

# Engenharia de Software

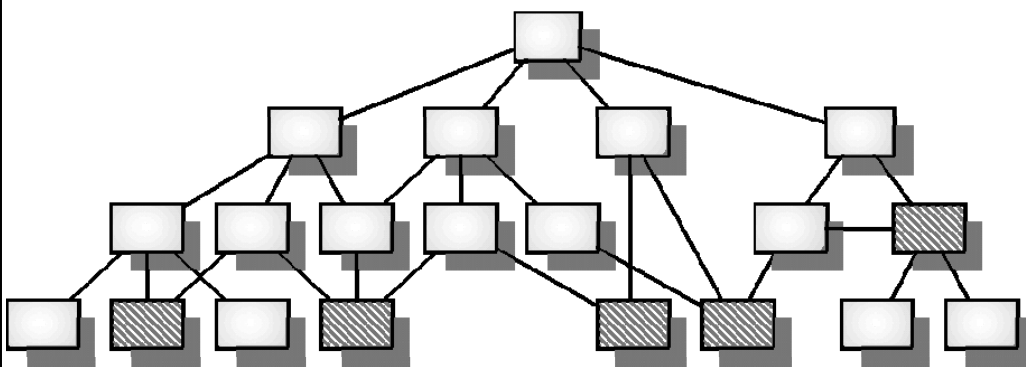
## Projeto (Design)

**Marcello Thiry**  
marcello.thiry@gmail.com



**LQPS**  
<http://www.univali.br/lqps>

## Projeto (Design)



## Objetivos

- ☐ Transformar os requisitos em um projeto que permitirá construir o sistema
- ☐ Nivelar os artefatos existentes com o ambiente de desenvolvimento
- ☐ Identificar os componentes de software e seus relacionamentos



Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

3

## Projeto Orientado a Objetos

- ☐ Explicar como um projeto de software pode ser representado na forma de um conjunto de objetos interagindo entre si
- ☐ Introduzir vários modelos que descrevem um projeto orientado a objetos
- ☐ Mostrar como a UML pode ser usada para representar estes modelos
- ☐ Introduzir padrões de projeto (*design patterns*)



Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

4

## Realização dos casos de uso

(Projeto/Design)

- ☐ O diagrama de caso de uso apresenta uma visão externa do sistema
- ☐ Diagramas de Interação descrevem como os casos de uso são realizados em interações entre sociedades de objetos
- ☐ Como identificar objetos que colaboram em um caso de uso?

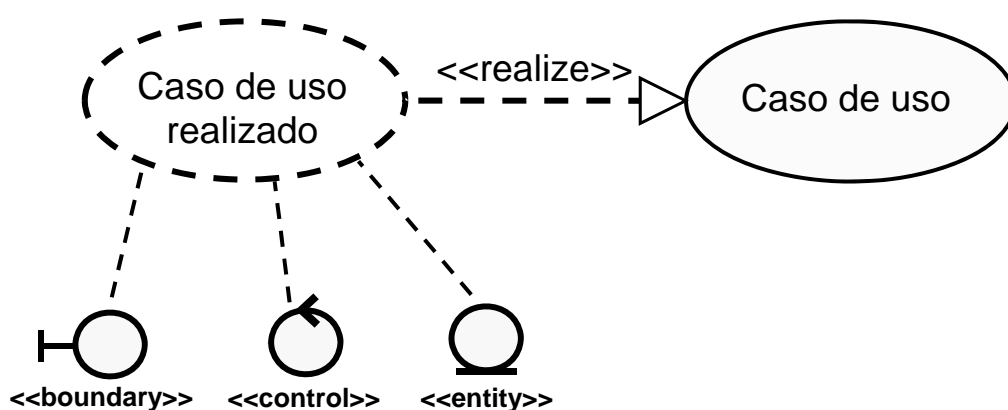


Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

5

## Uma realização na UML: colaboração

(Projeto/Design)



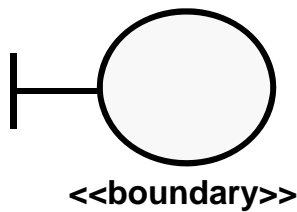
Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

6

## Análise do caso de uso: objetos de limite

(Projeto/Design)

- ☐ Acesso dos atores ao sistema
- ☐ Define o limite entre o sistema e o mundo externo
- ☐ Também conhecidos como objetos de fronteira



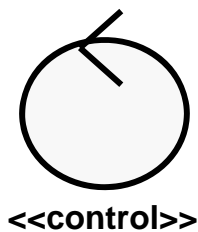
Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

7

## Análise do caso de uso: objetos de controle

(Projeto/Design)

- ☐ Controle em geral
- ☐ Processamento
- ☐ Regras de negócio



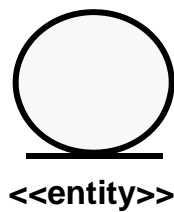
Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

8

## Análise do caso de uso: objetos de entidade

(Projeto/Design)

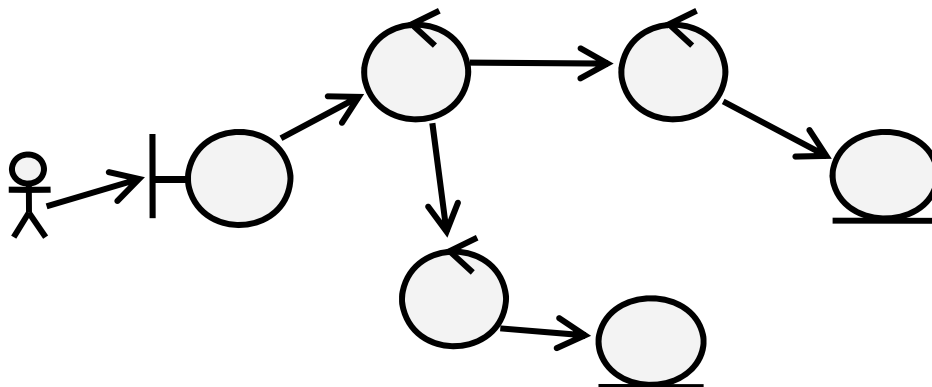
- ❑ Informação em geral
- ❑ Representam, usualmente, objetos persistentes

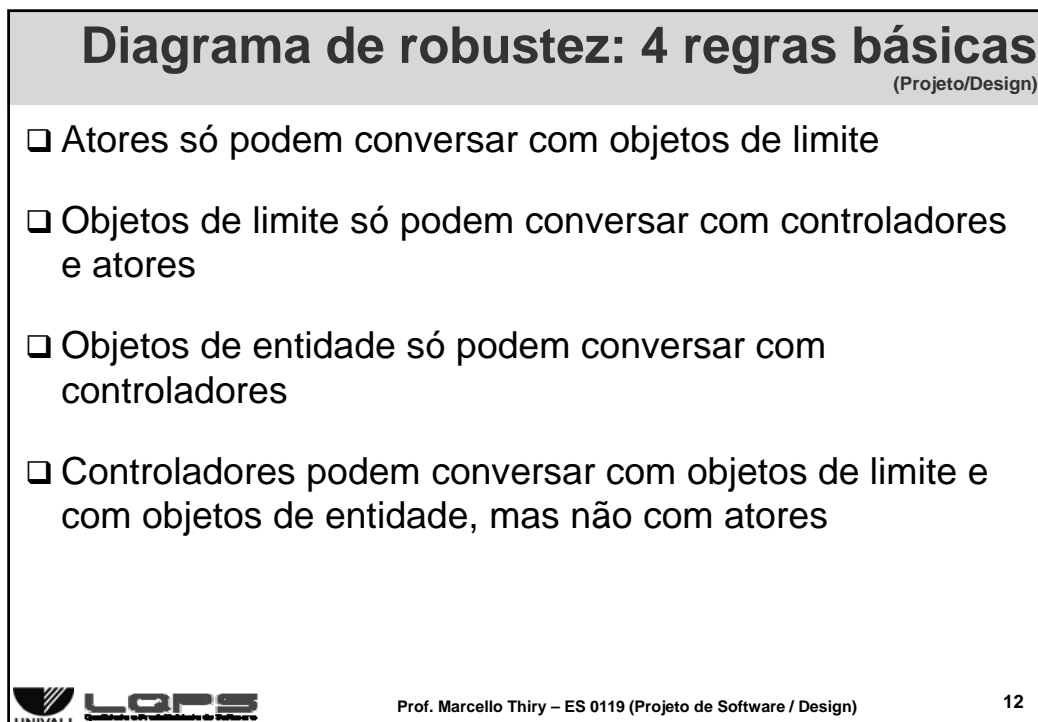
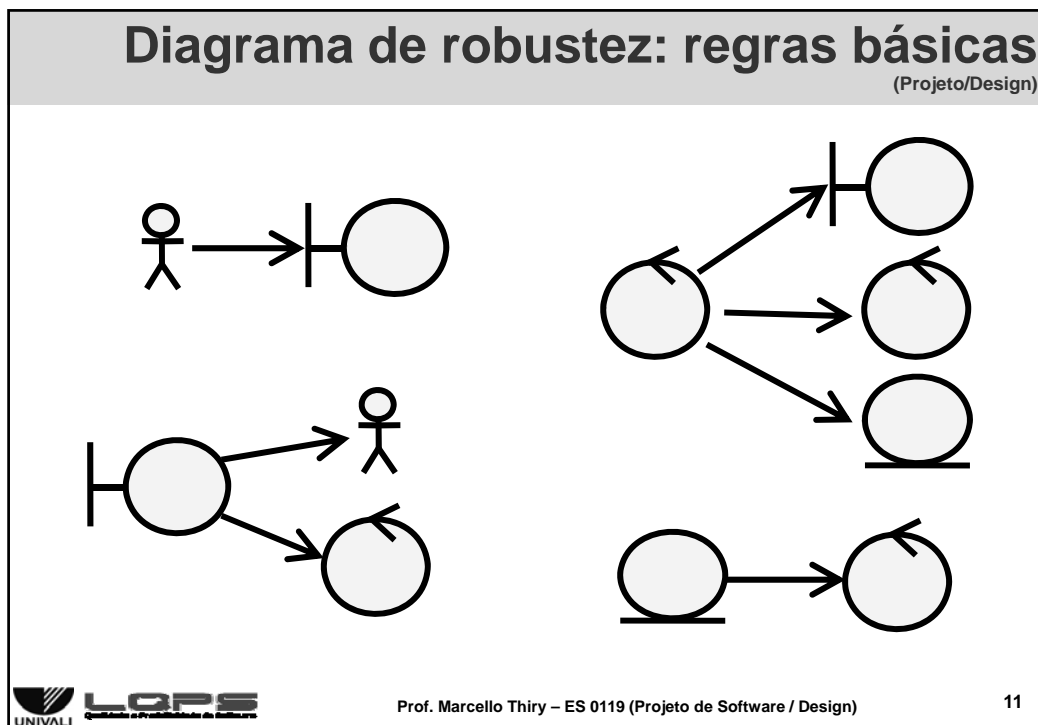


## Diagrama de robustez

(Projeto/Design)

- ❑ Preenche um espaço vazio entre o caso de uso e o diagrama de seqüência; não é um diagrama oficial da UML





## Diagrama de robustez: ICONIX

(Projeto/Design)

- ☐ Objetos de entidade são substantivos
- ☐ Controladores são verbos (ações): usualmente mapeados depois como operações
- ☐ Substantivos não conversam com substantivos, mas verbos podem conversar tanto com outros verbos como com substantivos



Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

13

## Revendo o caso de uso exemplo

(Projeto/Design)

### USC.01 Efetua Login

**Ator ativo:** Operador

**Fluxo base:**

- ☐ O sistema apresenta a página de controle de acesso
- ☐ O operador preenche os dados e confirma
- ☐ O sistema valida a conta e senha fornecidas
- ☐ O sistema busca as permissões do operador
- ☐ O sistema registra esta operação no histórico de operações efetuadas



Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

14

## Análise inicial


(Projeto/Design)

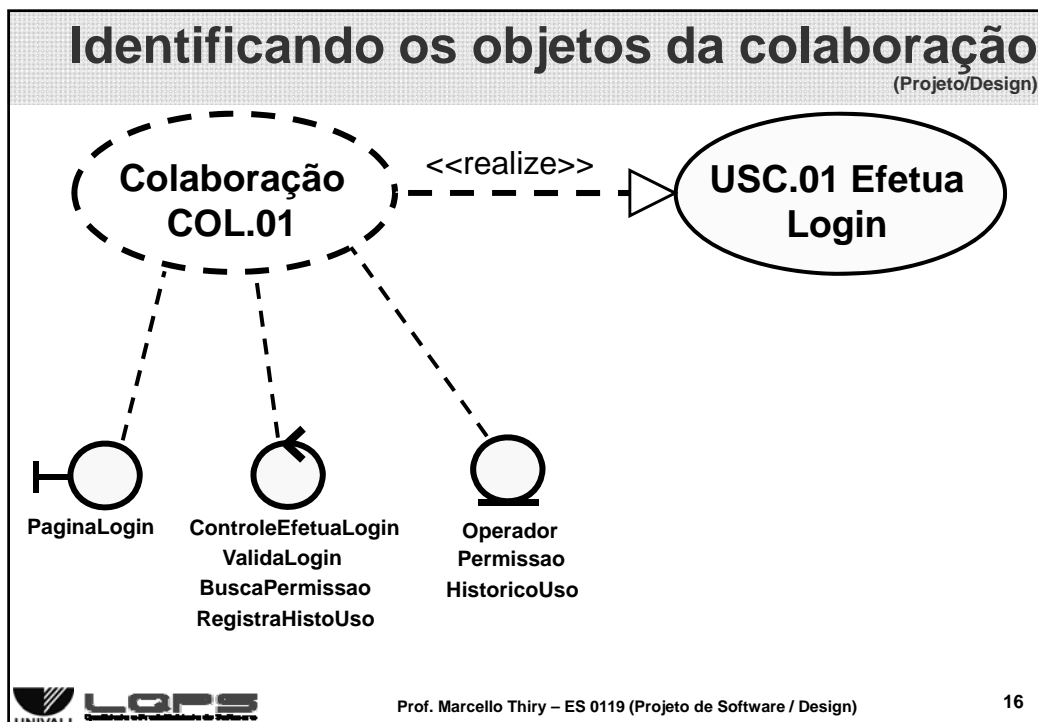
### USC.01 Efetua Login

**Ator ativo:** Operador

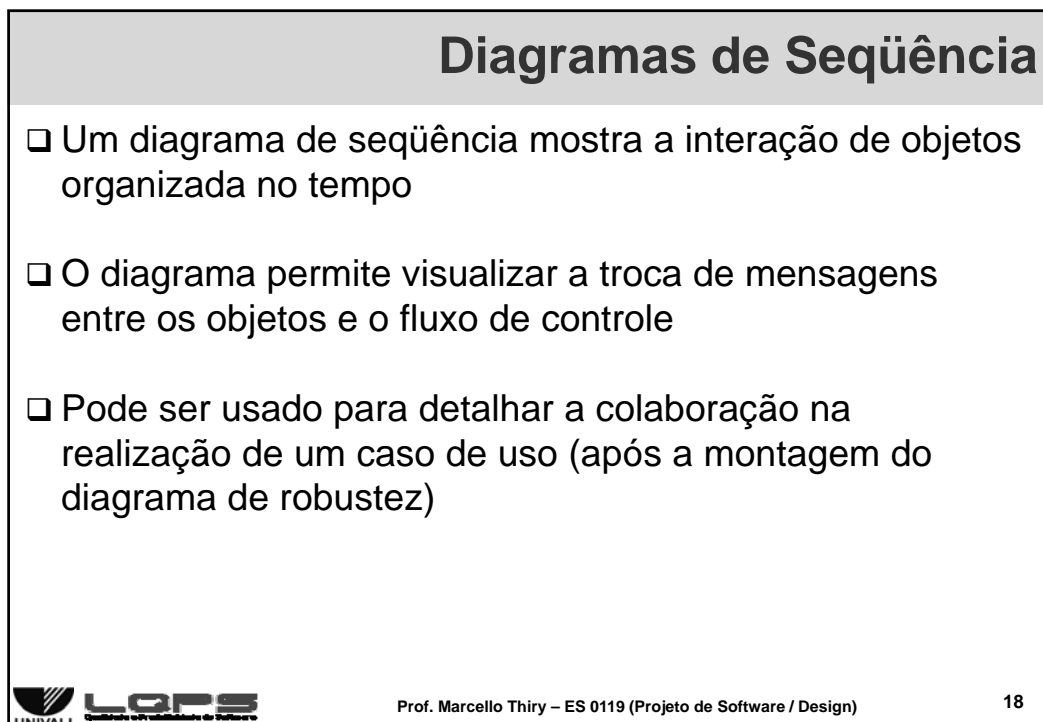
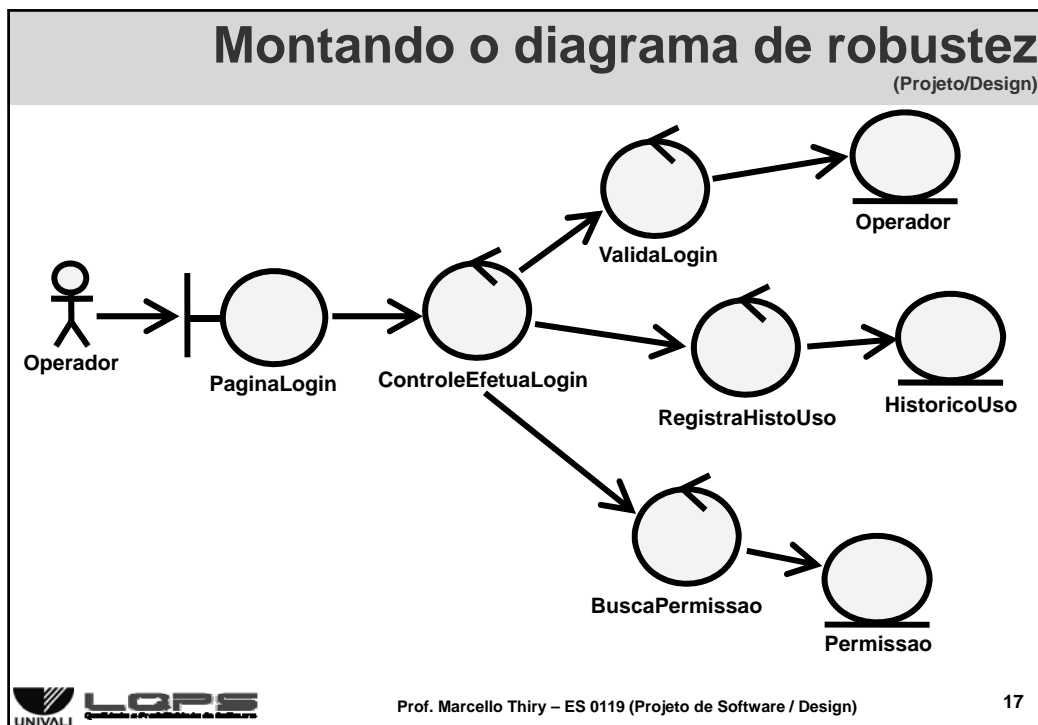
**Fluxo base:**

- ☐ O sistema apresenta a **página de controle de acesso**
- ☐ O operador preenche os dados e confirma
- ☐ O sistema **valida** a **conta e senha** fornecidas
- ☐ O sistema **busca as permissões** do **operador**
- ☐ O sistema **registra** esta operação no **histórico de operações** efetuadas

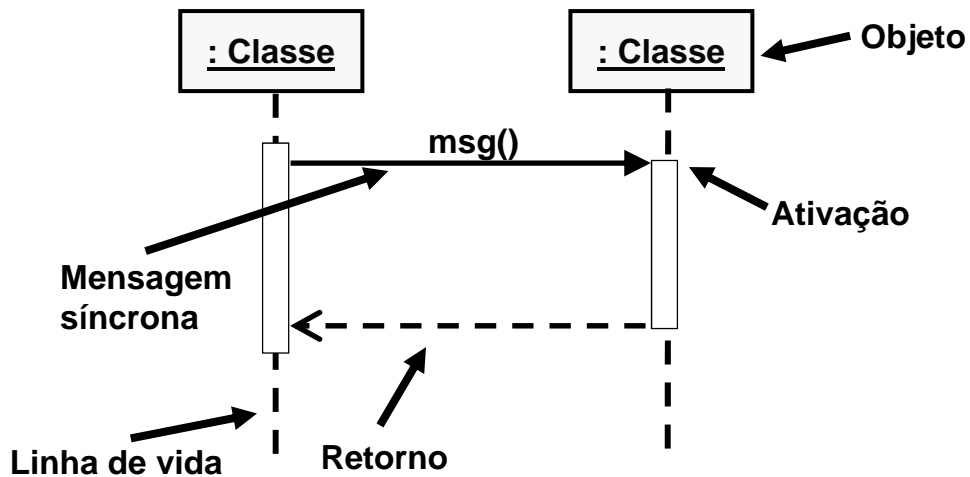
 Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design) 15



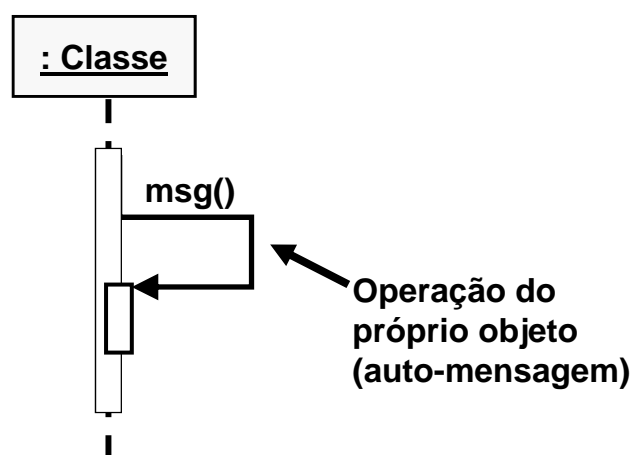




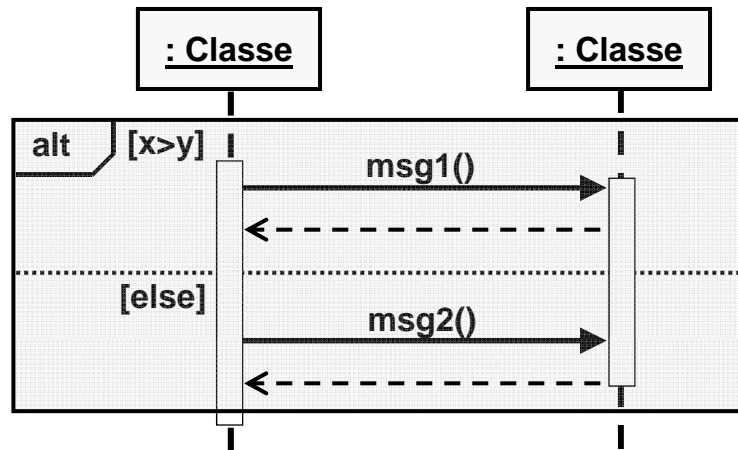
## Elementos do Diagrama de Seqüência



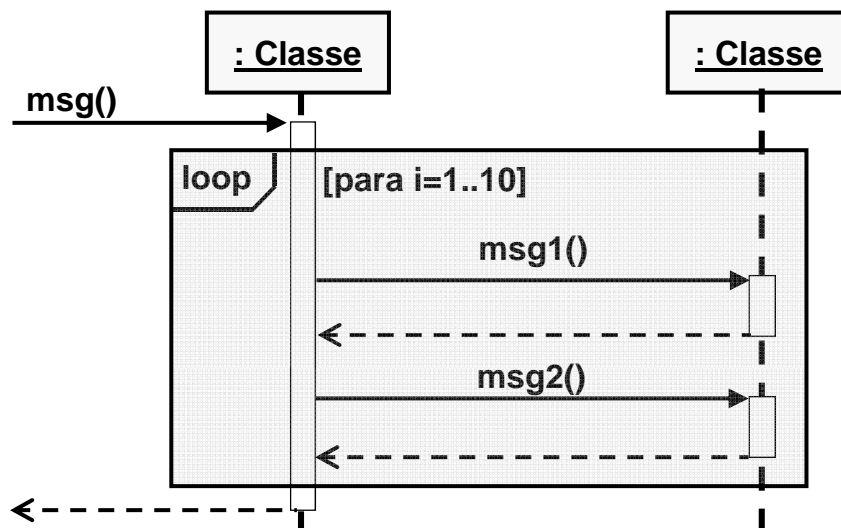
## Mensagens para o próprio objeto

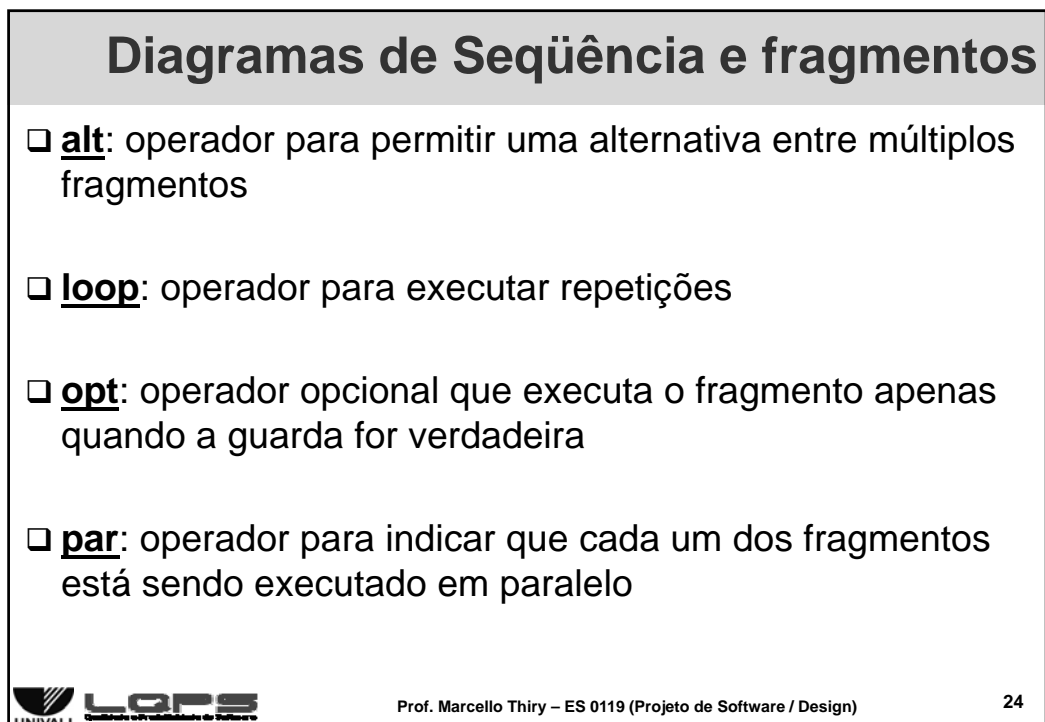
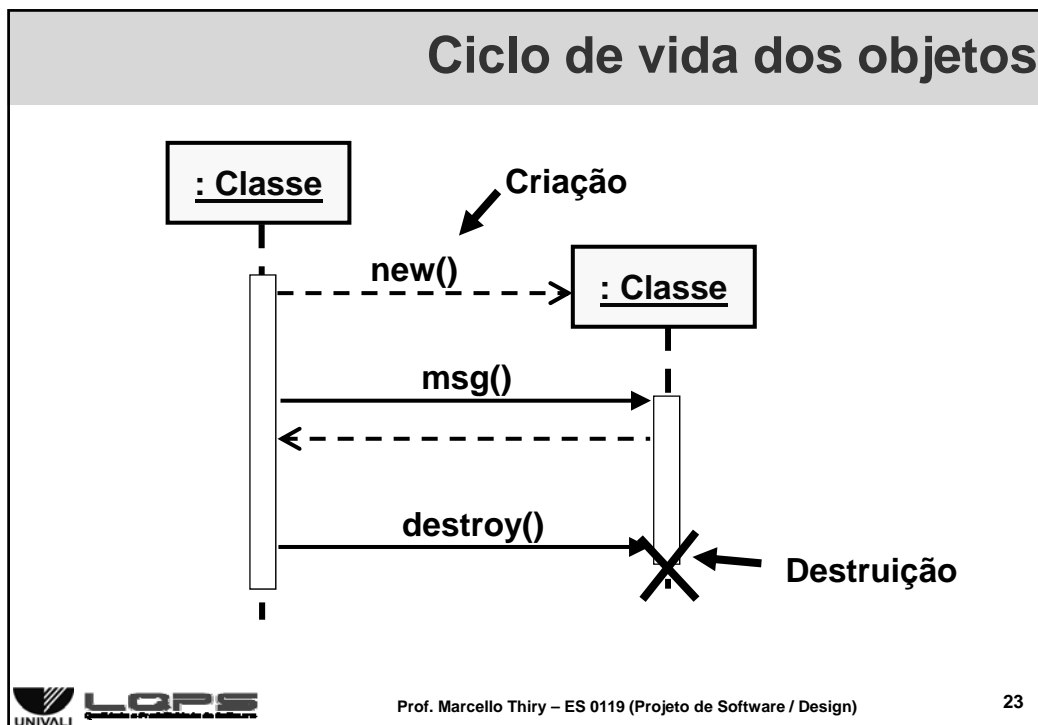


## Seleção de alternativas (alt)

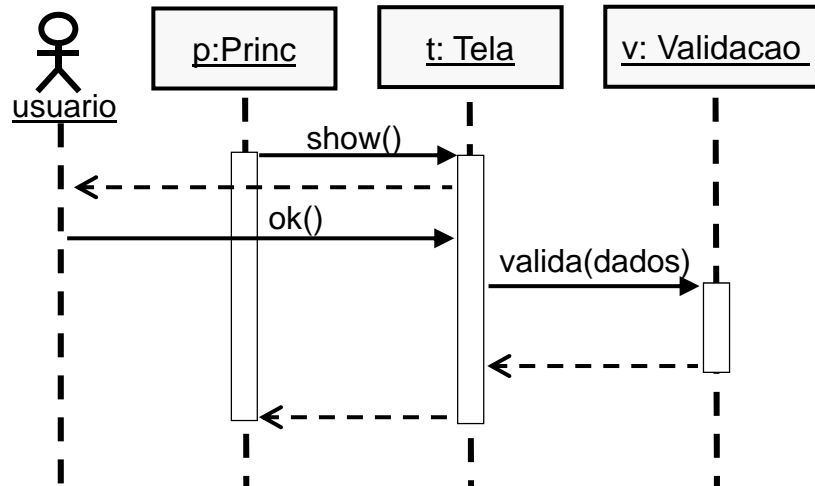


## Repetições (loop)



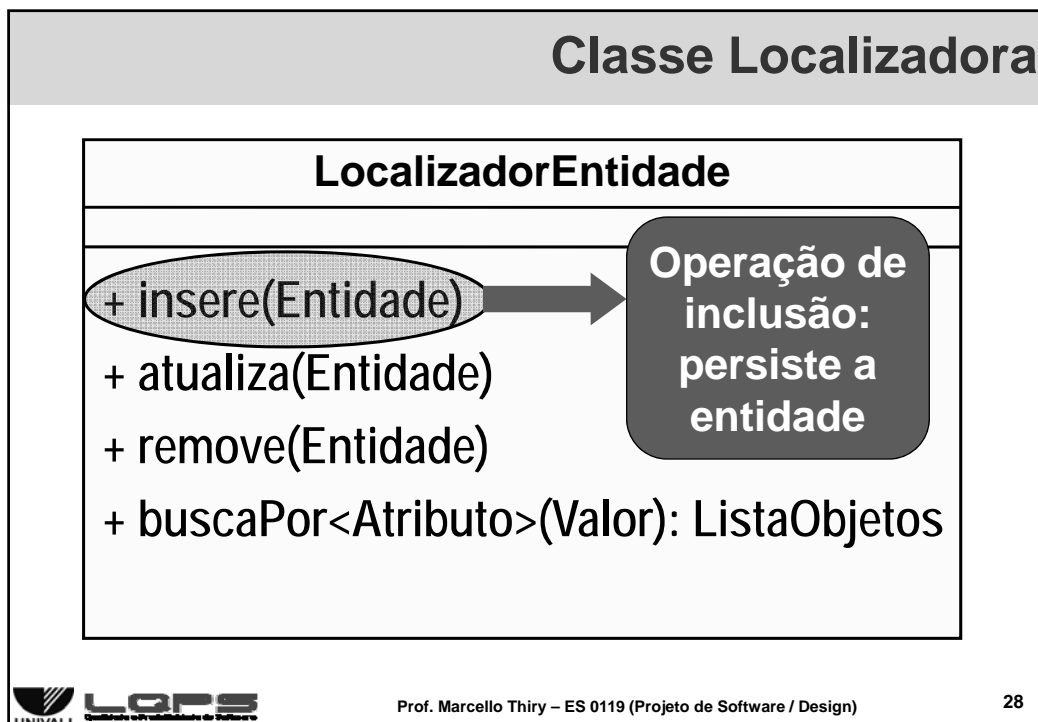
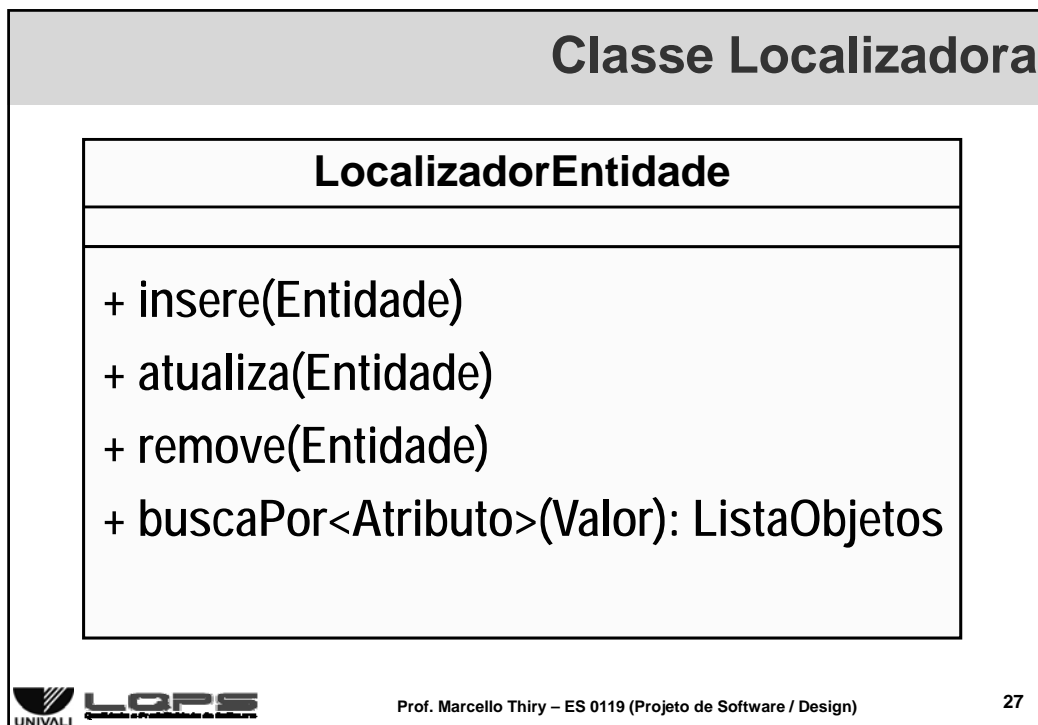


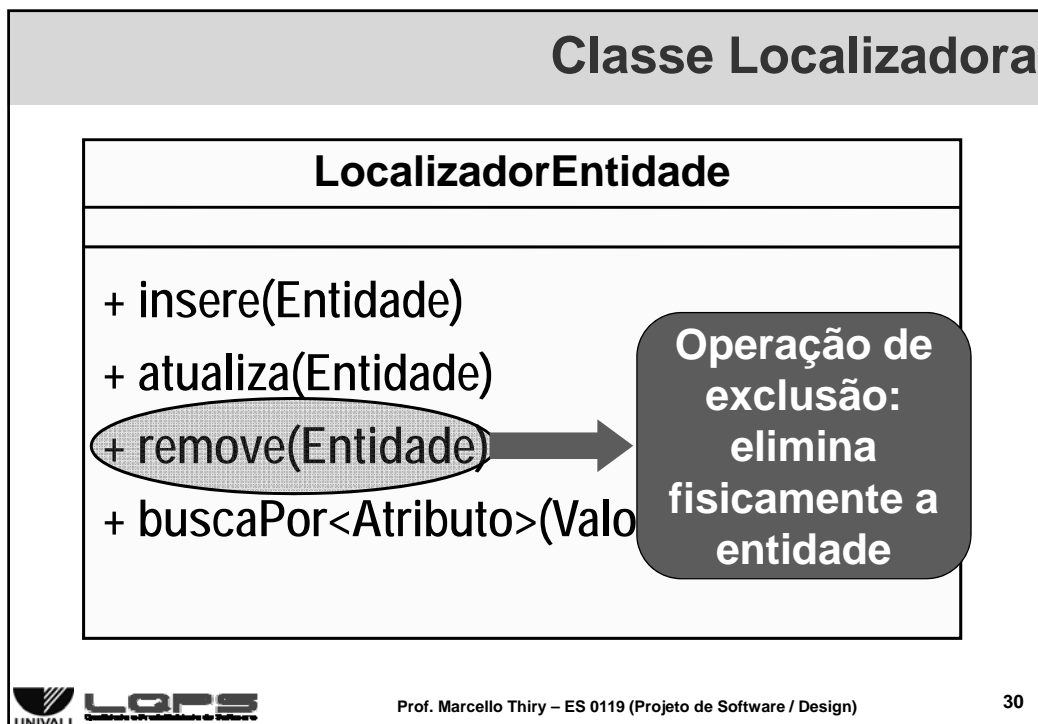
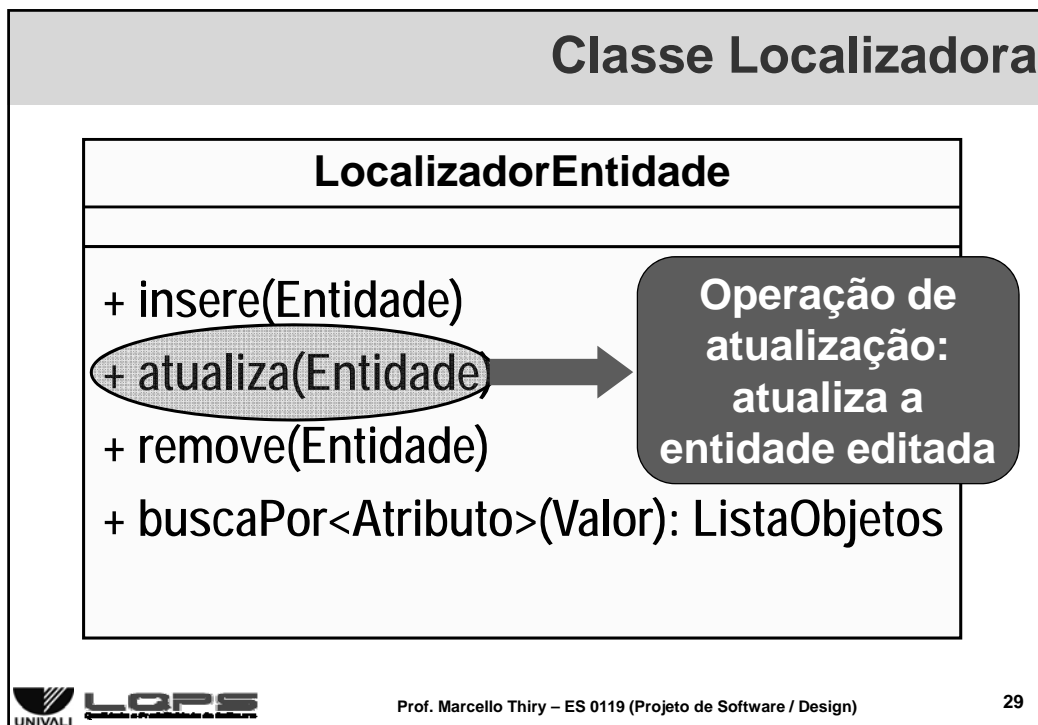
## Diagrama de Seqüência: exemplo

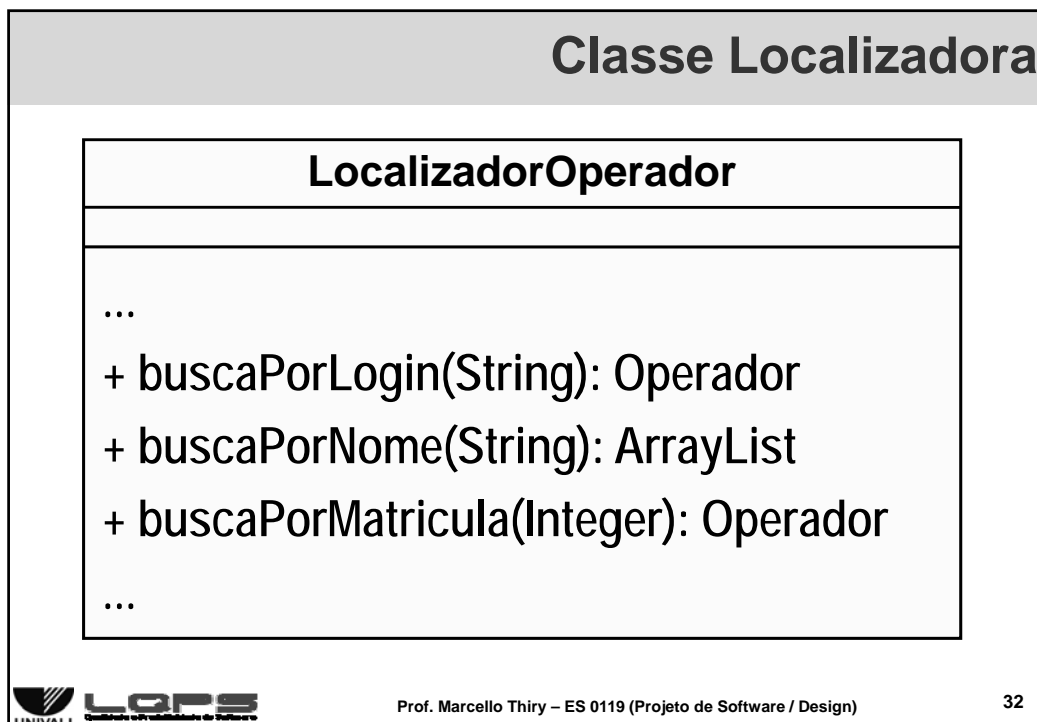
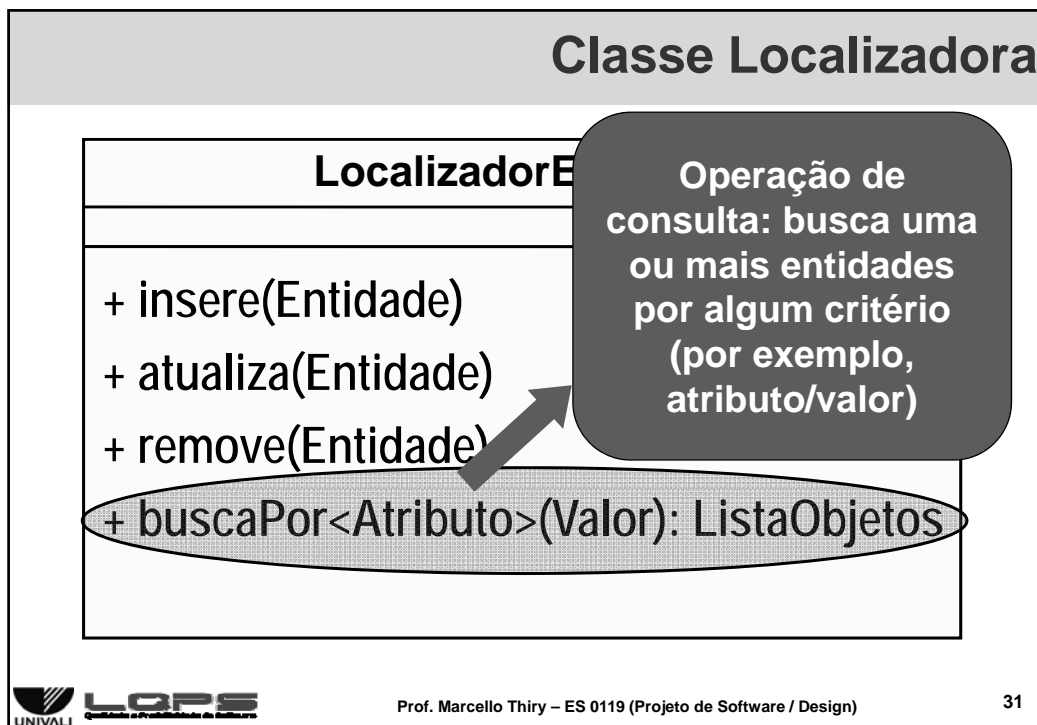


## Padrões de Projeto

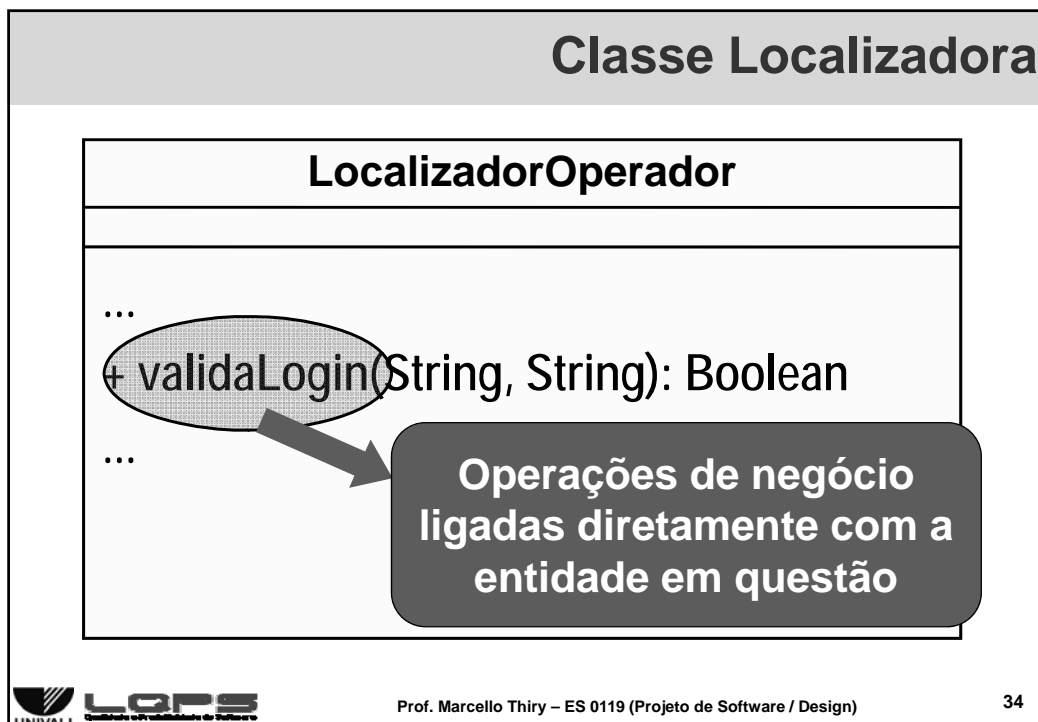
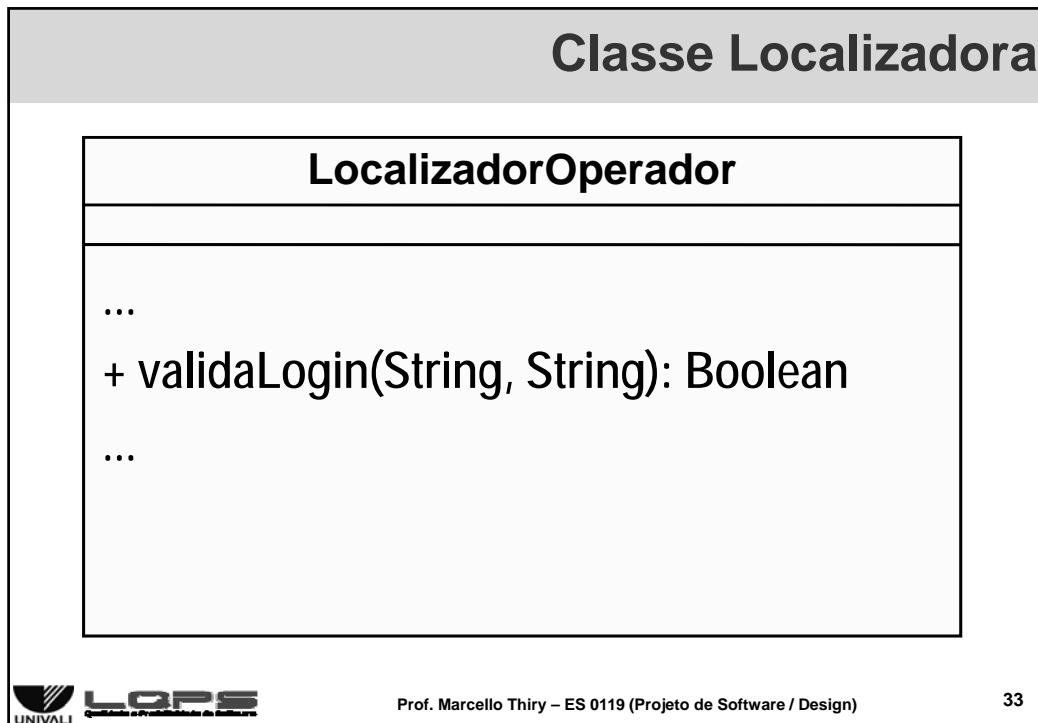
- ❑ Um padrão que fornece uma solução comum para um problema comum
- ❑ Utilização de experiências anteriores que tiveram sucesso comprovado
- ❑ Para cada padrão de projeto, é necessário identificar algumas informações que irão permitir sua utilização adequada:
  - ❑ Contexto
  - ❑ Problema
  - ❑ Solução

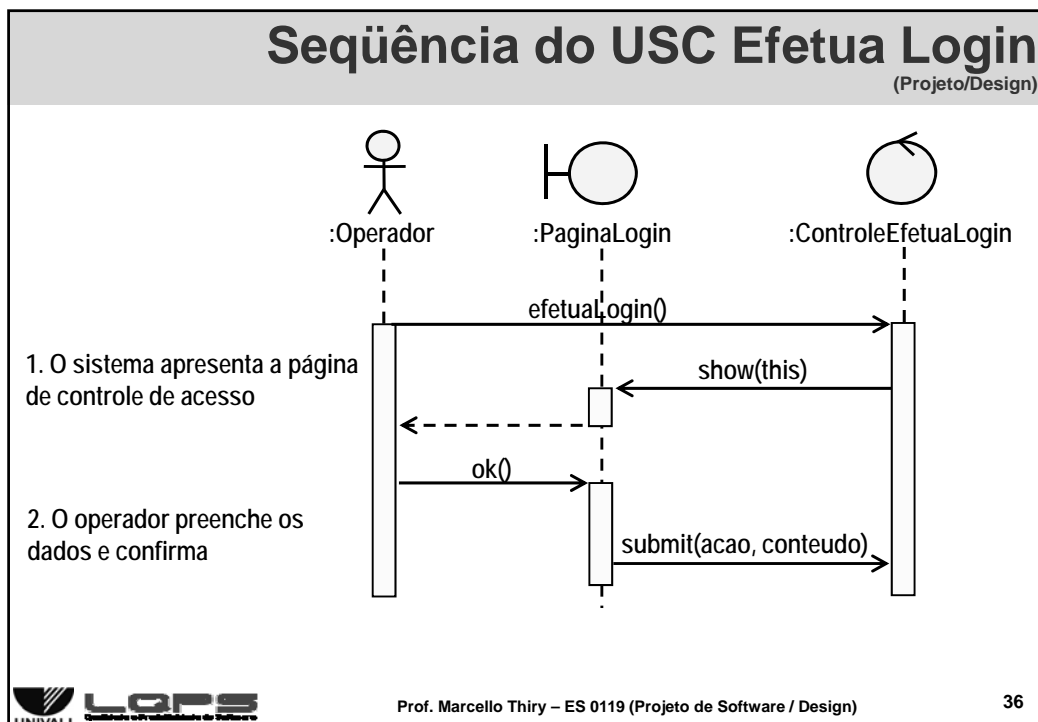
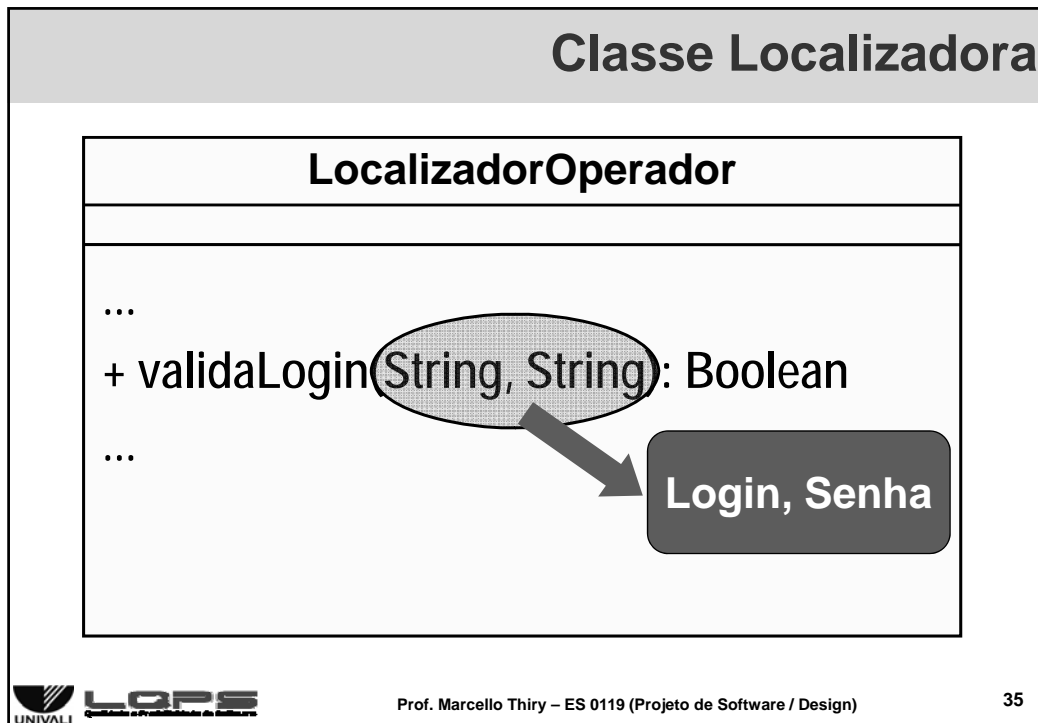


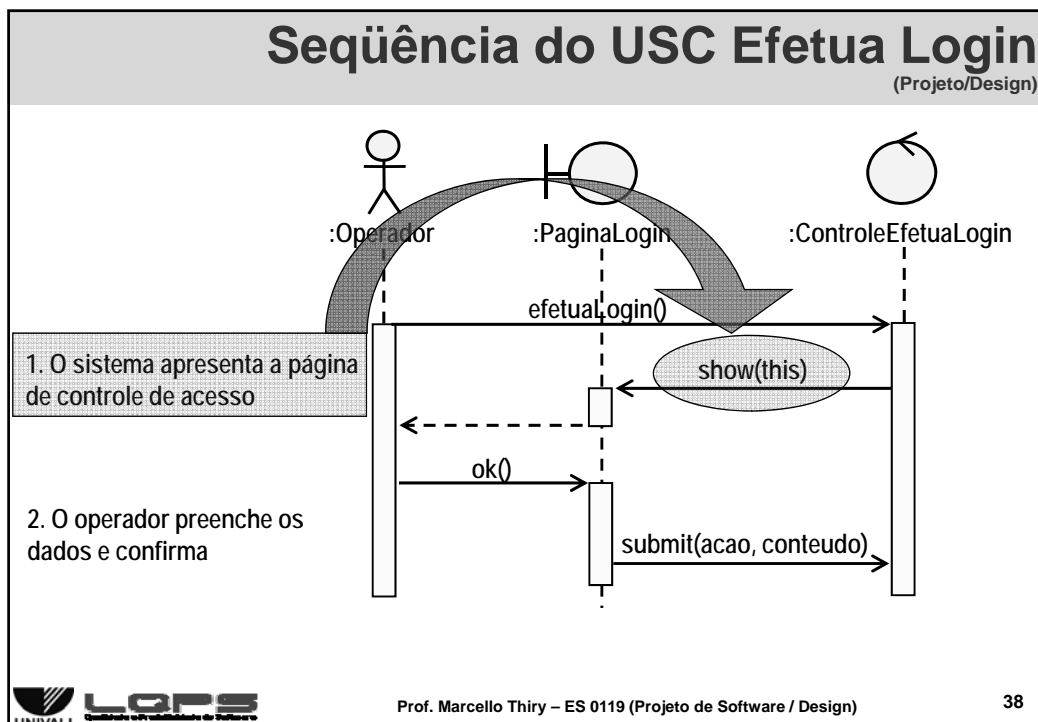
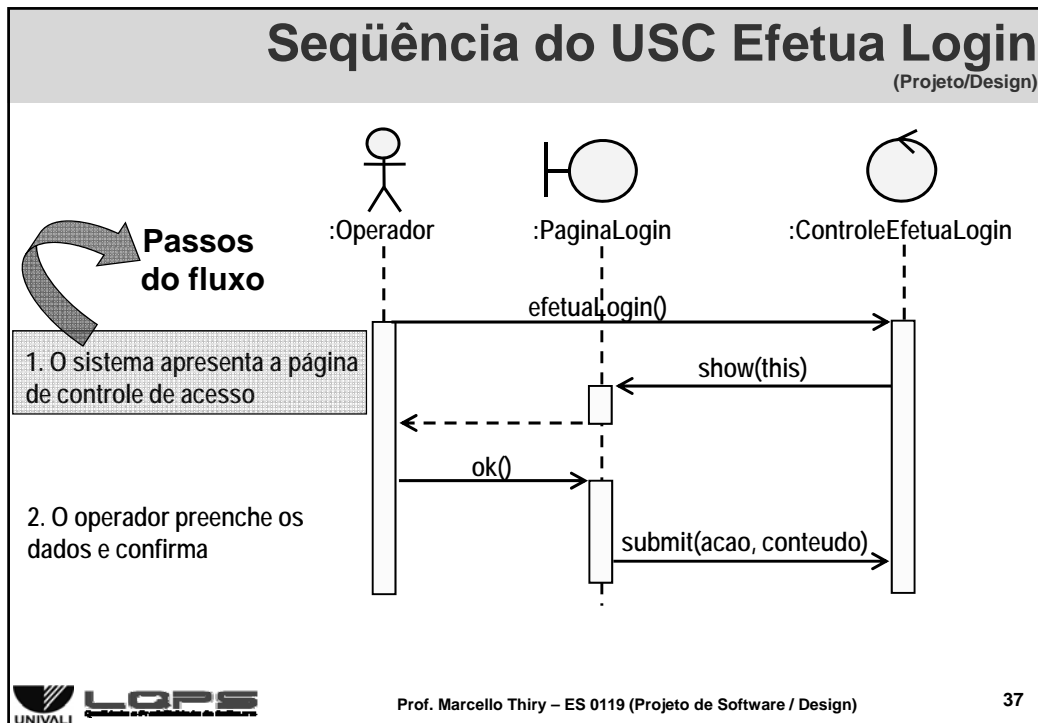


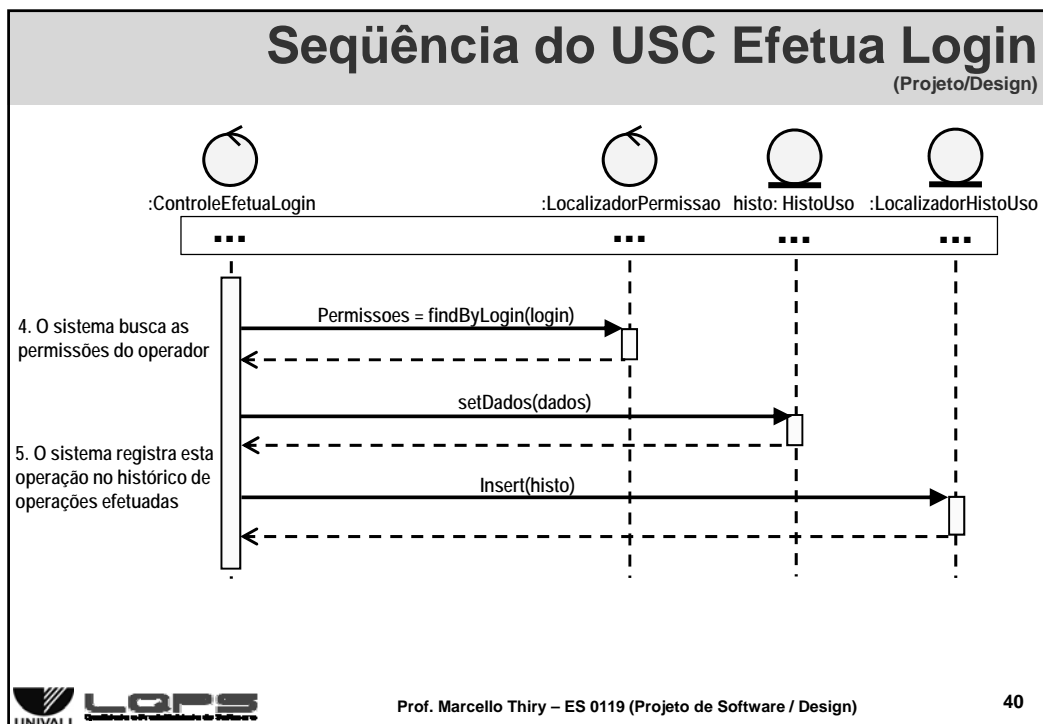
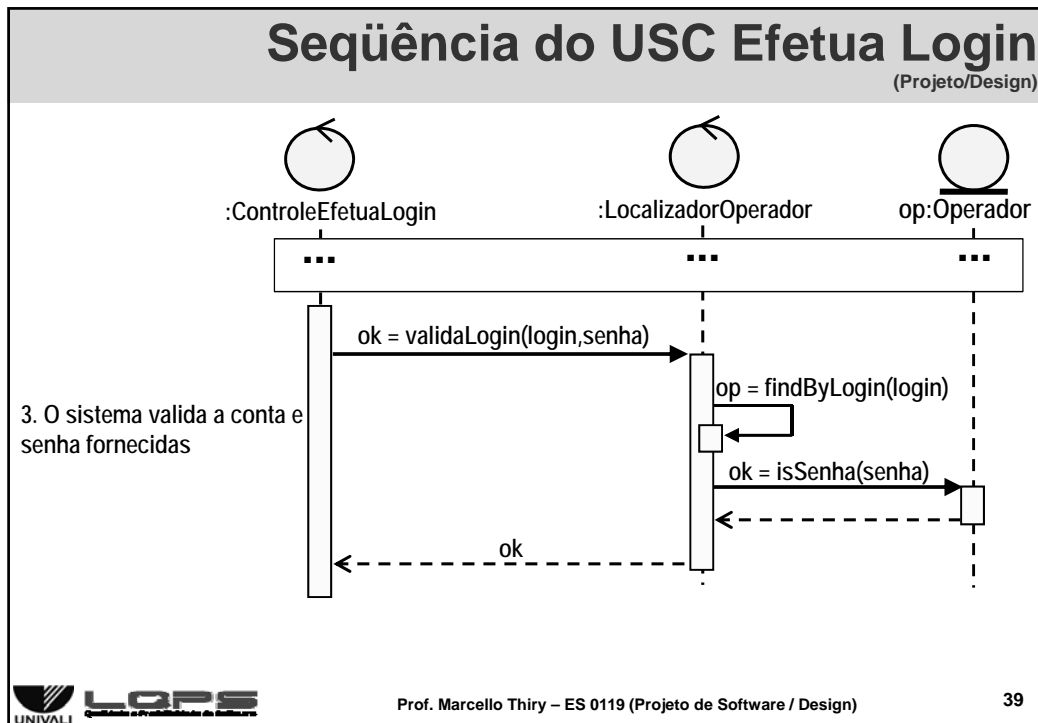


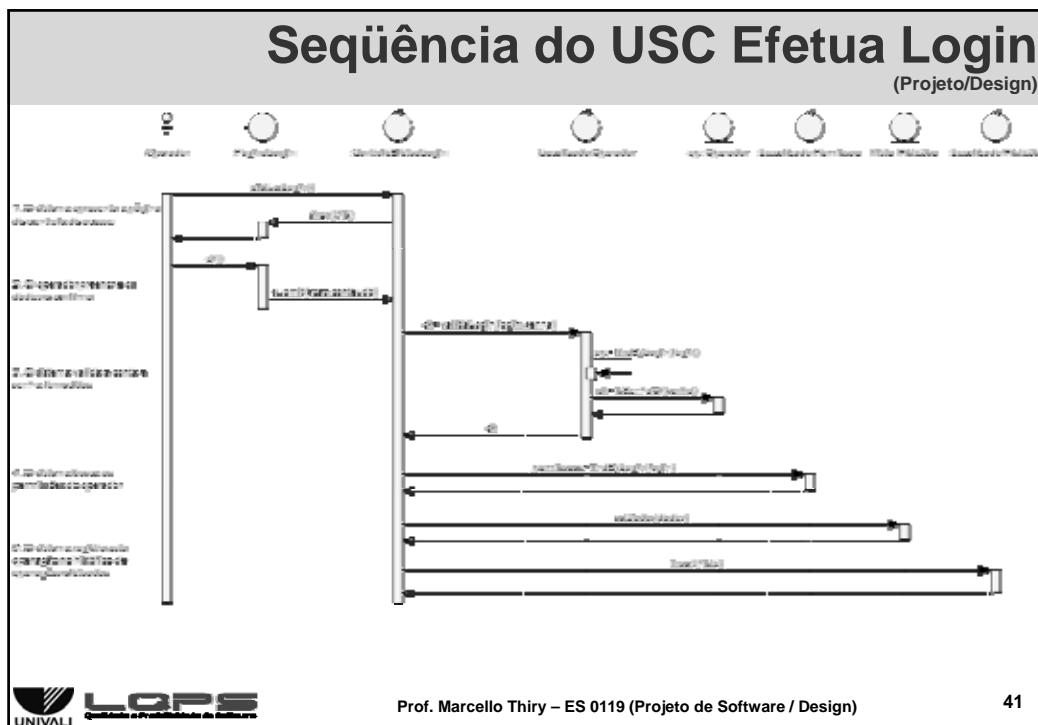












### Seqüência do USC Efetua Login

(Projeto/Design)

```


class CtrlEfetuaLogin {
    ...
    public void efetuaLogin() {
        PaginaLogin pg = new PaginaLogin();
        pg.show(this);
    }
    ...
    public void submit(String acao, Object conteudo) {
        String login = (ConteudoPgLogin)conteudo.getLogin();
        String senha = (ConteudoPgLogin)conteudo.getSenha();
        ok = LocalizadorOperador.validaLogin(login, senha);
        permissoes = LocalizadorPermissao.findByLogin(login);
        HistoUso log = new HistoUso();
        log.setAtributo1(valor); // setDados(dados)
        ...
        log.setAtributoN(valor); // setDados(dados)
        LocalizadorHistoUso.insere(log);
    }
    ...
}
    
```

**UNIVALI LQPS** Qualidade e Produtividade de Software

Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design) 42

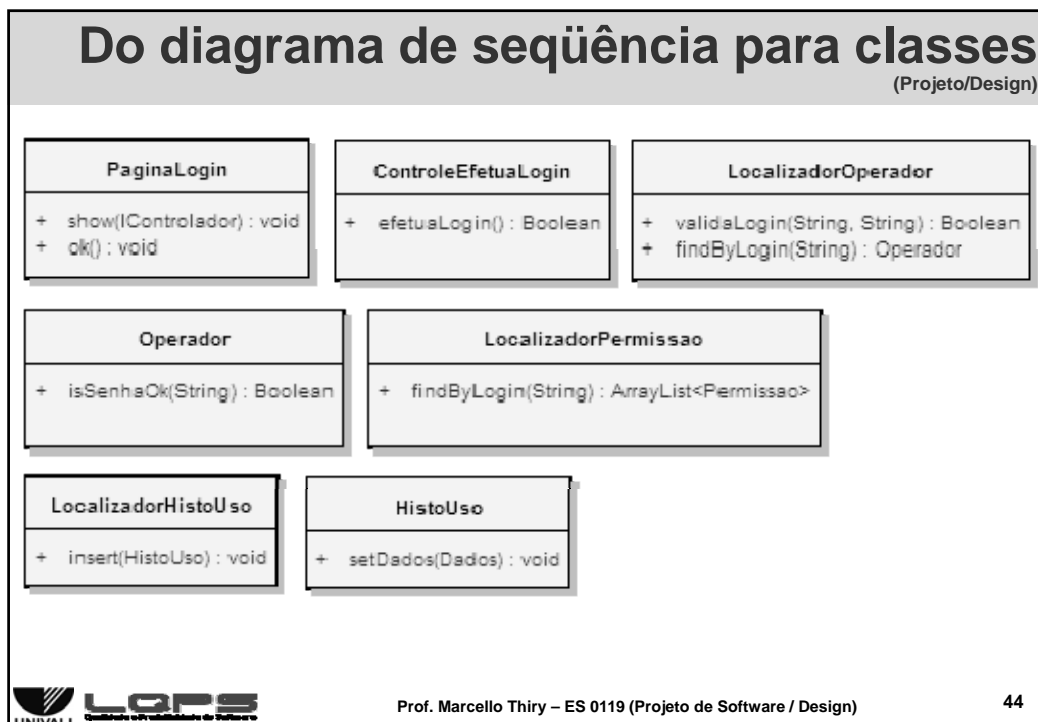
<b>Matriz de Rastreabilidade Bidirecional</b> (Projeto/Design)				
	Colaboração 1	Colaboração 2	...	Colaboração m
Caso de uso 1	X		...	
Caso de uso 2		X	...	
...	...	...	...	...
Caso de uso n			...	X

☐ Um caso de uso pode estar associado a mais de um diagrama de seqüência



Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

43



## Do diagrama de seqüência para classes

(Projeto/Design)

**PaginaLogin**

+ show(IControladorLogin): Boolean

+ ok(): void

**ControleEfetuaLogin**

+ show(IControladorLogin): Boolean

**LocalizadorOperador**

+ localizar(IControladorLogin): Boolean

+ localizar(IControladorLogin): Boolean


**Operador**

+ isSenhaCorreta(): Boolean

**LocalizadorOperador**

+ insert(HistoricoOperacao): void

# Como encontrar os atributos e os relacionamentos?

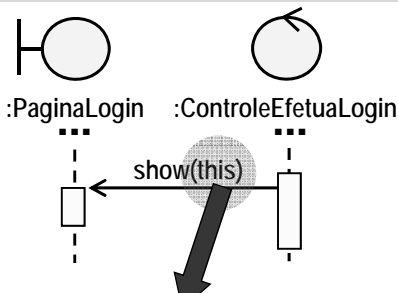


Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

45


## Analisando as mensagens - 1

(Projeto/Design)



**Pode sugerir  
relacionamento  
bidirecional**

- ❑ Uma mensagem de A para B pode indicar uma dependência de A para B:
  - ❑ Uma associação
  - ❑ Uma agregação
  - ❑ Apenas uma dependência mesmo
- ❑ **DICA:** controladores usualmente não tem atributos
  - ❑ As associações são temporárias, sendo apenas dependências



Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

46

## Analisando as mensagens - 2

(Projeto/Design)

The sequence diagram shows a message from `:PaginaLogin` to `:ControleEfetuaLogin` with the payload `submit(acao, conteudo)`. The class diagram below shows `PaginaLogin` with methods `show(IControlador): void` and `ok(): void`, and `ControleEfetuaLogin` with the method `efetuaLogin(): Boolean`. An arrow points from the `submit` message to the `efetuaLogin` method.

- ❑ A `PaginaLogin` conhece o endereço do controlador (ela recebeu o “this”)
- ❑ Em uma **aplicação web**, a página precisa conhecer o controlador, pois este não mantém a conexão com a página:
- ❑ Então a **PaginaLogin** tem uma **associação com o controlador**

UNIVALI Laboratório de Qualidade e Produtividade de Software

Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design) 47

## Analisando as mensagens - 3

(Projeto/Design)

The sequence diagram shows a self-message on the `:LocalizadorOperador` object with the payload `op = findByLogin(login)`. The message is highlighted with a grey oval.

- ❑ **Auto-mensagens** podem indicar que a operação possui visibilidade **privada**
- ❑ Entretanto, deve-se analisar se a operação não é ou poderia ser utilizada por outras classes
- ❑ Operações de busca em localizadores usualmente podem ser usadas em outros contextos

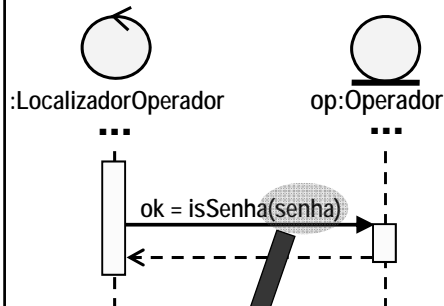
UNIVALI Laboratório de Qualidade e Produtividade de Software

Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design) 48



## Analisando as mensagens - 4

(Projeto/Design)



- ❑ Quem armazena a **senha**?
- ❑ Os atributos das entidades podem ser identificados a partir das mensagens enviadas para este tipo de objeto

Possível atributo  
"senha" na classe  
Operador

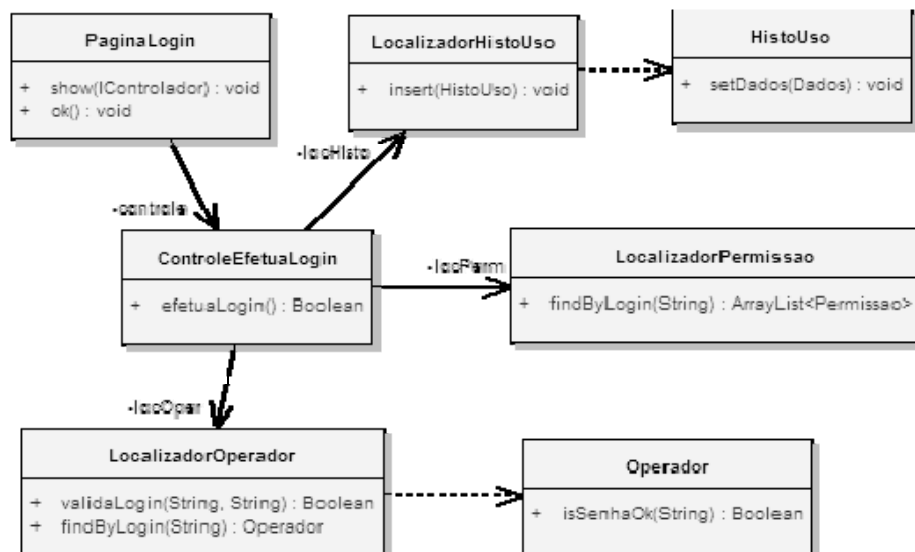


Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

49

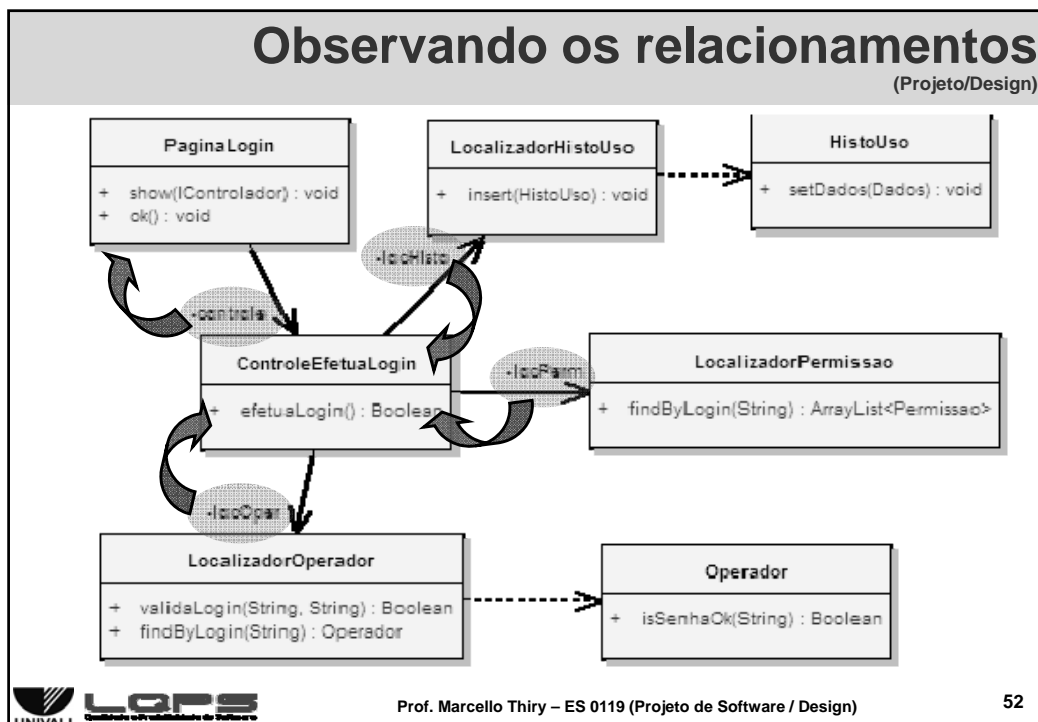
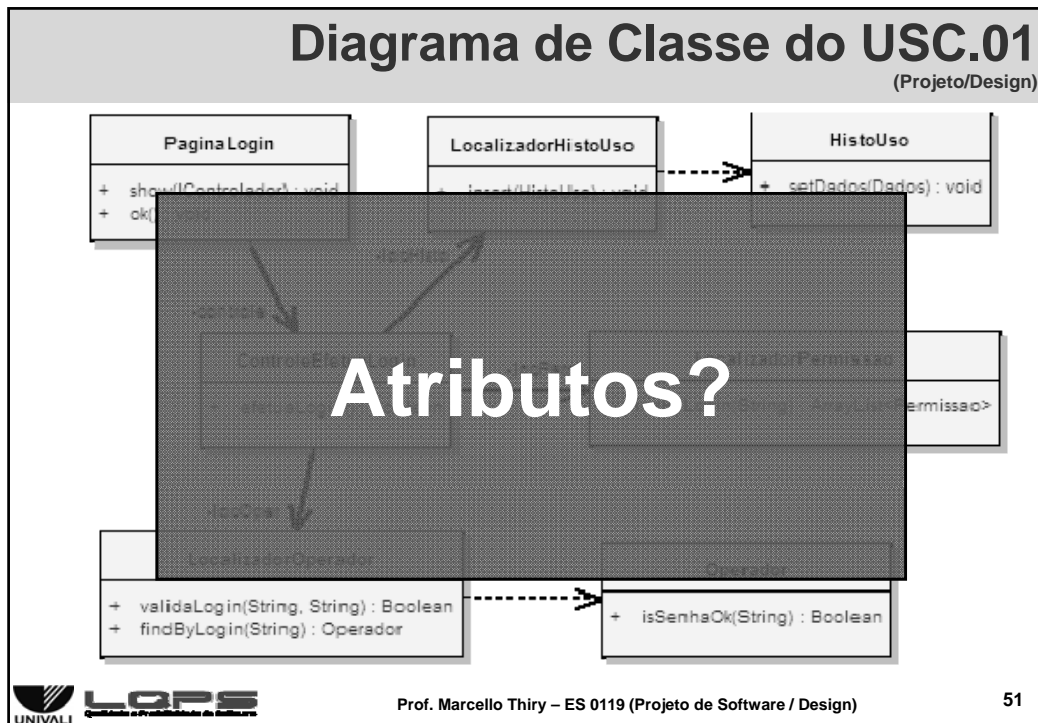
## Diagrama de Classe do USC.01

(Projeto/Design)



Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

50



## Atributos e entidades

(Projeto/Design)

Operador
- login: String
- senha: String
+ isSenhaOk(String) : Boolean
+ getLogin() : String
+ setLogin(String) : void
+ getSenha() : String
+ setSenha(String) : void

- ❑ As mensagens podem auxiliar na identificação dos atributos das entidades
  - ❑ **IsSenhaOk(senha)**
- ❑ Observar dados manipulados pelos casos de uso
- ❑ Observar documentos, formulários utilizados, etc
- ❑ As classes de limite também são fontes reais de atributos

Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

53

## Revisando as classes de entidade

(Projeto/Design)

- ❑ Revisar os atributos, incluindo os tipos definidos
- ❑ Buscar similaridades entre atributos e operações
  - ❑ Possível utilização do relacionamento de generalização
- ❑ Verificar os relacionamentos de associação e agregação (composição)
  - ❑ Se houver muita dúvida para decidir entre qual relacionamento usar, opte pela associação
- ❑ Nomear os relacionamentos: frases verbais ou nomes de papel
- ❑ Estabelecer a cardinalidade nos relacionamentos
- ❑ Revisar a visibilidade das operações
- ❑ Revisar as operações

Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

54

## Entidades e persistência

(Projeto/Design)

- ☐ As classes de entidade são a base para o mapeamento do banco de dados
- ☐ Cuidado para não mapear diretamente classes para tabelas
- ☐ Adotar padrões de mapeamento para garantir consistência entre o modelo de classe e modelo de dados



Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

55

## Matriz de Rastreabilidade Bidirecional

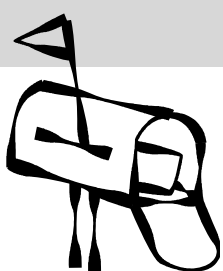

(Projeto/Design)

	Classe 1	Classe 2	...	Classe m
Colaboração 1	X		...	
Colaboração 2		X	...	
...	...	...	...	...
Colaboração n			...	X



Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)

56

<b>Contato</b>	
	<p><b>Marcello Thiry</b> marcello.thiry@gmail.com</p> <p><b>LQPS</b> <a href="http://www.univali.br/lqps">http://www.univali.br/lqps</a></p>
	<p>Prof. Marcello Thiry – ES 0119 (Projeto de Software / Design)</p> <p style="text-align: right;">57</p>