REÚSO DE SOFTWARE

Paulo C. Masiero

Eng. Software, Sommerville, Cap. 16, Ed. 9ª. e outros materiais

SUMÁRIO

1. Introdução
2. O Panorama de reúso
3. Geradores de Código
4. Framework de Aplicações
5. Linhas de Produtos de Software
6. Reúso de Produtos COTS
7. Padrões de Software

1- INTRODUÇÃO

- Reúso de software é uma estratégia em que o desenvolvimento de software é baseado no reúso de software existente → eng. de soft. baseada em reúso.
- Busca maximizar o reúso de software existente.
- História
  - Linguagens de Programação: sub-rotinas
  - McIroy, M. D. – Mass-produced software components, Proc. of the NATO Conf. On Software Engineering, Garmisch, Germany, Springer, 1968.

INTRODUÇÃO

- A engenharia de software baseada em reúso busca:
  - Menores custos de produção e manutenção de software;
  - Entregas mais rápidas;
  - Software com melhor qualidade;
  - Aumentar o retorno sobre o investimento em software .

INTRODUÇÃO (CONT.)

- O reúso pode acontecer com unidades de tamanhos muito diferentes:
  - Reúso de aplicações (ou sistemas): a aplicação toda é reusada, ou é configurada para diferentes clientes ou pode-se desenvolver uma família de aplicações com base em uma arquitetura comum;
  - Reúso de componentes (subsistemas): componentes de software de tamanho variado;
  - Reúso de objetos e funções: componentes que implementam uma única função (ocorre há mais de 40 anos). Ex. funções matemáticas.

INTRODUÇÃO (CONT.)

- Outra forma de reúso é o reúso de conceitos.
- Conceito: é uma representação abstrata que inclui detalhes de implementação. Por exemplo: um modelo de classes.
- O conceito pode ser instanciado e configurado para uma série de situações. Ex. padrões de projeto, padrões de arquitetura.

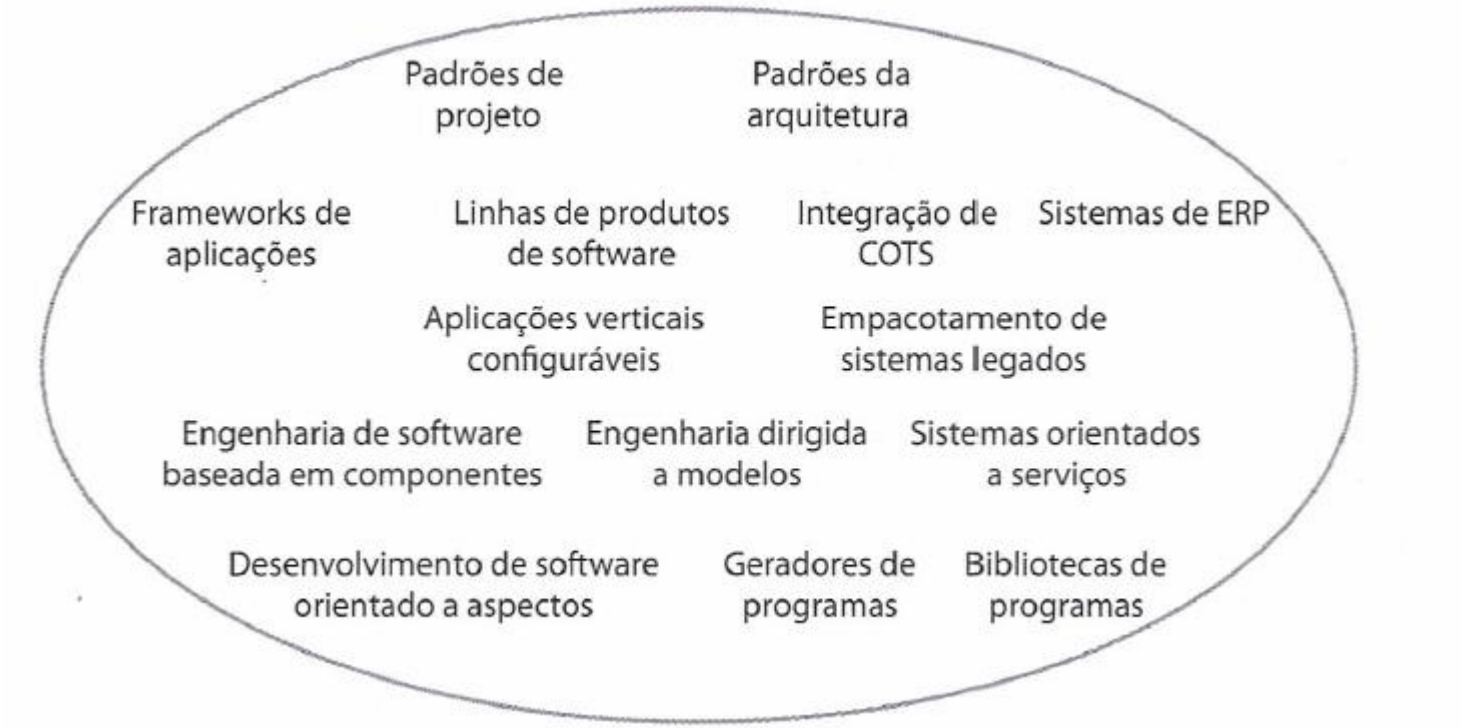
VANTAGENS (RESUMO)

- Redução dos custos de desenvolvimento.
- Confiança aumentada.
- Redução do risco de processo.
- Uso eficaz de especialistas.
- Conformidade com padrões.
- Desenvolvimento acelerado.

REÚSO: POSSÍVEIS PROBLEMAS

- Maiores custos de manutenção.
- Falta de ferramentas de suporte.
- Síndrome do “não-inventado-aqui”.
- Criação, manutenção e uso de uma biblioteca de componentes.
- Como encontrar, compreender e adaptar componentes reusáveis.

2 - O PANORAMA DE REÚSO



Abordagem	Descrição
Padrões de arquitetura	Padrões de arquitetura de software que oferecem suporte a tipos comuns de sistemas de aplicação são usados como base de aplicações. São descritos nos capítulos 6, 13 e 20.
Padrões de projeto	Abstrações genéricas que ocorrem em todas as aplicações são representadas como padrões de projeto, mostrando os objetos abstratos e concretos e as interações. São descritos no Capítulo 7.
Desenvolvimento baseado em componentes	Sistemas são desenvolvidos através da integração de componentes (coleções de objetos) que atendem aos padrões de modelos e componentes. São descritos no Capítulo 17.
Framework de aplicações	Coleções de classes abstratas e concretas são adaptadas e estendidas para criar sistemas de aplicação.
Empacotamento de sistemas legados	Sistemas legados (veja o Capítulo 9) são 'empacotados' pela definição de um conjunto de interfaces e acesso a esses sistemas legados por meio dessas interfaces.
Sistemas orientados a serviços	Sistemas são desenvolvidos pela ligação de serviços compartilhados, que podem ser fornecidos externamente. São descritos no Capítulo 19.
Linhas de produtos de software	Um tipo de aplicação é generalizado em torno de uma arquitetura comum para que esta possa ser adaptada para diferentes clientes.
Reúso de produto COTS	Sistemas são desenvolvidos pela configuração e integração de sistemas de aplicação existentes.
Sistemas de ERP	Sistemas de grande porte que sintetizam a funcionalidade e as regras de negócios genéricos são configurados para uma organização.
Aplicações verticais configuráveis	Sistemas genéricos são projetados para poder ser configurados para as necessidades dos clientes de sistemas específicos.
Bibliotecas de programas	Bibliotecas de classe e funções que implementam abstrações comumente usadas são disponibilizadas para reúso.
Engenharia dirigida a modelos	O software é representado como modelos de domínio e modelos de implementação independentes. O código é gerado a partir desses modelos. São descritos no Capítulo 5.
Geradores de programas	Um sistema gerador incorpora o conhecimento de um tipo de aplicação, e é usado para gerar sistemas nesse domínio a partir de um modelo de sistema fornecido pelo usuário.
Desenvolvimento de software orientado a aspectos	Quando o programa é compilado, os componentes compartilhados são integrados em uma aplicação em diferentes locais. São descritos no Capítulo 21.

QUESTÃO FUNDAMENTAL?

- Qual é a técnica mais apropriada para usar em determinada situação?
- Isso depende de....
  - Requisitos do sistema em desenvolvimento;
  - Tecnologia disponível;
  - Disponibilidade de ativos reusáveis;
  - Conhecimento e experiência (*expertise*) da equipe de desenvolvimento.

FATORES-CHAVE A CONSIDERAR PARA O REÚSO

- O cronograma de desenvolvimento do software.
- A expectativa de duração do software.
- O conhecimento, as habilidades e a experiência da equipe de desenvolvimento.
- A importância do software e seus requisitos não funcionais.
- O domínio da aplicação.
- A plataforma em que o sistema será executado.

HÁ UMA GRANDE GAMA DE ALTERNATIVAS E SITUAÇÕES

- Na maioria dos casos há a possibilidade de algum tipo de reúso (parcial).

- Muitas vezes o reúso não é atingido por problemas gerenciais e não técnicos. Ex. o gerente não quer comprometer os requisitos.

3 - GERADORES DE CÓDIGO (PROGRAMAS, APLICAÇÕES)

- Geradores de código traduzem especificações para código de programas (aplicações), ou outros artefatos, como casos de teste e interfaces.
- A especificação define o problema ou tarefa a ser realizada. Pode ser informada como um diálogo interativo, em forma gráfica, ou então como uma linguagem.

UMA ABORDAGEM PARA CRIAR UM GERADOR

1. Reconhecer um domínio.
2. Definir os limites do domínio.
3. Definir o modelo subjacente (ou arquitetura)
4. Definir as partes variáveis e invariáveis.
5. Definir a especificação de entrada.
6. Definir os produtos.
7. Implementar o gerador.

Building Application Generators, J. C. Cleaveland, IEEE Software, 5(4),88

EXEMPLO

- Uma coleção de mil palavras é classificada em conjuntos.
- Queremos criar um programa em C que define essas palavras representadas como um vetor, isto é, queremos gerar as estruturas de dados em C.
- Cada conjunto de plavras é representado como um vetor de strings
- As coleções são representadas como vetores com o nome, o tamanho e os valores de cada conjunto.

Especificação (A)

colors {red, blue, green}

cities {Boston, Andover}

bugs {ant, spider, fly, moth, bee}

Note que a especificação pode ter um número variável de conjuntos e eles podem ter um número variável de palavras.

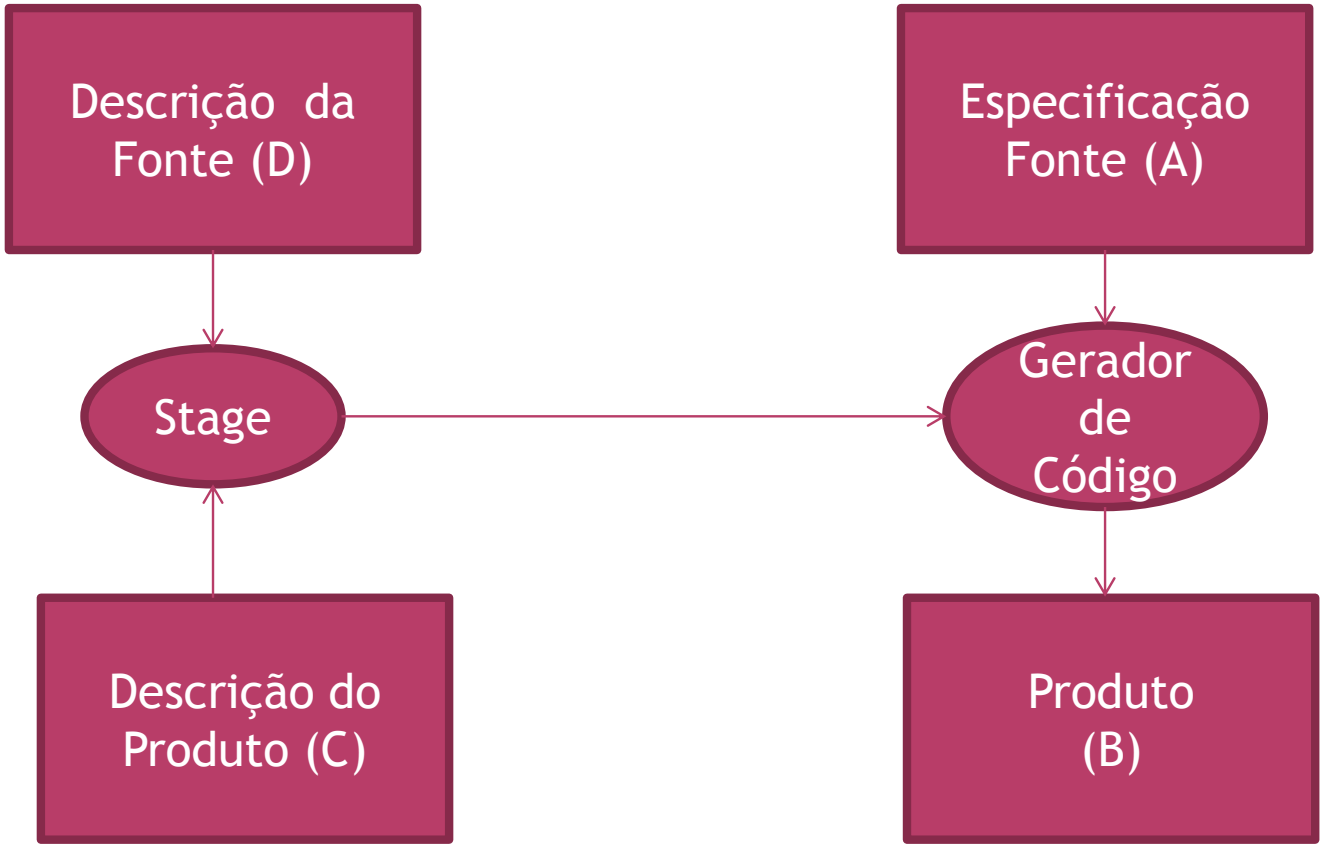
O RESULTADO OU PRODUTO (B)

```
int number_of_sets = 3;
char *name_of_set[]= { “colors”, “cities”,
                                    “bugs”,};
int size_of_set[] = {3,2,5,};
char *set_of_colors[] = { “red”, “blue”, “green”,};
char *set_of_cities[] = { “Boston”, “Andover”,};
char *set_of_bugs[] = { “ant”, “spider”, “fly”,
                          “moth” , “green”,};
char *values_of_set[] = { set_of_colors,
                            set_of_cities,
                            set_of_bugs,};
```

NOTE QUE

- A especificação é mais simples de escrever, ler e modificar.
- Exemplo: veja o que requer remover o conjunto de cidades.
- Tradicionalmente, você escreveria o programa (B) a partir das especificações (A).
- Um gerador resolveria o mesmo problema e ofereceria mais flexibilidade.

EXEMPLO DE UM GERADOR BASEADO EM LINGUAGEM (STAGE)

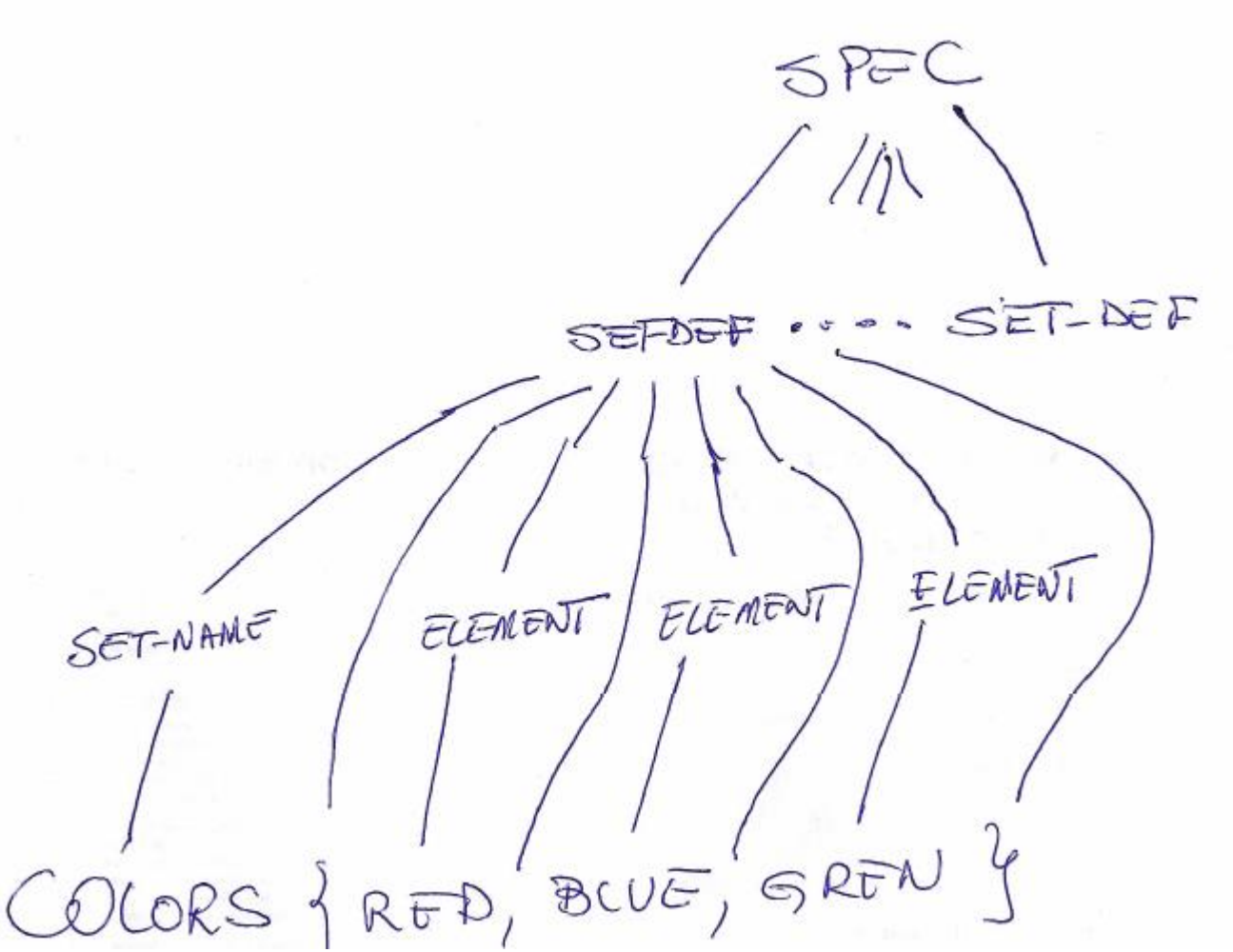


DESCRIÇÃO DA FONTE (D)

```
%grammar
spec: (set_def+)
set_def: ( set_name “{“
                element+ separated-by “,” “}”
          )
set_name: <[a-zA-Z0-9_]+>
element: <[a-zA-Z0-9]+>
%product_table.c
```

DESCRIÇÃO DO PRODUTO (C)

```
%template
int number_of_sets = %d(length(set_def(top))));

char *name_of-set[] = { %forall n:SET_NAME %loop  " %(tok(n))",
                                        %end-loop };

int size_of_set[] = { %forall s:SET-DEF
                          %loop  %d(length(element(s))) ,
%end-loop };

%forall  s:SET-DEF  %loop

char *set_of_%(tok(set_name(s)))[] = {
        %forall e:ELEMENT in s  %loop " %(tok(e)) ", %end-loop
%end-loop

char **values_of_set[] = { %forall n:SET_NAME  %loop
                                    set_of%tok(n)) ,
 %end-loop };
```


SPEC
SET-DEF .... SET-DEF
SET-NAME
ELEMENT
ELEMENT
ELEMENT
COLORS { RED, BLUE, GREN }

Evolução das idéias de geração de código

- Domínios de aplicações, Famílias de aplicações, linhas de produto de software.
- Linguagens de Específicas de Domínios
  - Textuais
  - Modelos
- OMG: modelos no contexto de Orientação a Objetos eUML.
  - MDD: Model Driven Development
  - MDA: Model Driven Arquitecture

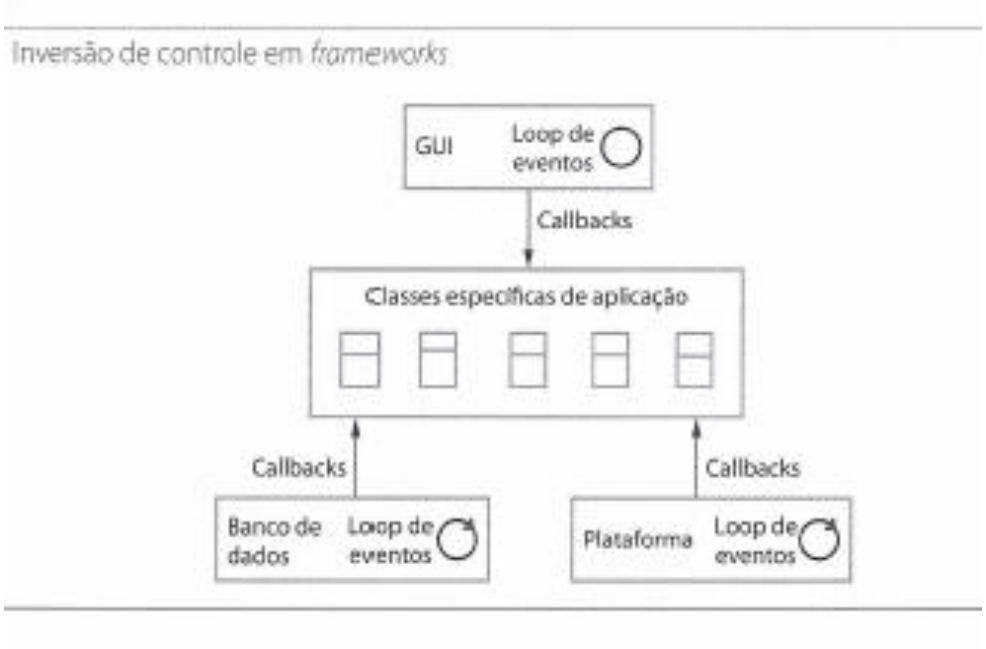
4 - FRAMEWORKS DE APLICAÇÕES

- É uma estrutura genérica que deve ser estendida para criar uma aplicação específica.
- Um conjunto integrado de artefatos de software (como classes, objetos e componentes) que colaboram para fornecer uma arquitetura reusável para uma família de aplicações relacionadas.
- São implementados como uma coleção de classes concretas e abstratas em uma linguagem OO. Exemplos: Hibernate, Struts.

FRAMEWORKS DE APLICAÇÕES

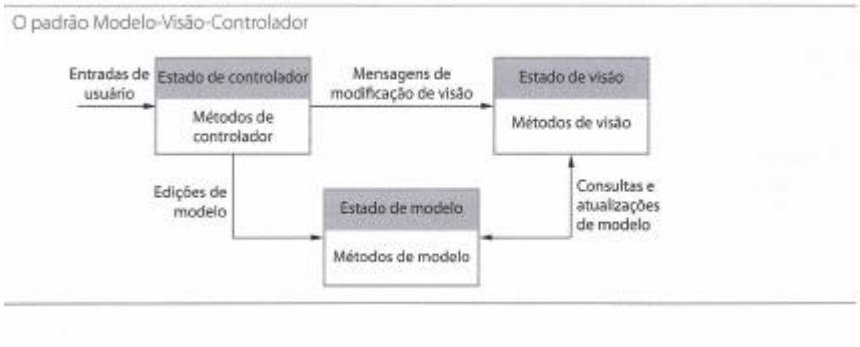
- Para instanciar um framework deve-se adicionar a ele classes concretas.
- Para usar/estender um Fm deve-se acrescentar classes concretas que herdam operações de classes abstratas e também inserir “call backs” para os métodos da aplicação (Inversão de controle).
- Podem ser caros e complexos para desenvolver.

INVERSÃO DE CONTROLE



EXEMPLO: O FRAMEWORK MVC

- O padrão MVC foi proposto na década de 60 para o projeto de Interface Humano-Computador.
- O Framework MVC permite representar dados de diferentes maneiras.
- O framework MVC inclui também os padrões GoF observador, estratégia e composição.



5 - LINHA DE PRODUTOS DE SOFTWARE - DEFINIÇÕES BÁSICAS

Definições Básicas:

- Domínio: É um conjunto de aplicações existentes e futuras que compartilham dados e funcionalidades comuns (também chamado de domínio de aplicação)
- Análise de domínio: é o processo de identificar, coletar, organizar e representar as informações relevantes de um domínio com base no estudo de sistemas existentes e suas histórias de desenvolvimento, no conhecimento de especialistas no domínio, na teoria subjacente e na tecnologia emergente dentro do domínio.

MODELO DE PROCESSO

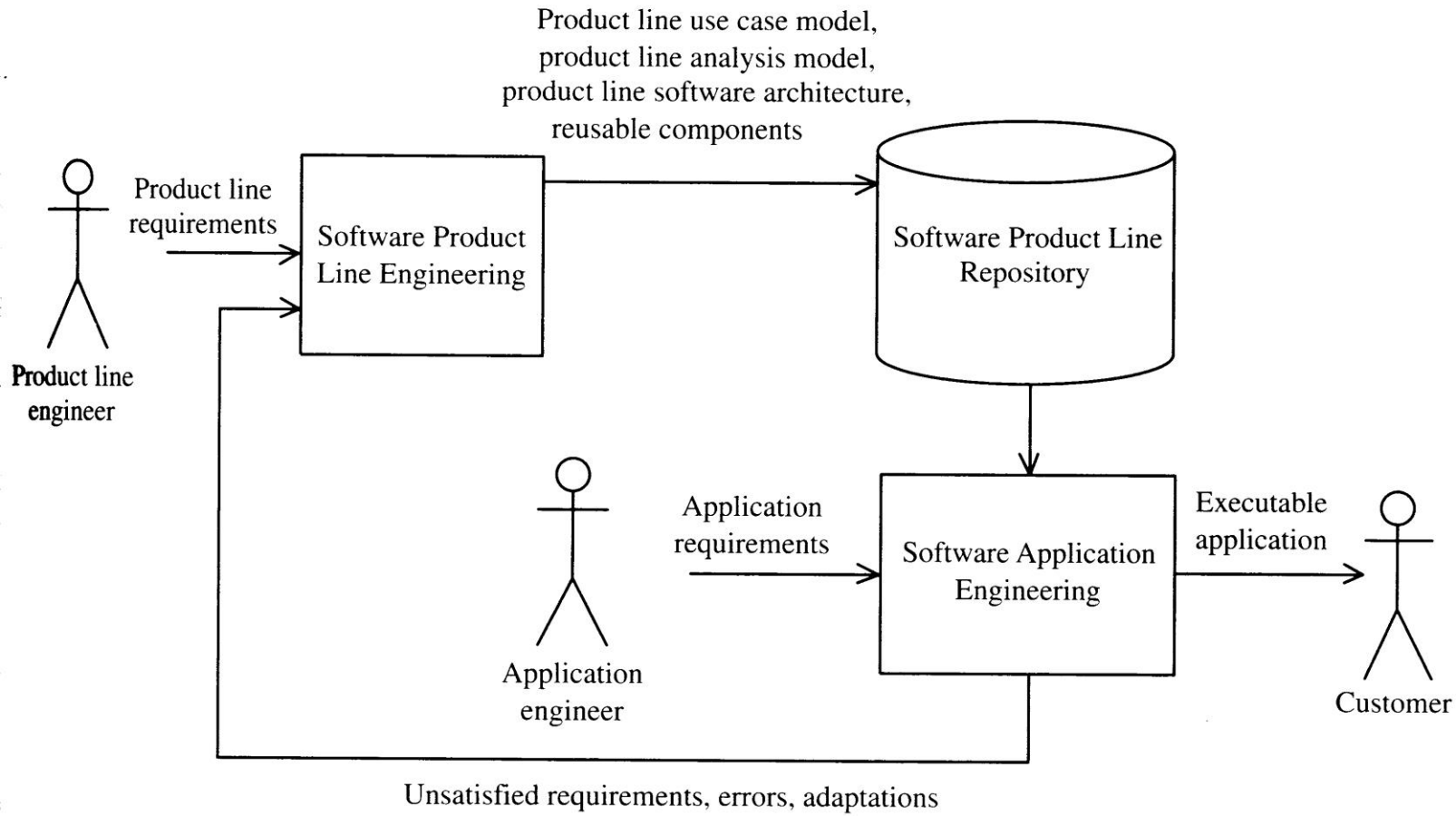


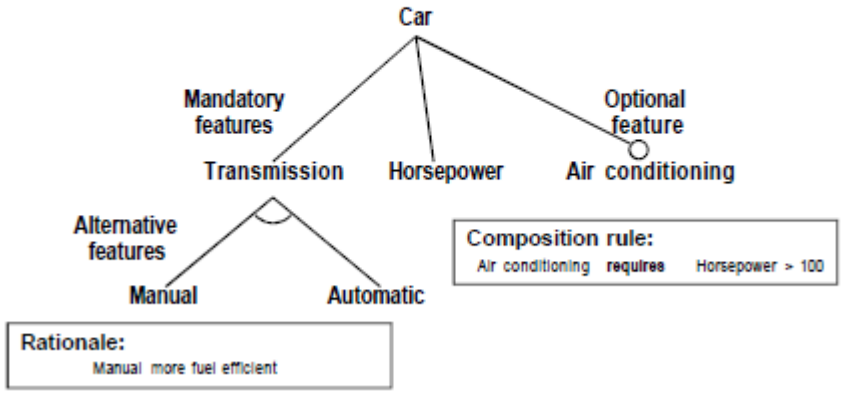
Figure 3.1 *Evolutionary Software Product Line Engineering Process*

LINHA DE PRODUTOS DE SOFTWARE

- Um conjunto de sistemas intensivos em software que compartilha um conjunto de características gerenciadas e comuns, que satisfaz a necessidades específicas de um segmento de mercado em particular ou missão de uma empresa e que é desenvolvido a partir de um conjunto principal de recursos (ativos) de uma forma pré-estabelecida (Clements and Northrop 2002).
- Podem ser implementadas usando geradores de aplicação, componentes, frameworks, compilação condicional, aspectos, etc.
- O reúso é limitado e planejado.

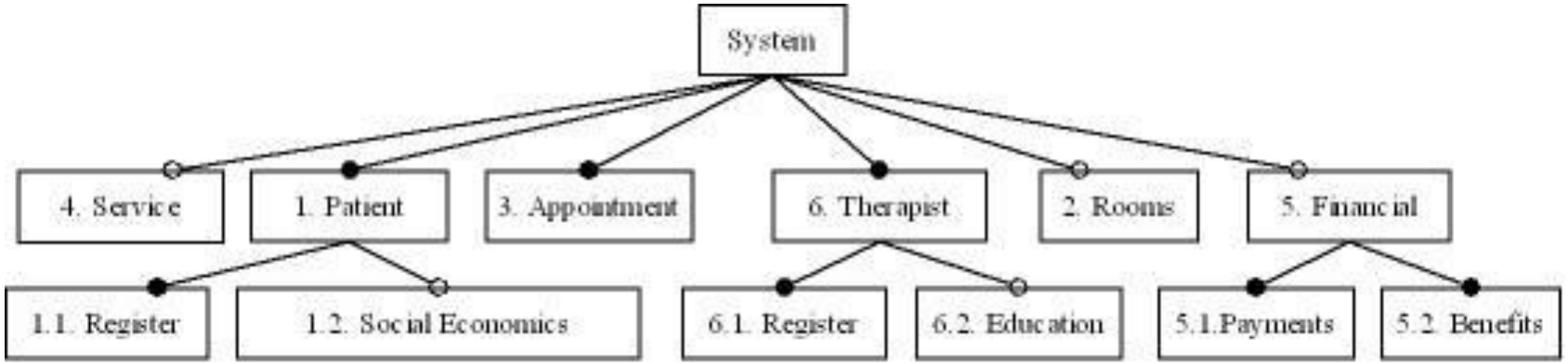
Feature: A prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems [American 85].

Análise de Features

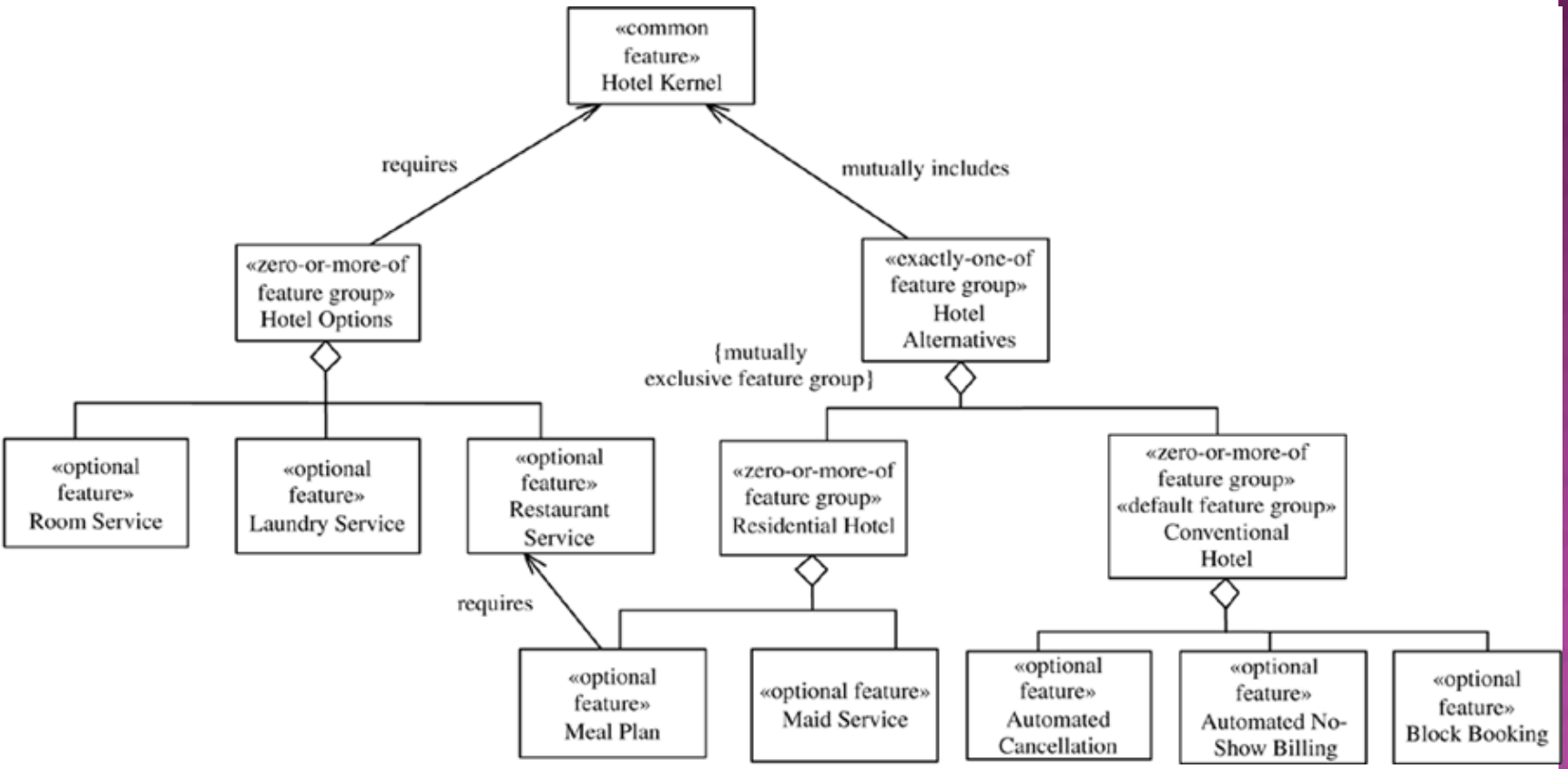


Kang, K. 1990, SEI

O Diagrama de Características "Original"



MODELAGEM AVANÇADA (GOMAA 2004)



6 - PADRÕES DE SOFTWARE

- Descrevem soluções para certos problemas que ocorrem frequentemente durante o desenvolvimento de sistemas, em diferentes níveis de abstração.
- Tipos de padrões
  - De código (idiomas)
  - De projeto (Go4)
  - De arquitetura
  - De análise

PADRÃO ITERADOR

- Nome: Iterador (Gamma 94)
- Objetivo:
  - Existe a necessidade de percorrer agregados (ou coleções) quaisquer, em diversas ordens, sem conhecer sua representação subjacente
- Motivação:
  - além de acessar os elementos sem conhecer a estrutura interna do agregado, pode ser desejável percorrer o agregado de diferentes formas, sem poluir a interface com inúmeros métodos para isso e mantendo controle do percurso.

PADRÃO ITERADOR

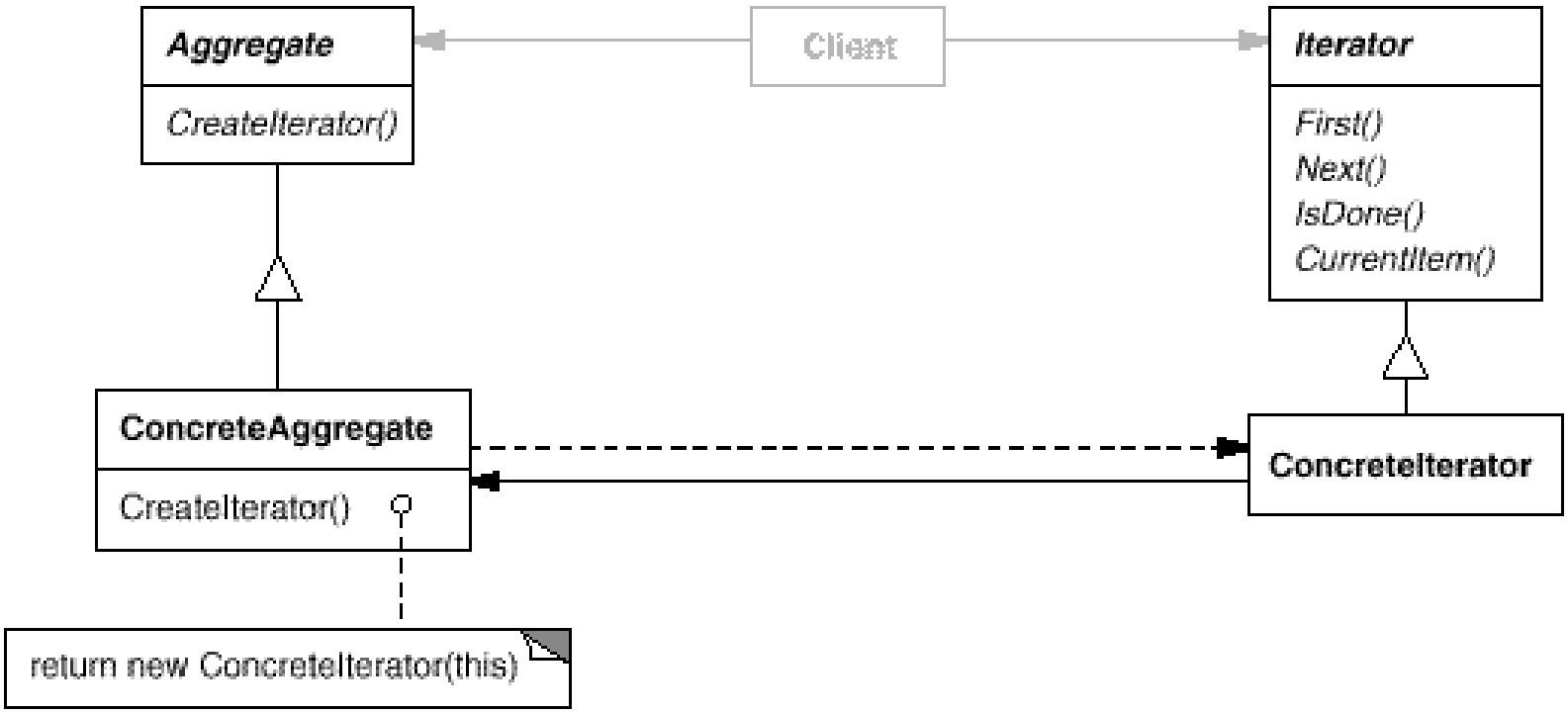
- Solução: Criar uma classe abstrata Iterador, que terá a interface de comunicação com o cliente. Para cada agregado concreto, criar uma subclasse de Iterador, contendo a implementação dos métodos necessários

PADRÃO ITERADOR

- Aplicabilidade
- Use o padrão Iterador
  - Para acessar o conteúdo de um objeto agregado sem expor sua representação interna.
  - Para apoiar diferentes percursos de objetos agregados.
  - Para conseguir uma interface uniforme para percorrer diferentes estruturas de agregados (isto é, para apoiar a iteração polimórfica).

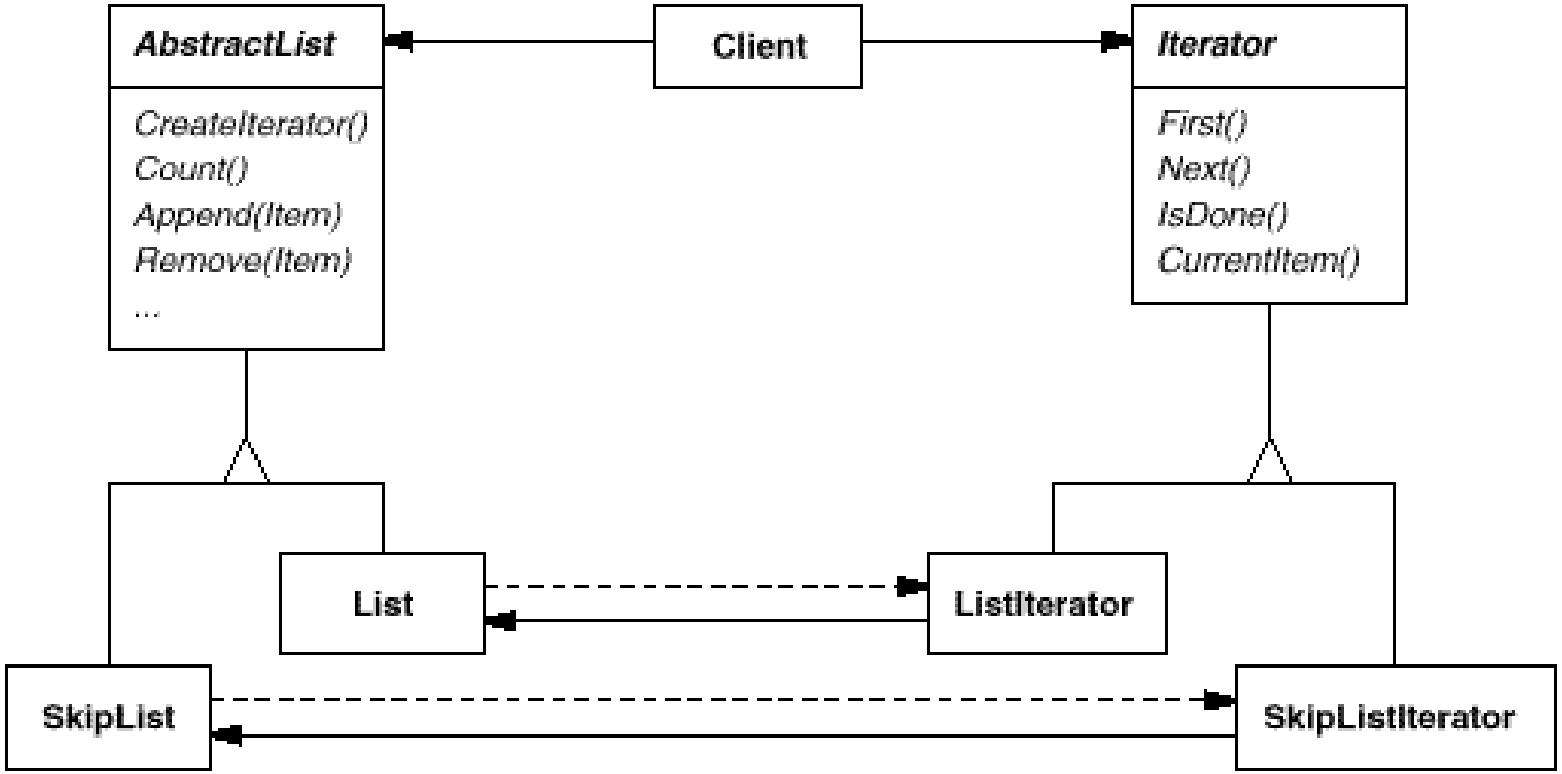
(Gamma 94)

PADRÃO DE PROJE Iterator

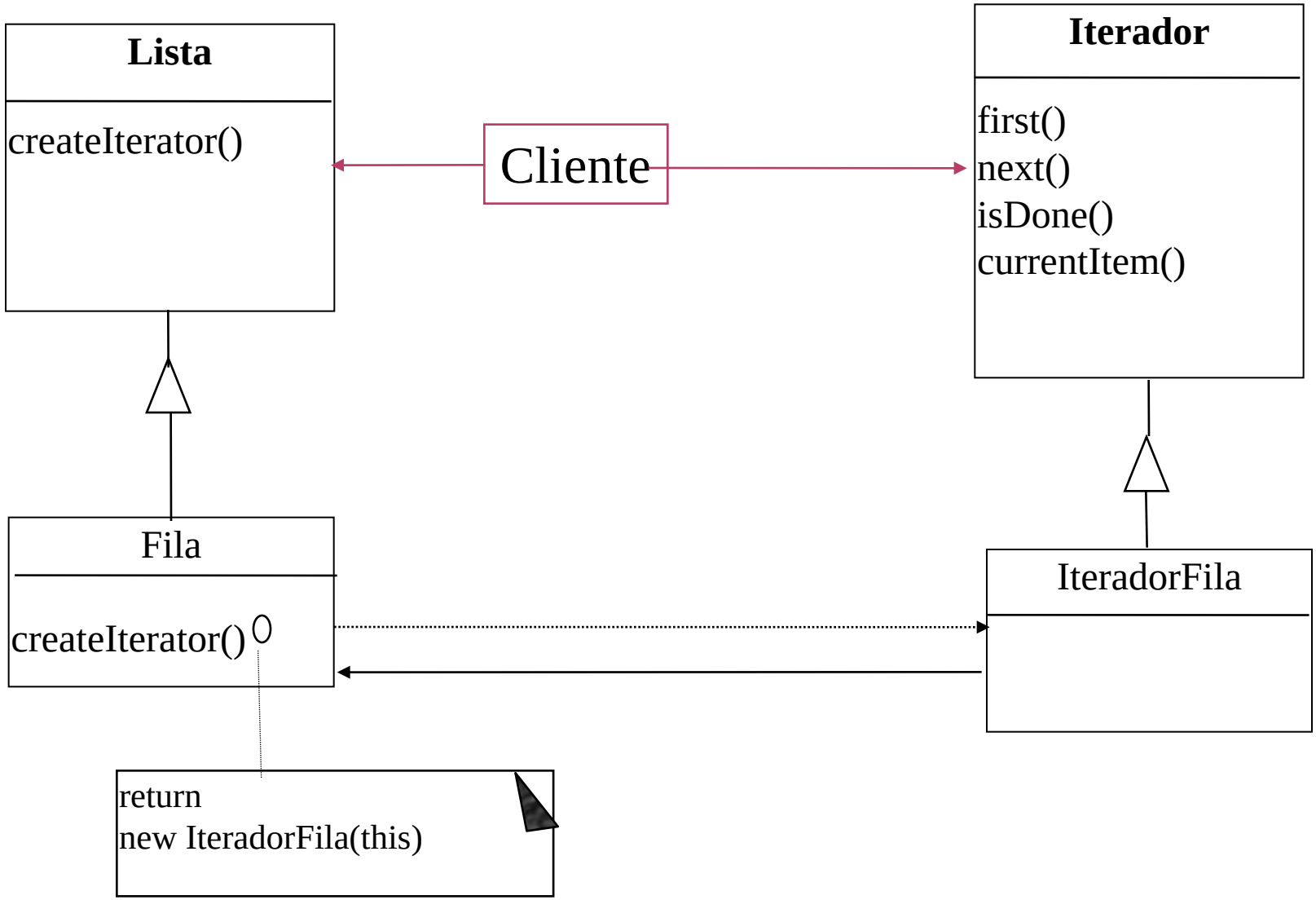


(Gamma 94)

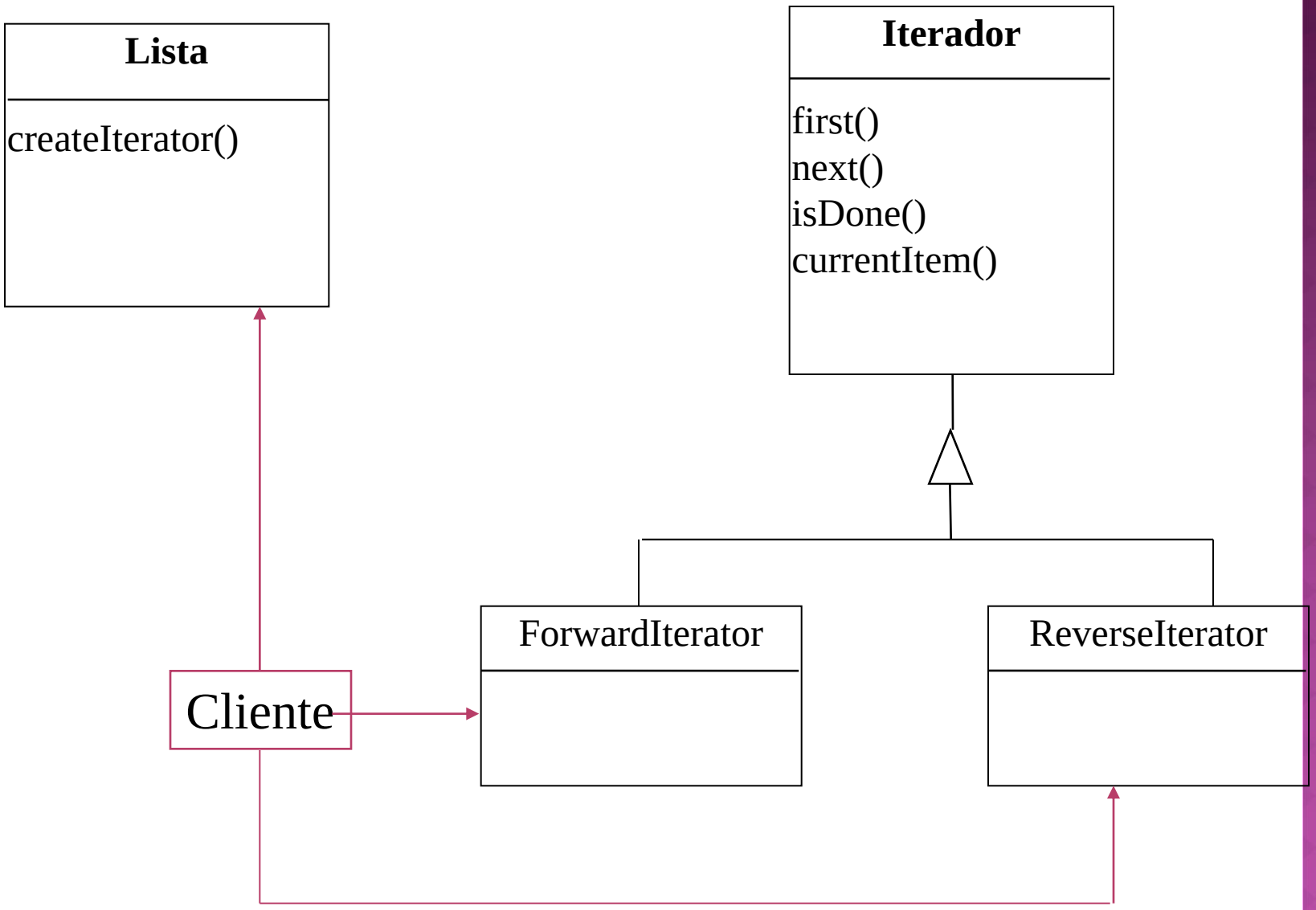
EXEMPLO 1



EXEMPLO 2


Lista
createIterator()
Cliente
Iterador
first()
next()
isDone()
currentItem()
Fila
createIterator()
IteradorFila
return
new IteradorFila(this)

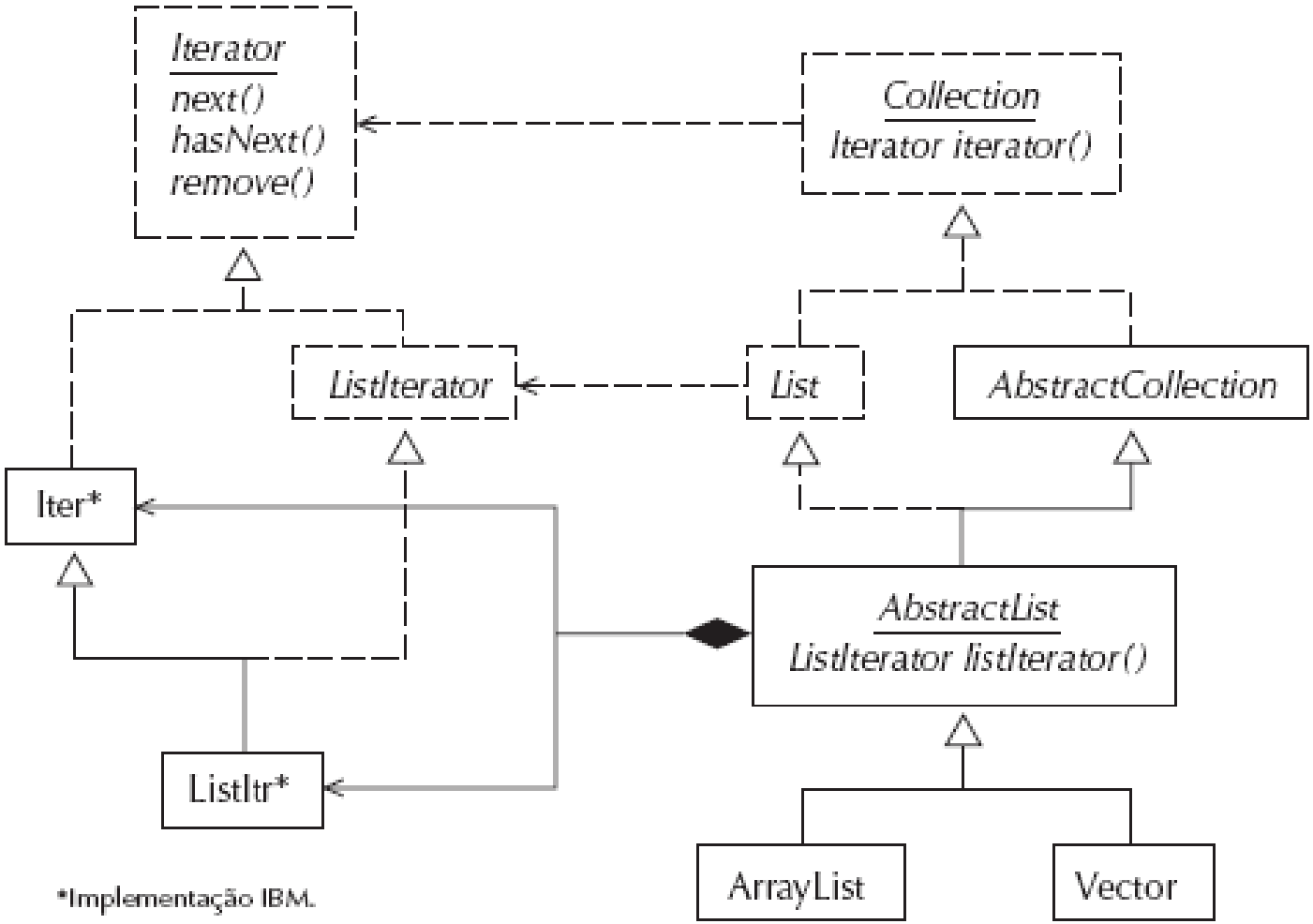
Exemplo 3


Lista
createIterator()
Iterador
first()
next()
isDone()
currentItem()
ForwardIterator
ReverseIterator
Cliente

ITERADOR NA API JAVA

- O pacote *java.util* contém uma interface *Iterator* com os métodos:*next()*, *hasNext()* e *remove()*.
- O método *next()* é uma combinação de *incrementar()* e *obterElementoAtual()*
- A subinterface *ListIterator* especifica esses métodos para operarem nos objetos *List* e adiciona os métodos apropriados aos objetos *List*.
- A interface *List* tem um método *listIterator()*, que retorna um objeto *ListIterator* que itera sobre ele. Isso permite ao programador mover-se e iterar sobre objetos *List*.

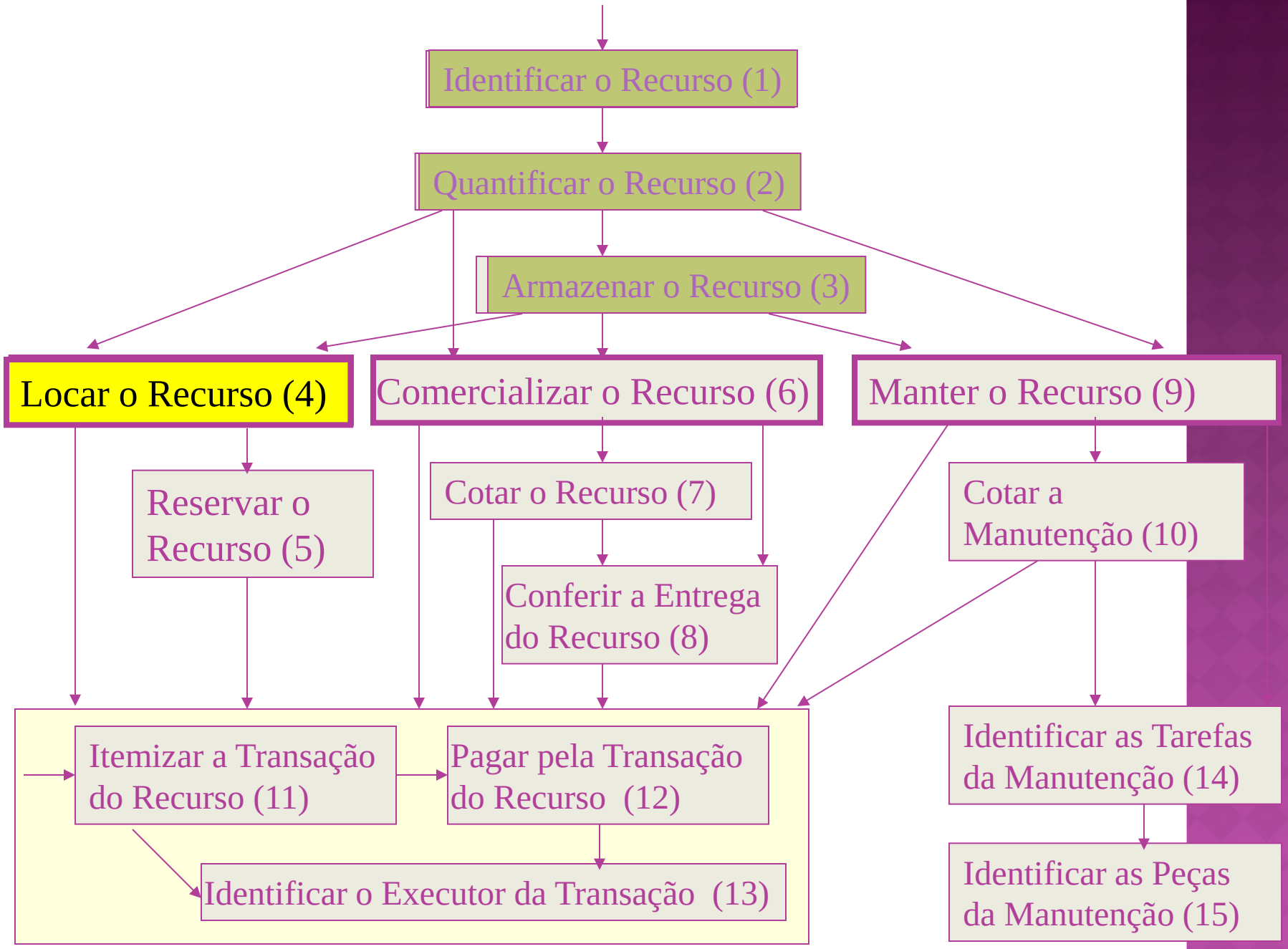
ITERADOR NA API JAVA



USO DO ITERADOR DA API JAVA

```
public void devolverCopia(int codCopia)
   {
    Iterator i = linhas.iterator();
    Date dataDeHoje = new Date();
    while (i.hasNext()) {
        LinhaDeEmprestimo linha = (LinhaDeEmprestimo) i.next();
        cc=linha.codigoCopia();
        if (cc==codCopia)
            linha.atualizaDataDev(dataDeHoje);
       }
   }
}
```

Exemplo: Uma linguagem de Padrões para Gestão de Recursos de Negócios



EXEMPLO DE UM PADRÃO

Padrão 4: Locar o Recurso

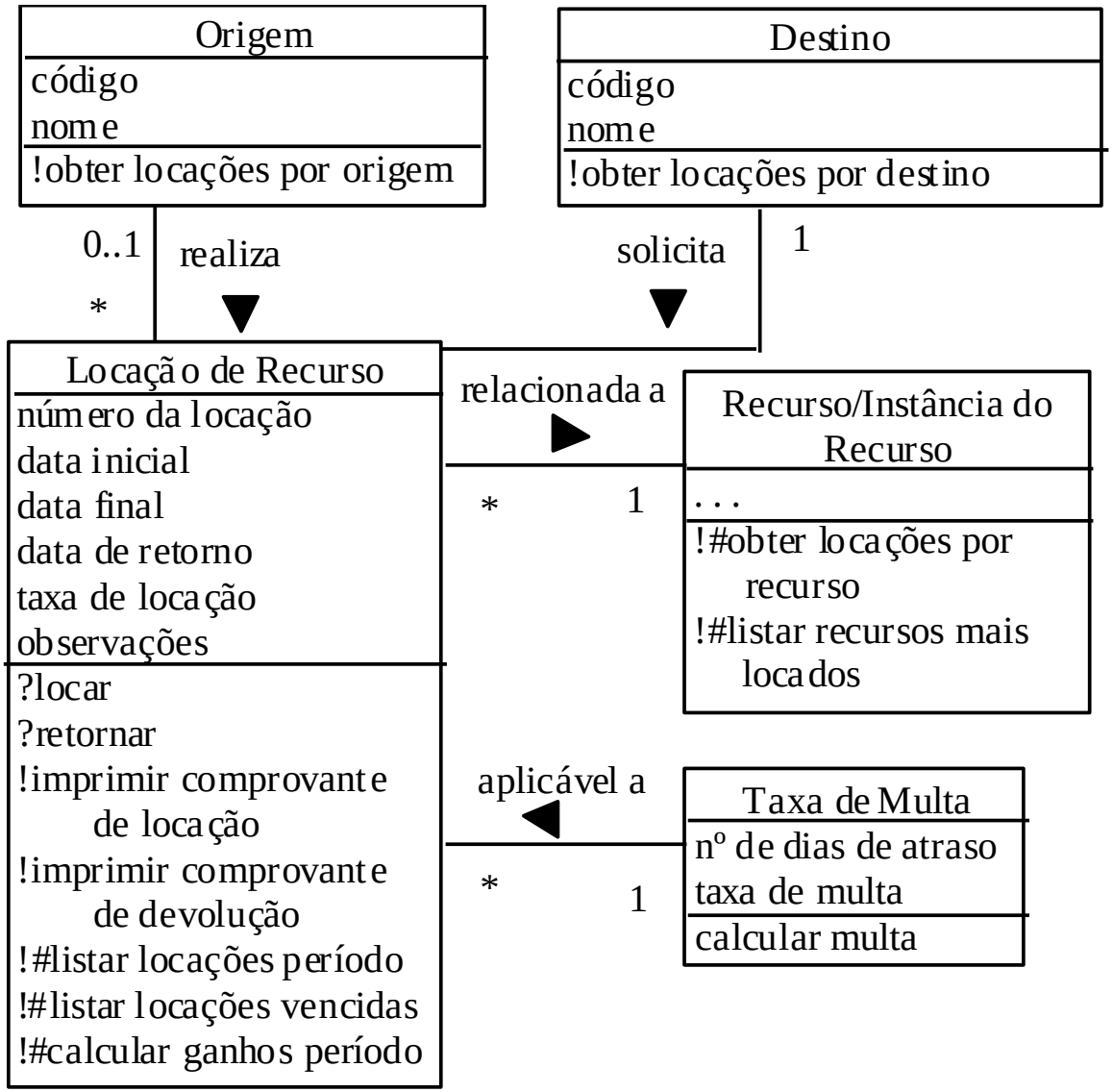
- Contexto
  - Sua aplicação lida com aluguel de recursos, que podem ser bens emprestados a um cliente por um certo período ou serviços efetuados por um especialista por determinado tempo. Você já identificou e quantificou tais recursos.
- Problema
  - Como gerenciar aluguéis de recursos realizados por sua aplicação?

EXEMPLO DE UM PADRÃO (CONT.)

- Influências
  - Existem diversos detalhes envolvidos no aluguel de um recurso. Armazenar informação sobre esses detalhes é importante para conseguir um bom gerenciamento de recursos disponíveis e alugados.
  - Saber quais foram os aluguéis anteriores pode ajudar gerentes a prever quais recursos merecem mais investimento em futuras aquisições.
  - Os prós e contras entre uma melhor funcionalidade do sistema, espaço de armazenamento adicional e maior tempo para processamento da informação devem ser analisados cuidadosamente.

EXEMPLO DE UM PADRÃO (CONT.)

- Estrutura



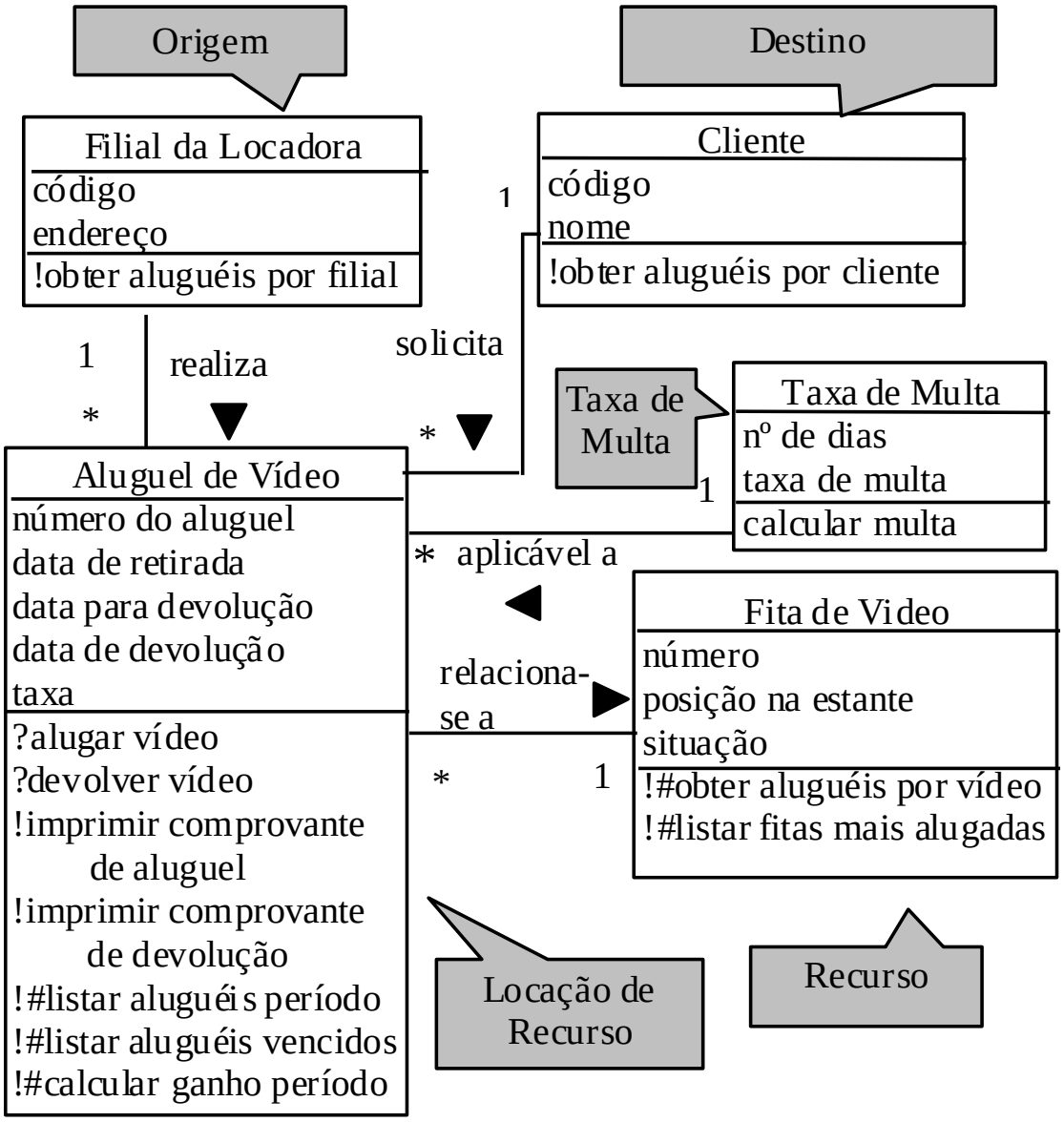
EXEMPLO DE UM PADRÃO (CONT.)

- Participantes
  - **Locação de Recurso**: representa todos os detalhes envolvidos na locação do recurso. Existem métodos para `locar` o recurso, `retornar` o recurso (quando o cliente o devolve ao sistema) e para `calcular ganhos` (por exemplo, ganho mensal ou semanal). Se o sub-padrão Recurso Mensurável foi adotado, um atributo deve ser adicionado a essa classe para denotar a `quantidade` de recursos locados.
  - **Recurso/Instância do Recurso**: conforme descrito em padrões anteriores. Utilize a classe Instância do Recurso apenas se o sub-padrão Recurso Instanciável tiver sido adotado. Caso contrário, utilize a classe Recurso. Deve-se também decidir se a instância a ser locada é automaticamente fornecida pelo sistema ou se o usuário informa esse dado. Se o sub-padrão Recurso Instanciável foi adotado, métodos para obter e/ou listar a quantidade de instâncias disponíveis para locação em um certo período podem ser adicionados à classe Recurso.

EXEMPLO DE UM PADRÃO (CONT.)

- Participantes (cont.)
  - **Origem**: representa a filial ou departamento da organização que realiza a locação. Essa classe é opcional nesse padrão, já que em pequenos sistemas a origem é a própria organização e portanto não se justifica criar uma classe para representá-la.
  - **Destino**: representa o beneficiário da locação (por exemplo, o cliente que solicitou o aluguel).
  - **Taxa de Multa**: contém as regras de negócio que guiam o cálculo da multa a ser cobrada quando o recurso é devolvido após a data de vencimento.

EXEMPLO DE UM PADRÃO (CONT.)



7- REÚSO DE PRODUTOS COTS (SOFTWARE PRONTO)

- Produto de prateleira (Commercial-of-the-shelf): é um produto que pode ser adaptado para as necessidades de vários clientes sem alterar o código fonte do sistema.
- Como é projetado para uso geral, costuma incluir muitos recursos e funções.
- A adaptação se faz por mecanismos internos de configuração.
- Ex. Sistema de Controle de Pacientes em um Hospital: relatórios de saída e formulários de entrada podem ser adaptados para diferentes tipos de clientes.

7- REÚSO DE PRODUTOS COTS (SOFTWARE PRONTO)

- Baseado em “melhores práticas”.
- Muitas vezes, produtos de código aberto também são usados sem alterar o código fonte, como se fossem COTS.

REÚSO DE PRODUTOS COTS (CONT.)

- Esta abordagem tem sido amplamente usada por empresas nas últimas duas décadas, para Sistemas de Informação.
- Vantagens:
  - Rapidez de implantação
  - Não precisam dedicar muitos recursos para TI
  - A evolução é simplificada
  - Alguns riscos de desenvolvimento são evitados
  - É possível escolher as funções que necessita

REÚSO DE PRODUTOS COTS (CONT.)

- Desvantagens/Riscos
  - Os requisitos precisam ser adaptados ao COTS
  - Quando os requisitos não podem ser adaptados, a empresa precisa mudar seus processos de negócio.
  - A escolha do COTS que melhor se adequa à empresa pode ser difícil e custosa, principalmente se não houver especialistas locais e for preciso contratar consultores. Pode ser desastroso se errar.
  - O fornecedor do COTS controla a evolução do sistema.

Reúso de produtos COTS (cont.)

- Há dois tipos de reúso de produtos COTS:
  - Sistemas de solução COTS
    - Solução genérica de um único fornecedor , configurada para os requisitos do cliente.
    - Ex. Um sistema de gestão de bibliotecas
  - Sistemas integrados COTS
    - Envolve a integração de dois ou mais sistemas COTs, talvez de diferentes fornecedores.
    - Ex. Um sistema ERP, como os da SAP, ORACLE, TOTVS etc.