PUC Minas

Projeto de Software

Prof. Dr. João Paulo Aramuni

Unidade 4

Modelagem de Interação

PDS - Manhã /Noite

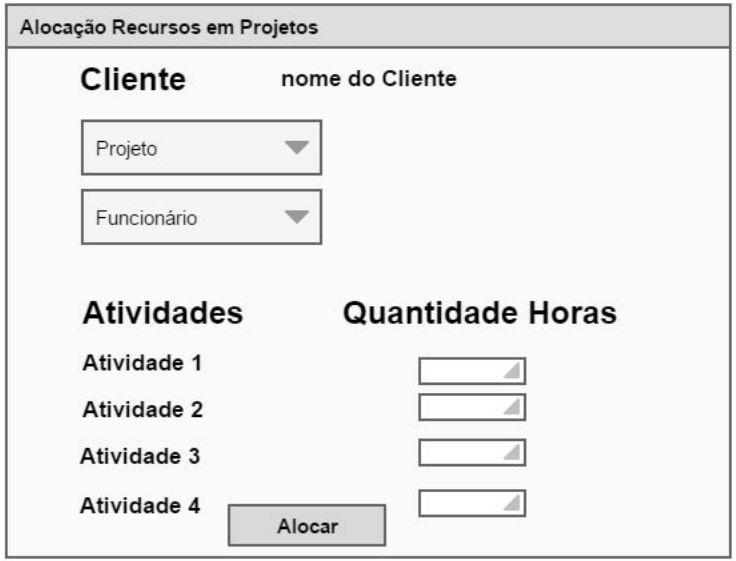
Modelagem de Interação

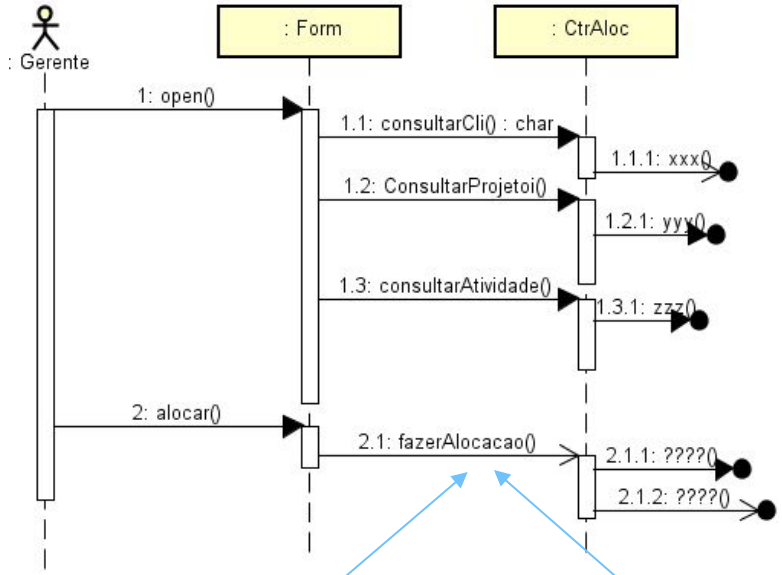
Sumário

- Conceito
- Definição de pré-condição
- Definição de pós-condição

Modelagem de Interação

Responsabilidade de operações





Como gerar informações para testes de unidade?

O que ela deve produzir?

O que essa operação espera receber como parâmetros?

Modelagem de Interação

Contratos

- A modelagem funcional produz os seguintes artefatos:
  - Modelo conceitual.
  - Os casos de uso expandidos ou diagramas de sequencia.

- Na construção dos diagramas de sequencia, os comandos e consultas de sistemas que devem ser implementados são identificados.

- Isso implica na existência de uma intenção por parte do usuário.

- Essa intenção é capturada pelos contratos de Operações ou Contratos de Consultas de sistemas.

Modelagem de Interação

Contratos

- Casos de uso ou características do sistema são a principal maneira para descrever o comportamento do sistema
  - Às vezes uma descrição mais detalhada ou precisa do comportamento tem valor

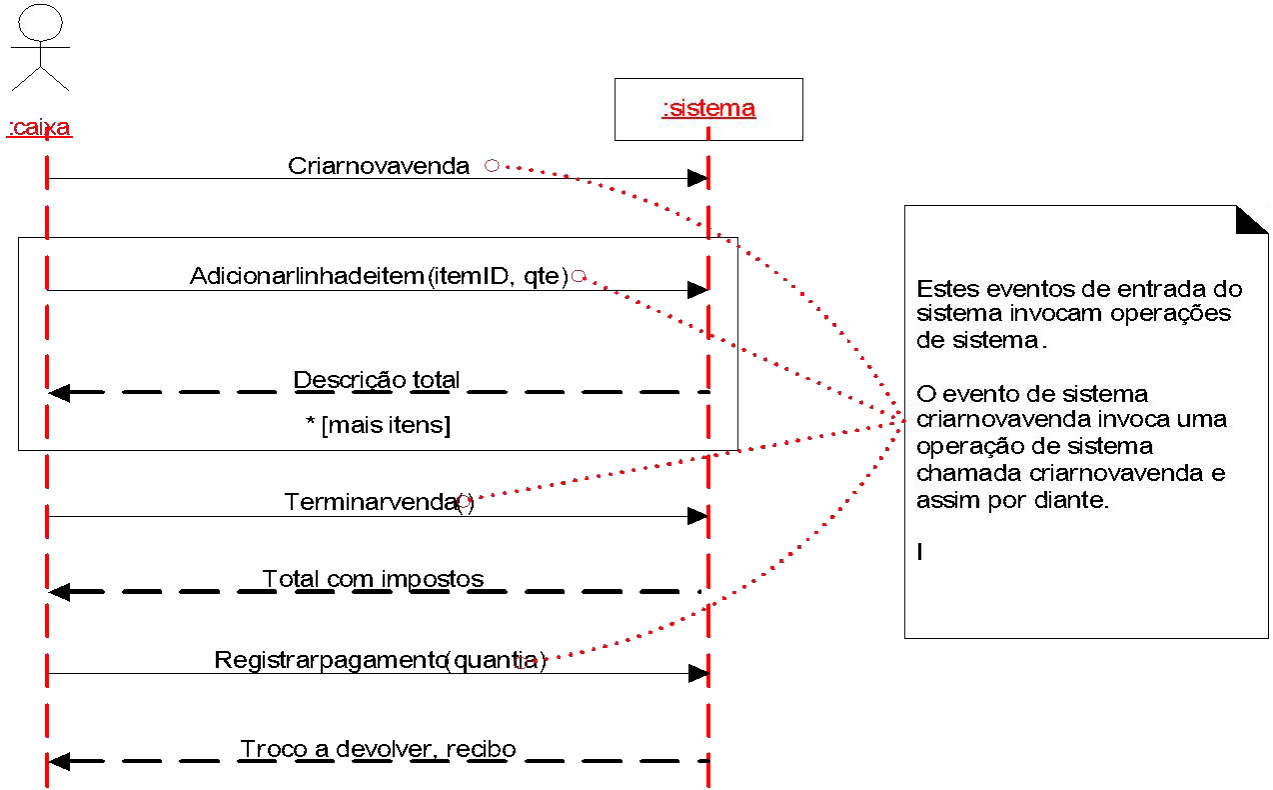
- Contratos de uma operação usam uma forma pré e pós-condição para descrever modificações detalhadas em objetos em um modelo de domínio, como resultado de uma operação do sistema

- Contratos de operação podem ser considerados parte do modelo de casos de uso, porque fornecem mais detalhes de análise sobre o efeito das operações do sistema implícito nos casos de uso

Modelagem de Interação

Contratos

- Um evento descrito no caso de uso normalmente não mostra tudo que acontece no sistema.
- Para a execução de um evento descrito no caso de uso pode ser necessária a colaboração de várias operações.



Modelagem de Interação

- Um contrato é um documento que descreve os compromissos de uma operação
  - Estilo declarativo
  - Pré e pós-condições de mudanças de estado
  - Para métodos, classes, ou operações gerais de sistema

- Contratos para operações podem ajudar a definir o comportamento do sistema descrevendo os resultados da execução de operações do sistema em termos de mudança de estado dos objetos do domínio.

Modelagem de Interação

Modelagem Funcional

Especificação das funções externas do sistema
- Operações de Sistema – inputs
- Consultas de Sistema – outputs

Artefatos necessários
- Modelo conceitual
- Diagramas de sequência ou casos de uso expandidos

Modelagem de Interação

Contrato de Operação de Sistema

Um contrato de operação de sistema pode ter até três seções:

- Pré-condições (opcional)
  - ✓ Estabelecem o que é assumido como verdade pela operação e que, portanto, não será verificado por ela.

- Pós-condições (obrigatório)
  - ✓ Estabelece como a operação muda a informação existente se for executado com sucesso.

- Exceções (opcional)
  - ✓ Estabelecem quais condições que poderiam evitar que o comando tivesse sucesso as quais serão verificadas por ela.

Modelagem de Interação

Exceções

- Eventos que, se ocorrerem, impedem o prosseguimento correto da operação.
- Usualmente não podem ser garantidas a priori.
- Serão testadas durante a execução.

Modelagem de Interação

Pré-condições

- Definem o que deve ser verdadeiro na estrutura da informação armazenada para que a operação ou consulta possa ser executada.
- Elas não serão testadas durante a execução.
- Algum mecanismo externo deverá garantir sua validade antes de habilitar a execução da operação ou consulta de sistema correspondente.

Modelagem de Interação

Tipos de Pré-condições

- *Garantia de parâmetros*: pré-condições que garantem que os parâmetros da operação ou consulta correspondem a elementos válidos do sistema de informação

- *Restrição complementar*: pré-condições que garantem que a informação se encontra em uma determinada situação desejada

Modelagem de Interação

Pré-condição de garantia de parâmetros é semântica

- Verificações sintáticas são feitas por tipagem.
- Ex.: Ao invés de escrever “x deve ser maior do que zero”, usar x:InteiroPositivo na declaração do parâmetro.
- Uma pré-condição é semântica se para testá-la for necessário consultar informações gerenciadas pelo sistema.

Modelagem de Interação

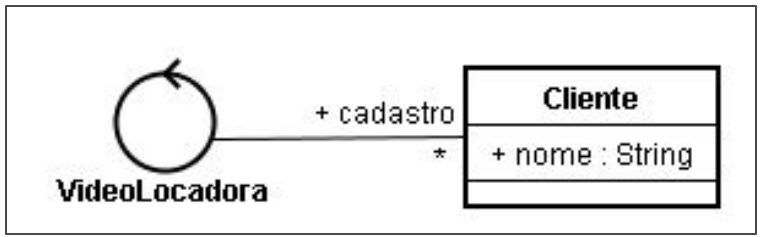
Pré-condições - Garantia de parâmetros

Classe Videolocadora

operação: identificaCliente(nomeCliente:String)

pré:

Existe uma instância da classe *Cliente* tal que o atributo *nome* desta instância é igual ao parâmetro *nomeCliente*.



Modelagem de Interação

Em um contexto não ambíguo

- é possível simplificar a escrita da pré-condição

Classe Videolocadora

operação: identificaCliente(nomeCliente:String)

pré:

Existe um *Cliente* cujo *nome* é igual a *nomeCliente*.

- self.cadastro→exists(nome=nomeCliente)

Modelagem de Interação

Pré-condições - Restrição complementar

- exemplo: se o modelo conceitual especifica que uma associação tem multiplicidade de papel 0..1, uma pré-condição complementar poderá especificar que, para uma instância específica, a associação efetivamente existe (ou não existe)

- uma pré-condição nunca poderá contradizer as especificações do modelo conceitual, apenas complementá-las

Modelagem de Interação

Tipos de restrições complementares

- Existe (ou não existe) uma instância (ou um conjunto de instâncias) com determinadas propriedades.
- Todas as instâncias (ou nenhuma das instâncias) de uma determinada classe (ou um conjunto definido por uma associação) têm determinadas propriedades.
- Uma associação não obrigatória (com multiplicidade de papel 0..1 ou *) existe (ou não existe) entre determinadas instâncias.
- Um determinado atributo de uma instância tem um certo valor.

Modelagem de Interação

Exemplo

Alias: cliente = self.cadastro→select(nome=nomeCliente)
Pré:
cliente→size() == 1
cliente.debito == 0

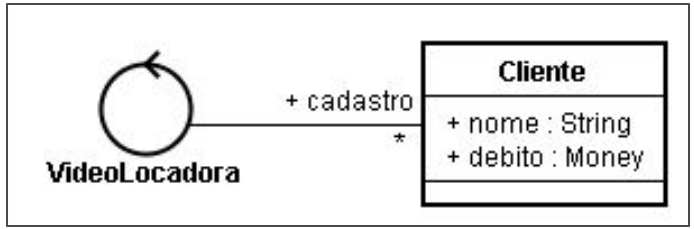
Classe Videolocadora

operação: identificaCliente(nomeCliente:String)

pré:

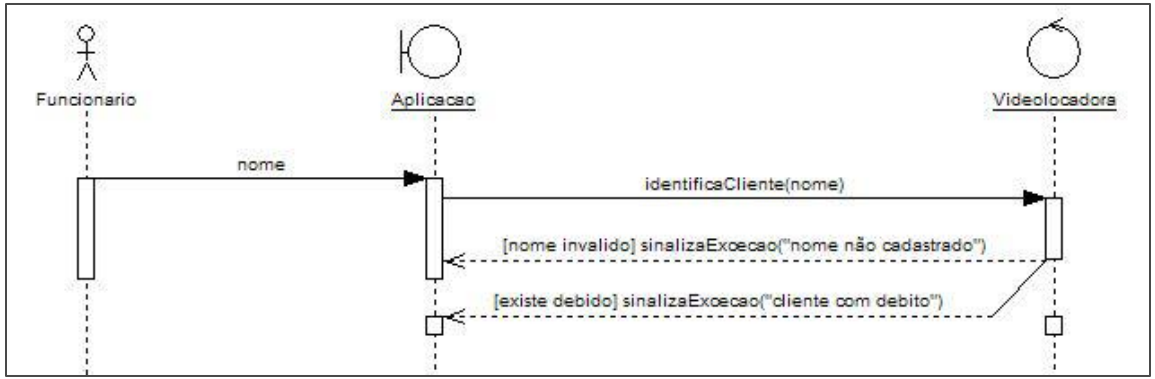
Existe um *Cliente* cujo *nome* é igual a *nomeCliente*.

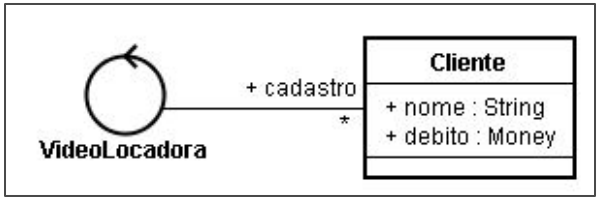
Este *Cliente* possui *débito* igual a zero.



Modelagem de Interação

Diagrama de sequência com exceções





Operação: identificaCliente(nome:String)
Alias: cliente = self.cadastro→select(nome=nomeCliente)
Pré: -
Exceções:
"Nome invalido" se cliente→size == 0
"Cliente com debito" se cliente.debito != 0

Modelagem de Interação

Pós-condições

- Pós-condições devem ser expressas no passado, enfatizando mudanças de estado já ocorridas.
- Devem ser declarativas.
- Expressa no contexto de objetos do modelo de domínio.
- Orientada por mudança de estados.
  - não por ações
  - pós-condições são declarações sobre estados ou resultados
  - não uma descrição de ações ou um projeto de solução

Modelagem de Interação

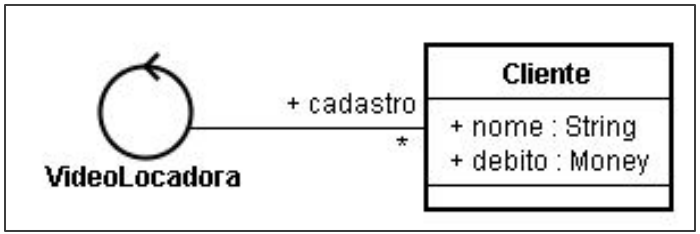
Pós-condições semânticas

- Instância: criação e destruição.
- Associação: criação e destruição.
- Atributo: modificação de valor.

Modelagem de Interação

Criação de uma instância e sua associação com outra instância preexistente

Pós: foi criado um Cliente e associado à Videolocadora



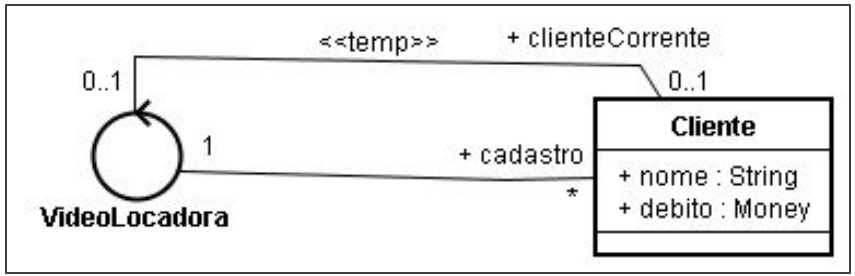
Pós:

cliente = Cliente.new()
self.addToCadastro(cliente).

Modelagem de Interação

Criação de uma associação entre duas instâncias

Pós: O cliente cujo nome é nomeCliente foi associado à VideoLocadora como clienteCorrente

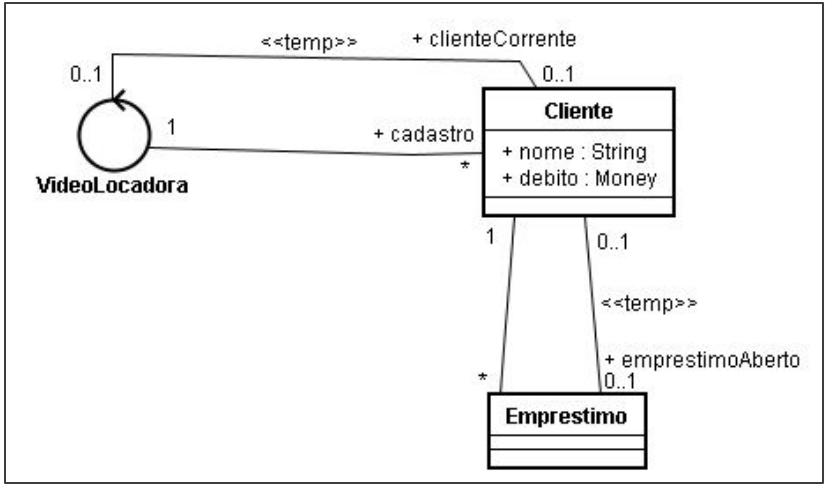


Alias: cliente = self.cadastro→select(nome=nomeCliente)
Pós: self.clienteCorrente = cliente

Modelagem de Interação

Pós-condição condicional

Pós: se não havia nenhum emprestimoAberto associado ao clienteCorrente, então um novo Emprestimo foi criado e associado ao clienteCorrente como emprestimoAberto.

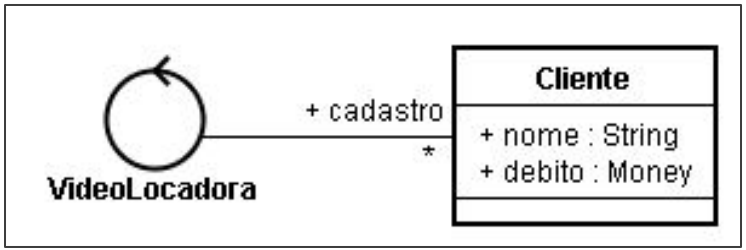


Pós: self.clienteCorrente.emprestimoAberto@pre→size==0 IMPLIES self.clienteCorrente.emprestimoAberto = Emprestimo.new()

Modelagem de Interação

Pós-condições - Destruição de uma instância

Pós: “foi destruído um Cliente cujo nome é igual a nomeCliente”.



Presume-se que quando uma instância é destruída, todas as associações ligadas a ela também o sejam.

Deve-se tomar cuidado com questões estruturais (associações obrigatórias) quando um objeto é destruído.

Modelagem de Interação

Contrato para Inserção

Classe Videolocadora

operação: cadastraCliente(nomeC,enderecoC,telefoneC:String)

pré:

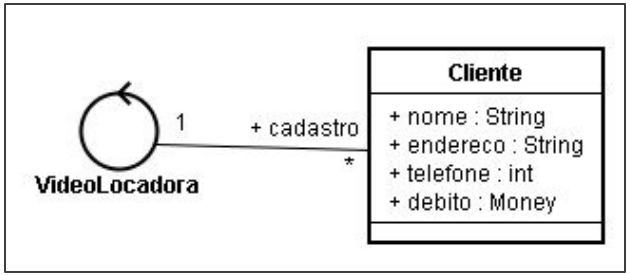
Não existe nenhum *Cliente* com *nome* = *nomeC*.

pós:

Foi criado um *Cliente* e adicionado ao *cadastro*.

Os atributos *nome*, *endereco* e *telefone* do *Cliente* foram alterados para *nomeC*, *enderecoC* e *telefoneC*.

O atributo *debito* do *Cliente* foi alterado para 0,00.



Modelagem de Interação

Em OCL

Classe Videolocadora

operação: cadastraCliente(nomeC,enderecoC,telefoneC:String)

pré:

self.cadastro→select(nome=nomeC)→size==0

pós:

cliente = Cliente.new()

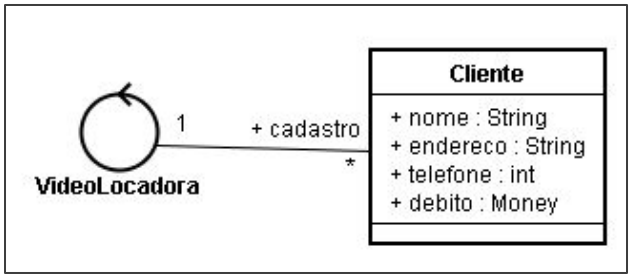
self.addToCadastro(cliente)

cliente.nome = nomeC

cliente.endereco = enderecoC

cliente.telefone = telefoneC

cliente.debito = 0



Modelagem de Interação

Quiz

Que cuidado deve-se tomar quando um contrato possui uma pós-condição do tipo “foi criada uma instância”?

a) Deve-se inserir a instância em uma lista ou conjunto de instâncias da classe.

b) Deve-se colocar uma pré-condição para verificar se já não existe uma instância alocada para a mesma variável.

c) Deve-se garantir que algum outro contrato terá uma pós-condição do tipo “foi destruída uma instância”.

d) Deve-se garantir que no mesmo contrato exista uma pré-condição do tipo “existe uma instância de...”.

e) Deve-se sempre adicionar uma pós-condição do tipo “foi criada uma associação” entre a instância recém criada e alguma outra instância pré-existente.

Modelagem de Interação

Quiz

Que cuidado deve-se tomar quando um contrato possui uma pós-condição do tipo “foi criada uma instância”?

a) Deve-se inserir a instância em uma lista ou conjunto de instâncias da classe.

b) Deve-se colocar uma pré-condição para verificar se já não existe uma instância alocada para a mesma variável.

c) Deve-se garantir que algum outro contrato terá uma pós-condição do tipo “foi destruída uma instância”.

d) Deve-se garantir que no mesmo contrato exista uma pré-condição do tipo “existe uma instância de...”.

e) Deve-se sempre adicionar uma pós-condição do tipo “foi criada uma associação” entre a instância recém criada e alguma outra instância pré-existente.

Modelagem de Interação

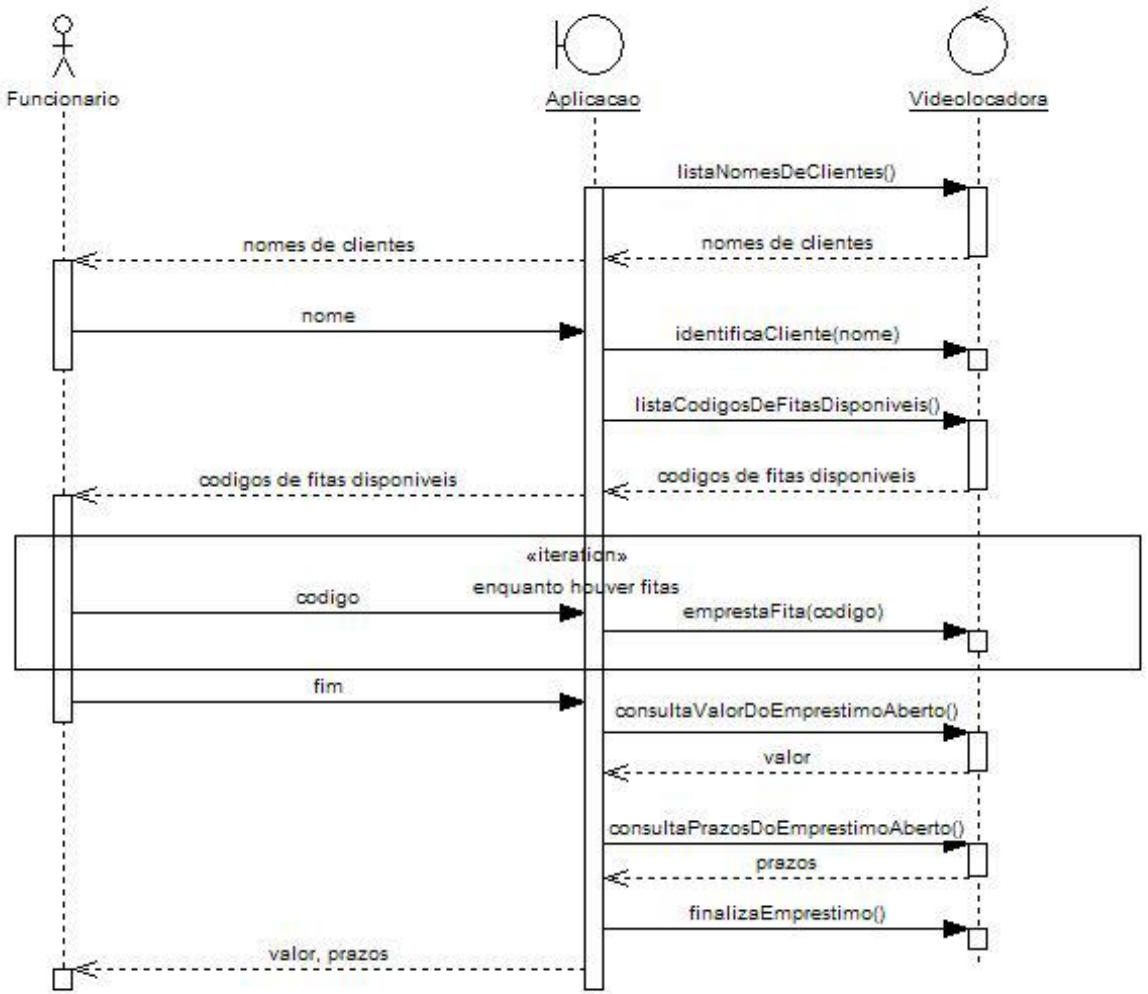
Outras Consultas e Operações (específicas dos casos de uso)

- Frequentemente haverá uma cadeia de execução ao longo de um dos fluxos, explicitada no diagrama de sequência.

- Verificar:
  - Qual é o objetivo de cada operação?
  - O que cada uma delas espera que tenha sido produzido pelas anteriores?
  - O que cada uma delas produz?
  - Que exceções poderiam ocorrer durante a execução?

Modelagem de Interação

Exemplo:



Modelagem de Interação

Exemplo : Contrato operação indentificaCliente

Classe Videolocadora

operação: identificaCliente(nomeC:String)

alias:

cliente = self.cadastro→select(nome=nomeC)

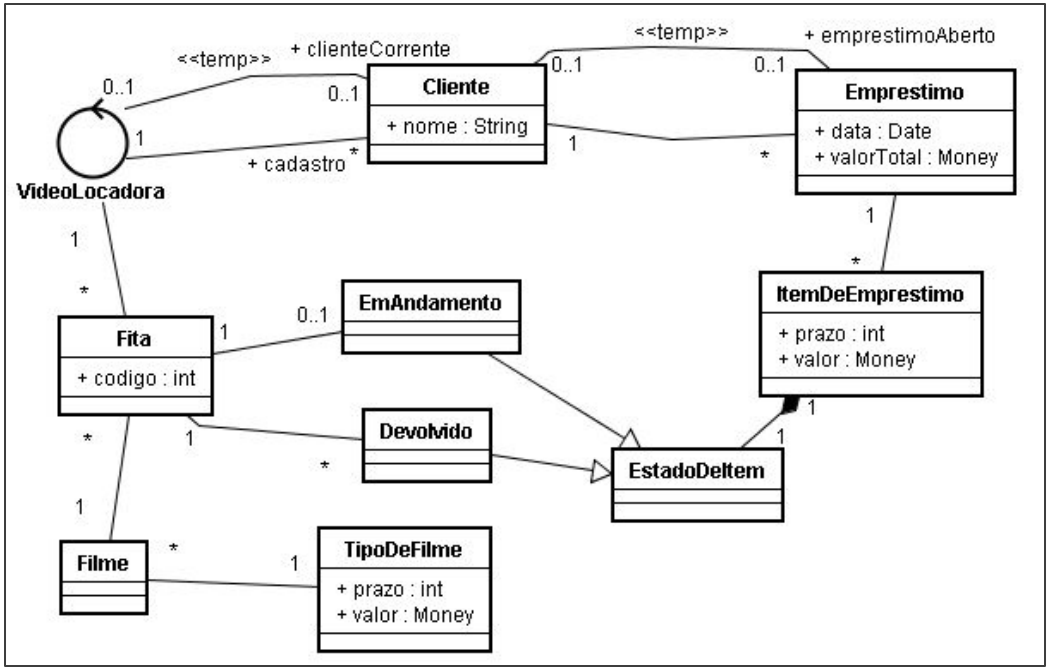
pré:

cliente→size() == 1

self.clienteCorrente→size() == 0

pós:

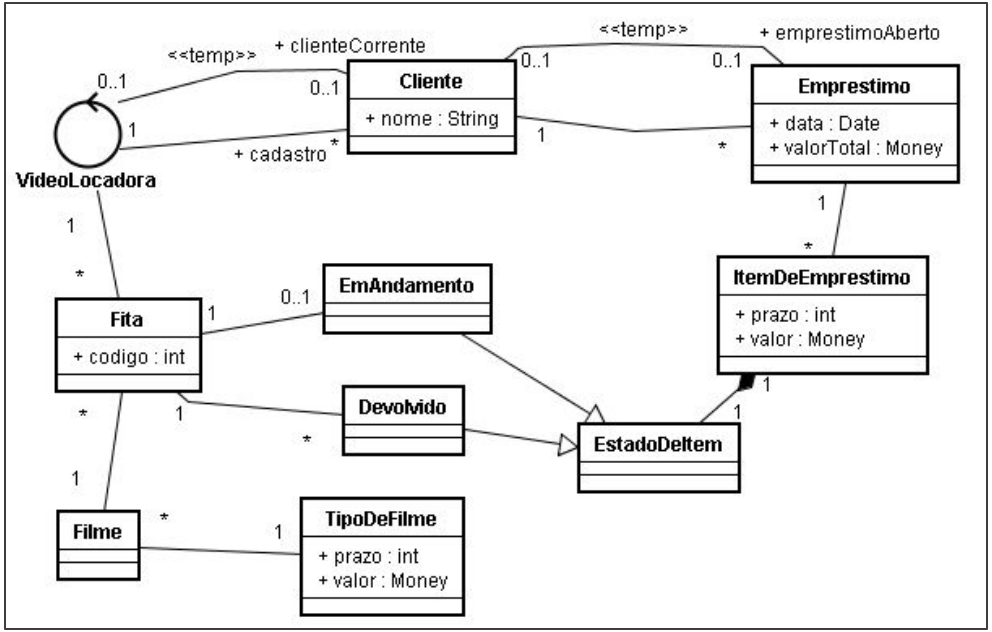
self.setClienteCorrente(cliente)



Modelagem de Interação

Exemplo : Contrato operação emprestarFita

Classe Videolocadora
operação: emprestaFita(codigoF:String)
alias:
fita = self.fitas→select(codigo=codigoF)
pré:
self.clienteCorrente→size() == 1
fita→size() == 1
pós:
self.clienteCorrente.emprestimoAberto→size() == 0 IMPLIES
emprestimo = Emprestimo.new()
emprestimo.data = today()
emprestimo.valorTotal = 0
self.clienteCorrente.setEmprestimoAberto(emprestimo)
item = ItemDeEmprestimo.new()
emprestimo.addItem(item)
estado = EmAndamento.new()
item.addEstado(estado)
estado.setFita(fita)
item.setPrazo(fita.filme.tipoDeFilme.prazo)
item.setValor(fita.filme.tipoDeFilme.valor)
emprestimo.setValorTotal(emprestimo.valorTotal@pre + item.valor)



Modelagem de Interação

Quando os Contratos são úteis

Faça os contratos quando:

- Muitos objetos do domínio são atualizados, criados ou associados em um único passo do fluxo do caso de uso.
- Não é claro pela descrição do caso de uso quais atributos devem ser atualizados, ou quais associações devem ser feitas ou desfeitas.

Não faça um contrato para cada operação do sistema.

Casos de uso em geral são suficientes para descrever os detalhes necessários para se derivar o projeto - Criação e remoção de instância.

Modelagem de Interação

Como fazer um Contrato

Regras úteis:

1. Identificar operações de sistema a partir dos diagramas de sequência.
2. Para cada operação, construir um contrato.
3. Começar escrevendo a seção Responsabilidades, descrevendo informalmente o propósito da operação.
4. Completar a seção Pós-condições, descrevendo declarativamente as mudanças de estado que ocorrem aos objetos do modelo conceitual:
    - ✓ Criação e remoção de instância
    - ✓ Modificação de atributo
    - ✓ Formação e quebra de associações (erro mais comum!)

Modelagem de Interação

Como fazer um Contrato

Regras úteis (cont.):

5. Completar a seção Pré-condições, descrevendo as pré-suposições sobre o estado do sistema no início da operação:

- ✓ Coisas que devem ser testadas pelo sistema em algum ponto durante a execução da operação.
- ✓ Coisas que não são testadas, mas sobre as quais depende fortemente o sucesso da operação.

Modelagem de Interação

Quiz

Assinale a opção que representa uma pré-condição de contrato inconsistente com o modelo conceitual ao lado.

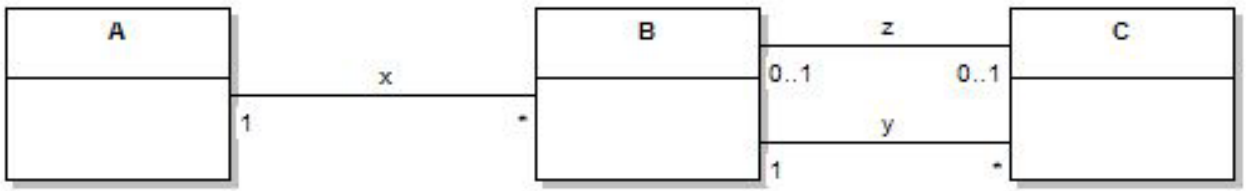
a) Existe pelo menos uma instância de B associada a uma instância de A por x.

b) Existe uma instância de C que não está associada a nenhuma instância de B por y.

c) Existe uma instância de C que não está associada a nenhuma instância de B por z.

d) Não existe nenhuma instância de B.

e) Existe uma instância de A associada a dez instâncias de B por x.



Modelagem de Interação

Quiz

Sobre contratos de operação de sistema, pode-se afirmar que:

a) Cada contrato define o que deve ser verdadeiro antes de a operação ser executada através de precondições, portanto, um contrato de operação de sistema não pode prever exceções.

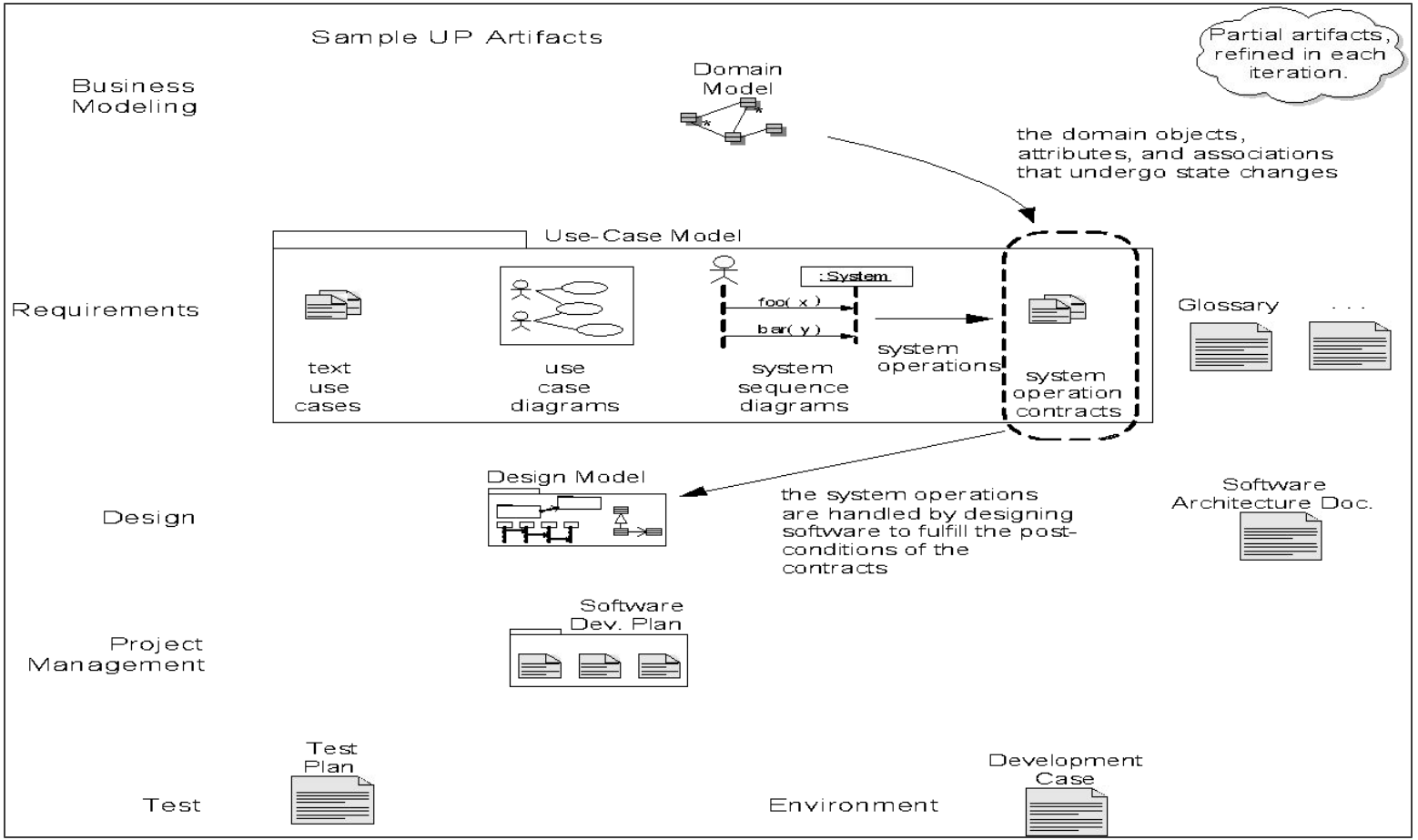
b) É feito um contrato para cada caso de uso, indicando o que ele produz como resultado para os atores .

c) São feitos para cada operação de sistema. Contém a especificação do algoritmo que realiza a operação.

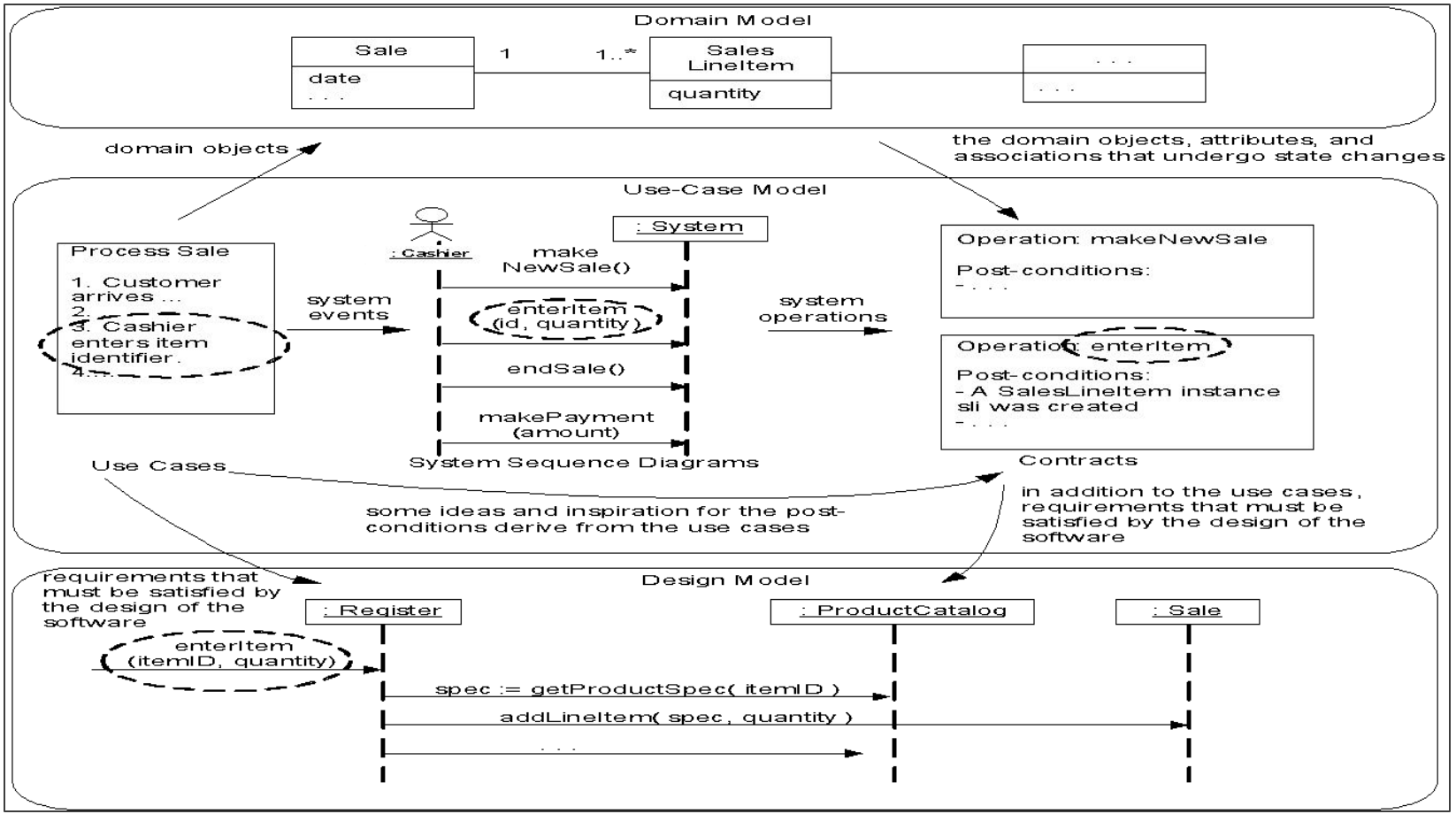
d) São celebrados entre o desenvolvedor e seus clientes para definir o cronograma do desenvolvimento do software e seus custos.

e) São feitos para cada operação de sistema. Podem conter precondições e contém necessariamente pós- condição.

Contratos e Outros Artefatos



Contratos e Outros Artefatos: Relacionamento entre os artefatos da UML



Contrato – documentação importante

Projeto de Software

Referências básicas:

- **ACM TRANSACTIONS ON SOFTWARE ENGINEERING AND METHODOLOGY**. New York, N.Y., USA: Association for Computing Machinery, 1992-. Trimestral. ISSN 1049-331X. Disponível em: https://dl.acm.org/toc/tosem/1992/1/2. Acesso em: 19 jul. 2024. (Periódico On-line).

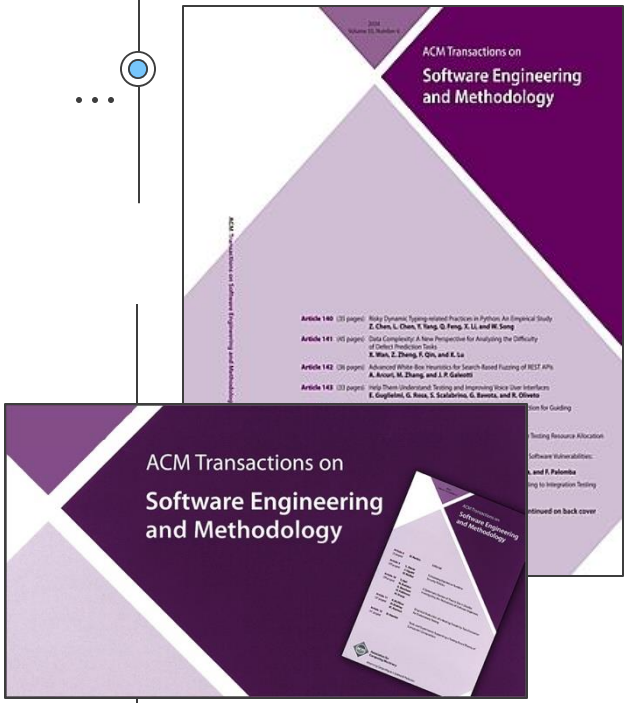
- LARMAN, Craig. **Utilizando UML e padrões**: uma introdução á análise e ao projeto orientados a objetos e desenvolvimento iterativo. 3. ed. Porto Alegre: Bookman, 2007. E-book. ISBN 9788577800476. (Livro Eletrônico).

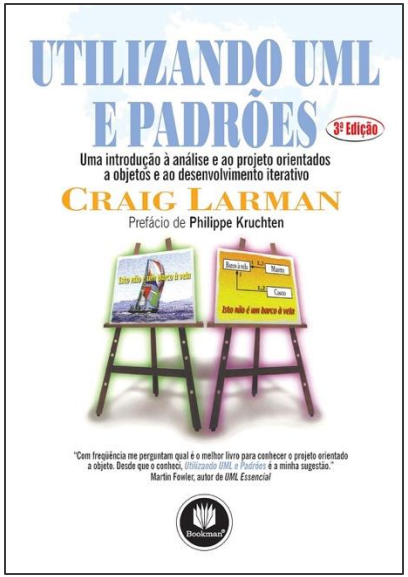
- SILVEIRA, Paulo et al. **Introdução à arquitetura e design de software**: uma visão sobre a plataforma Java. Rio de Janeiro, RJ: Elsevier, Campus, 2012. xvi, 257 p. ISBN 9788535250299. (Disponível no Acervo).

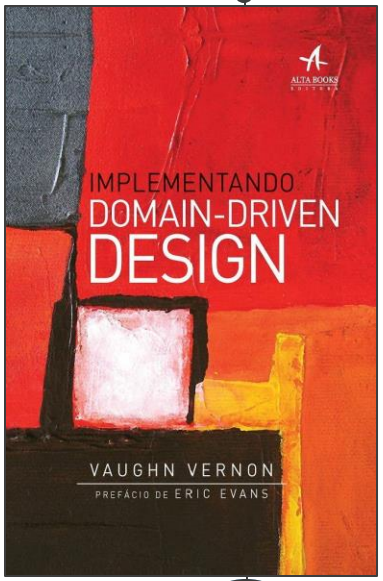
- VERNON, Vaughn. **Implementando o Domain-Driven Design**. Rio de Janeiro, RJ: Alta Books, 2016. 628 p. ISBN 9788576089520. (Disponível no Acervo).

Projeto de Software

Referências básicas:







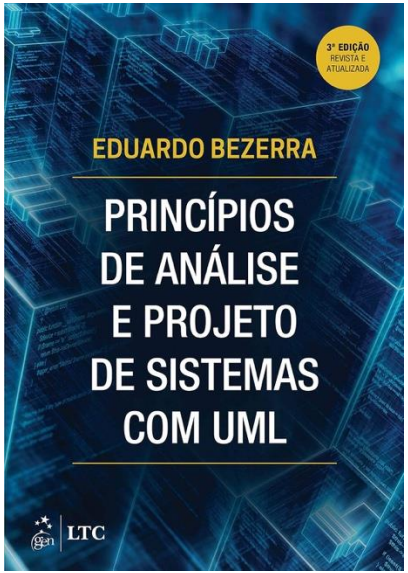
Projeto de Software

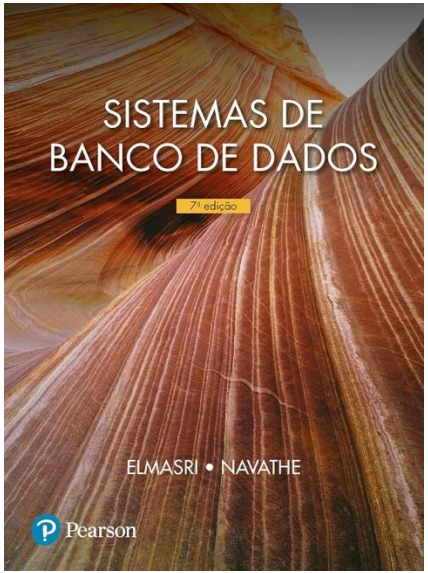
Referências complementares:

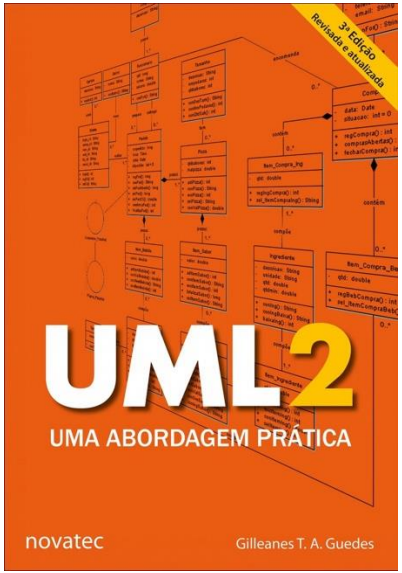
- BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 3. ed. rev. e atual. Rio de Janeiro: Elsevier, 2015. xvii, 398 p. ISBN 9788535226263. (Disponível no Acervo).
- ELMASRI, Ramez; Navathe, Shamkant B. **Sistemas de banco de dados**, 7ª ed. Editora Pearson 1152 ISBN 9788543025001. (Livro Eletrônico).
- GUEDES, Gilleanes T. A. **UML 2**: uma abordagem prática. 2. ed. São Paulo: Novatec, c2011. 484 p. ISBN 9788575222812. (Disponível no Acervo).
- **IEEE TRANSACTIONS ON SOFTWARE ENGINEERING**. New York: IEEE Computer Society,1975-. Mensal,. ISSN 0098-5589. Disponível em: https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=32. Acesso em: 19 jul. 2024. (Periódico On-line).
- SOMMERVILLE, Ian. **Engenharia de software**. 10. ed. São Paulo: Pearson Education do Brasil, c2019. xii, 756 p. ISBN 9788543024974. (Disponível no Acervo).
- WAZLAWICK, Raul Sidnei. **Análise e design orientados a objetos para sistemas de informação**: modelagem com UML, OCL e IFML. 3. ed. Rio de Janeiro, RJ: Elsevier, Campus, c2015. 462 p. ISBN 9788535279849. (Disponível no Acervo).

Projeto de Software

Referências complementares:

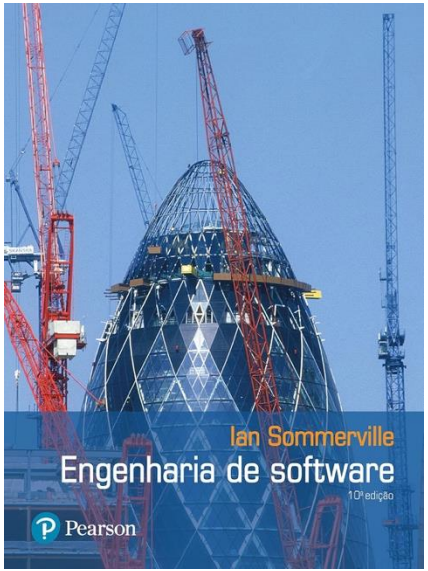






Projeto de Software

Referências complementares:



Obrigado!

Dúvidas?

joaopauloaramuni@gmail.com



GitHub LinkedIn Lattes