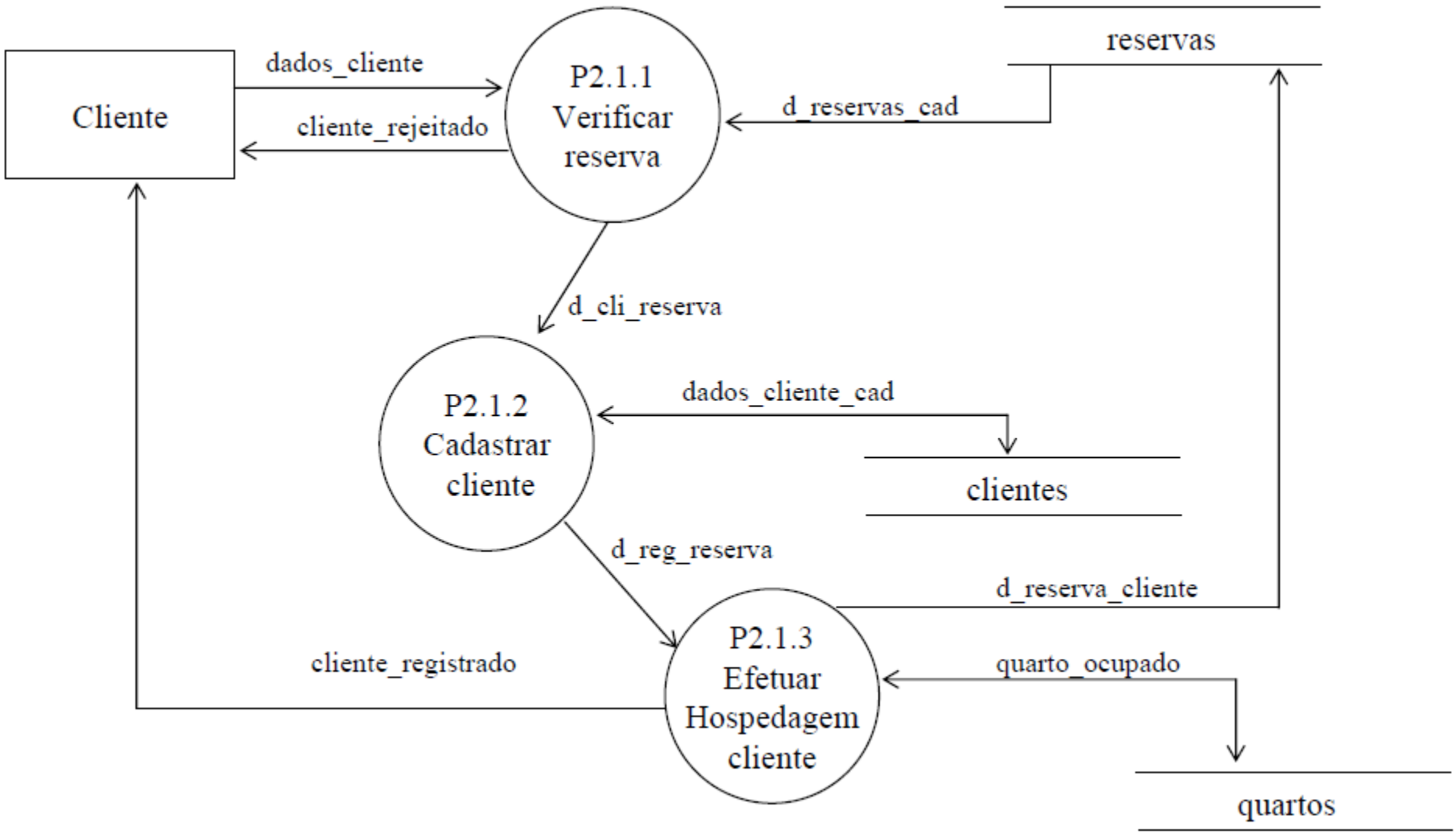
- Processo 3 detalhado



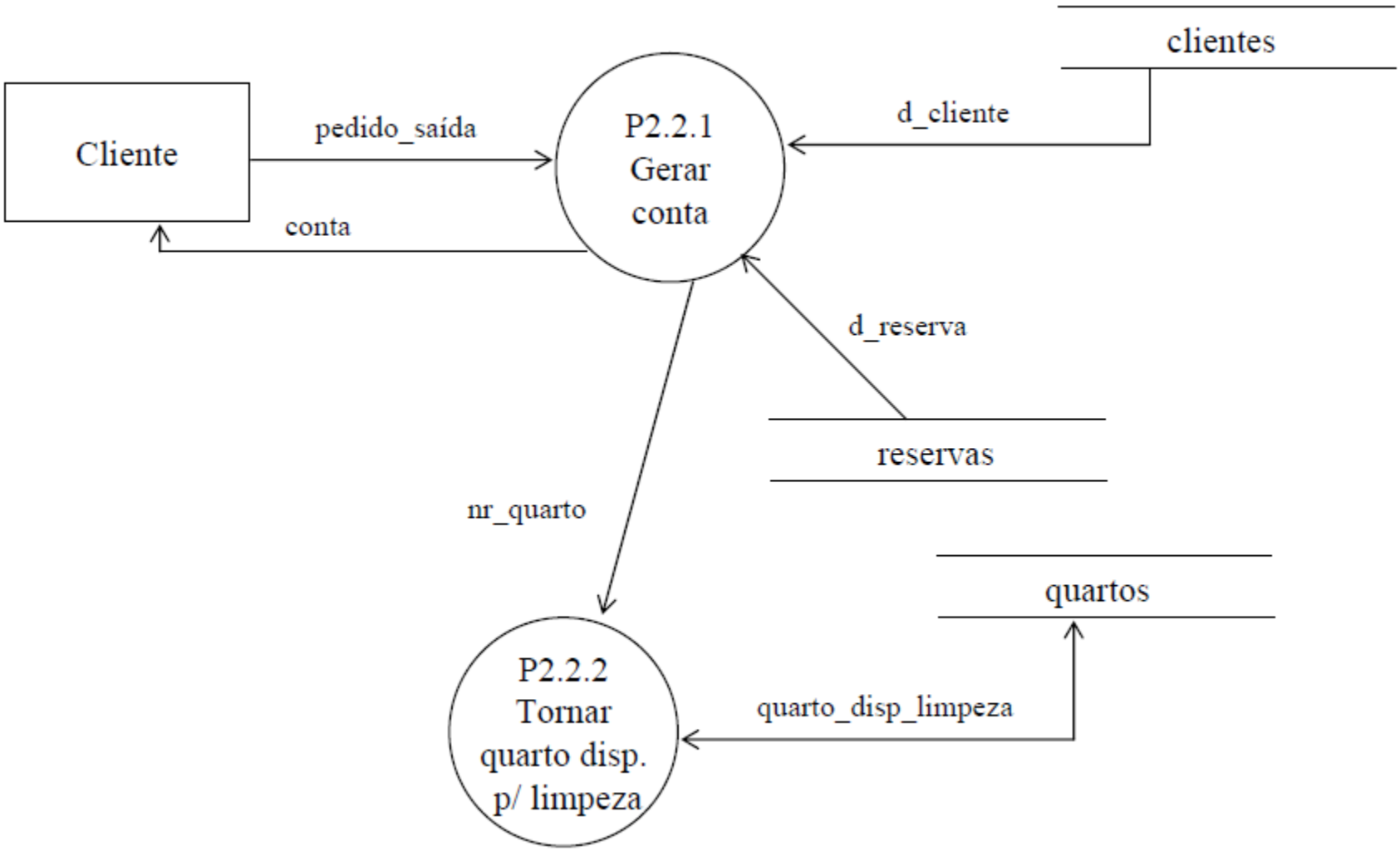
DFD – Nível 2: Refinamento de P1.1



DFD – Nível 2: Refinamento de P2.1



DFD – Nível 2: Refinamento de P2.2



Bibliografia

- **Básica:**

  BEZERRA, E. Princípios de Análise e Projetos de Sistemas com UML. Rio de Janeiro: Campus, 2003.

  PRESSMAN, R.S. Engenharia de Software. São Paulo: Makron Books, 2002.

  SOMMERVILLE, I. Engenharia de Software. São Paulo: Addison Wesley, 2003.

- **Complementar:**

  WARNIER, J. Lógica de Construção de Programas. Rio de Janeiro: Campus, 1984.

  JACKSON, M. Princípios de Projeto de Programas. Rio de Janeiro: Campus, 1988.

  PAGE-JONES, M. Projeto Estruturado de Sistemas. São Paulo: McGraw-Hill, 1988.