Desenvolvimento de Sistemas Software

Aula Teórica 7: Diagramas de Use Case (cont.)

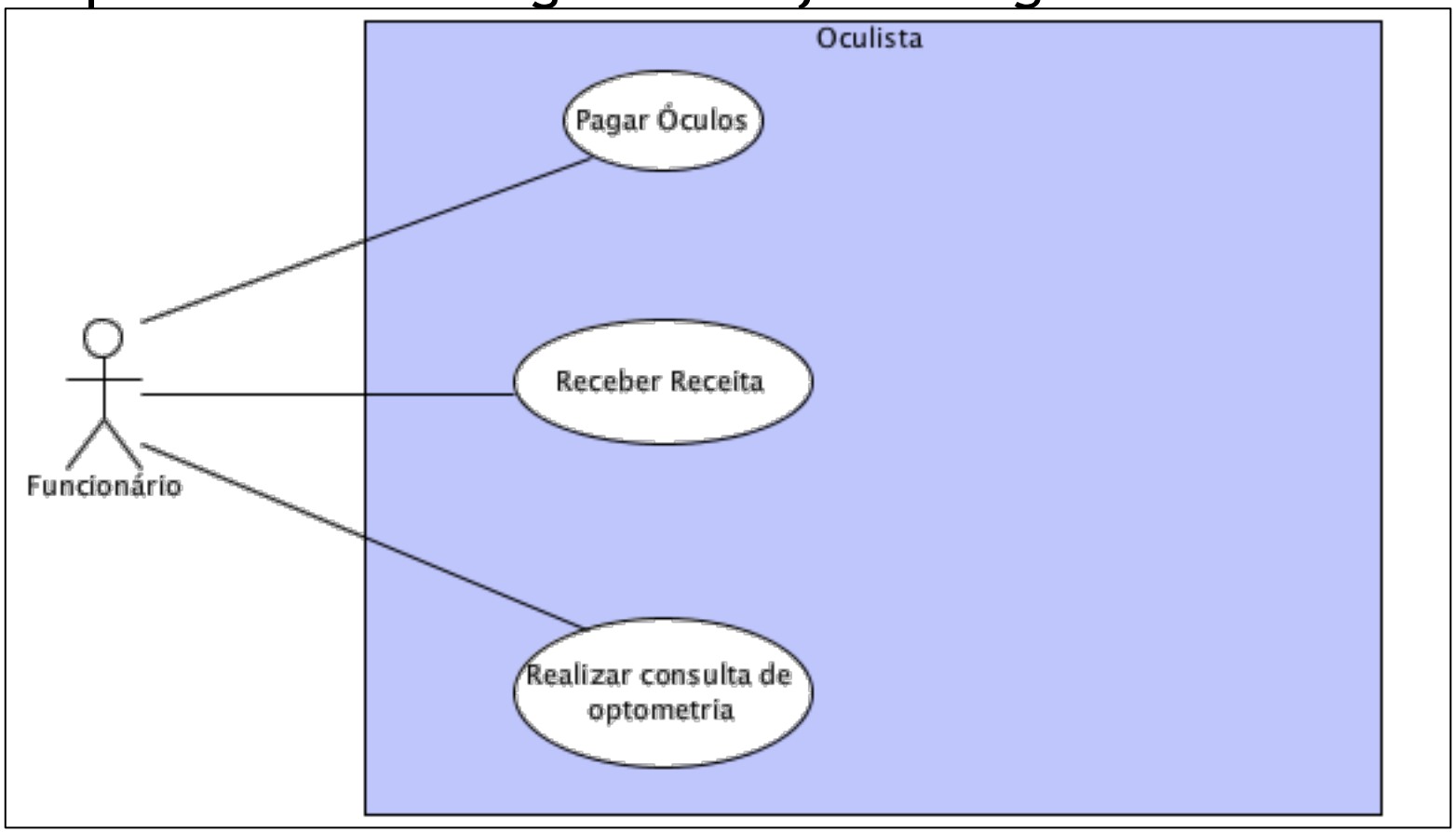
Desenvolvimento de Sistemas Software
José Creissac Campos/António Nestor Ribeiro

"Good use cases are balanced, describing essential system behavior while providing only the necessary details about the interactions between a system and its users"

Patterns for Effective Use Cases

Dependências revisitadas (<<include>> / <<extend>>)

- Mecanismos de estruturação dos modelos
- Adoptar uma abordagem de *refactoring*.



Use Case: Receber receita
Descrição: Funcionário processa a receita de um cliente
Pré-condição: Existe papel para imprimir talões
Pós-condição: Pedido de óculos fica registado
Comportamento normal:

1. Funcionário indica nome e/ou data de nascimento do cliente
2. Sistema apresenta lista de clientes correspondentes
3. Funcionário selecciona cliente
4. Sistema apresenta detalhes do cliente
5. Funcionário confirma dados
6. Funcionário indica código de armação e de
7. Sistema procura produto e apresenta detal
8. Funcionário confirma
9. Sistema regista reserva e imprime talão

Comportamento Alternativo [lista de clientes co

2.1. Sistema apresenta detalhes do único clien
2.2. regressa a 5

Comportamento Alternativo

3.1. Funcionário escolhe criar novo cliente
3.2. Funcionário introduz dados do cliente
3.3. Sistema regista cliente
3.4. regressa a 6

Excepção

8.1. Funcionário não confirma produto
8.2. Sistema cancela reserva

Use Case: Realizar consulta de o ometria
Descrição: Funcionário cobra a re lização de uma consulta
Pré-condição: Existe papel para i rimir talões
Pós-condição: Consulta fica paga
Comportamento normal:

1. Funcionário indica nome e/ou data de nascimento do cliente
2. Sistema apresenta lista de clientes correspondentes
3. Funcionário selecciona cliente
4. Sistema apresenta detalhes do cliente
5. Funcionário confirma dados
6. Sistema determina dívidas do cliente
7. Sistema indica valor total a cobrar (dívidas + consulta)
8. Funcionário confirma pagamento
9. Sistema regista reserva e imprime talão

Comportamento Alternativo [lista de clientes correspondentes tem tamanho 1]

2.1. Sistema apresenta detalhes do único cliente da lista
2.2. regressa a 5

Comportamento Alternativo

3.1. Funcionário escolhe criar novo cliente
3.2. Funcionário introduz dados do cliente
3.3. Sistema regista cliente
3.4. regressa a 6

Excepção

8.1. Funcionário não confirma pagamento
8.2. Sistema regista dívida na ficha de cliente

Use Case: Receber receita		
Descrição: Funcionário processa a receita de um cliente		
Pré-condição: Existe papel para imprimir talões		
Pós-condição: Pedido de óculos fica registado		
	Actor	**Sistema**
Comportamento Normal	1. indica nome e/ou data de nascimento do cliente	
		2. apresenta lista de clientes correspondentes
	3. selecciona cliente	
		4. apresenta detalhes do cliente
	5. cg	
	6. ir	
	8. C	
Comp. Alternativo 1 [lista de clientes corresponentes tem tamanho 1] (passo 2)		
Comp. Alternativo 2 (passo 3)	3.1.	
	3.2.	
Excepção 1 (passo 8)	8.1.	

Use Case: Receber receita		
Descrição: Funcionário processa a receita de um cliente		
Pré-condição: Existe papel para imprimir talões		
Pós-condição: Pedido de óculos fica registado		
	Actor	**Sistema**
Comportamento Normal	1. indica nome e/ou data de nascimento do cliente	
		2. apresenta lista de clientes correspondentes
	3. selecciona cliente	
		4. apresenta detalhes do cliente
	5. confirma dados	
	6. indica código de armação e de lentes	
		7. procura produto e apresenta detalhes
	8. Confirma dados	
		9. regista reserva e imprime talão
Comp. Alternativo 1 [lista de clientes corresponentes tem tamanho 1] (passo 2)		2.1. spresenta detalhes do único cliente da lista
		regressa a 6
Comp. Alternativo 2 (passo 3)	3.1. escolhe criar novo cliente	
	3.2. introduz dados do cliente	
		3.3. regista cliente
		regressa a 6
Excepção 1 (passo 8)	8.1. não confirma produto	
		8.2. cancela reserva

Use Case: Receber receita
Descrição: Funcionário processa a receita de um cliente
Pré-condição: Existe papel para imprimir talões
Pós-condição: Pedido de óculos fica registado
Comportamento normal:
1. <<include>> identificar cliente
2. Funcionário indica código de armação e de lentes
3. Sistema procura produto e apresenta detalhes
4. Funcionário confirma
5. Sistema regista reserva e imprime talão

Excepção
4.1. Funcionário não confirma produto
4.2. Sistema cancela reserva

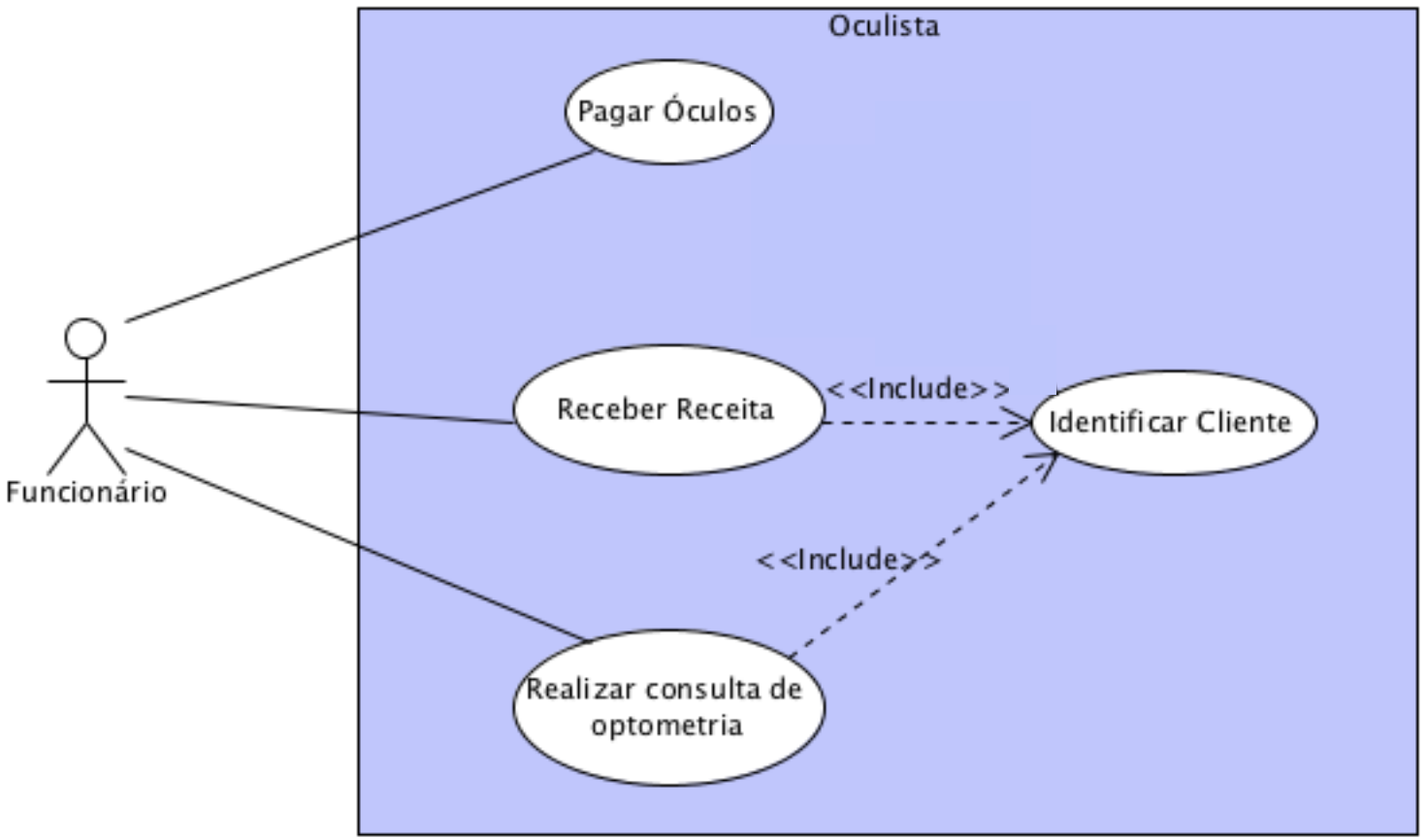
Use Case: Realizar consulta de optometria
Descrição: Funcionário cobra a realização de uma consulta
Pré-condição: Existe papel para imprimir talões
Pós-condição: Consulta fica paga
Comportamento normal:
1. <<include>> identificar cliente
2. Sistema determina dívidas do cliente
3. Sistema indica valor total a cobrar (dívidas + consulta)
4. Funcionário confirma pagamento
5. Sistema regista reserva e imprime talão

Excepção
4.1. Funcionário não confirma pagamento
4.2. Sistema regista dívida na ficha de cliente

Use Case: Identificar cliente
Descrição: identificação de um cliente por nome
Pré-condição:
Pós-condição: Cliente pretendido fica seleccionado
Comportamento normal:
1. Funcionário indica nome e/ou data de nascimento do cliente
2. Sistema apresenta lista de clientes correspondentes
3. Funcionário selecciona cliente
4. Sistema apresenta detalhes do cliente
5. Funcionário confirma dados

Comportamento Alternativo [lista de clientes correspondentes tem tamanho 1]
2.1. Sistema apresenta detalhes do único cliente da lista
2.2. regressa a 5

Comportamento Alternativo
3.1. Funcionário escolhe criar novo cliente
3.2. Funcionário introduz dados do cliente
3.3. Sistema regista cliente


Oculista
Pagar Óculos
Receber Receita
<<Include>>
Identificar Cliente
<<Include>>
Realizar consulta de optometria
Funcionário

Use Case: Realizar consulta de optometria
Descrição: Funcionário cobra a realização de uma consulta
Pré-condição: Existe papel para imprimir talões
Pós-condição: Consulta fica paga
Comportamento normal:
1. <<include>> identificar cliente
2. Sistema determina dívidas do cliente
3. Sistema indica valor total a cobrar (dívidas + consulta)
4. Funcionário confirma pagamento
5. Sistema regista reserva e imprime talão

Excepção
4.1. Funcionário não confirma pagamento
4.2. Sistema regista dívida na ficha de cliente

Alternativa [cliente tem seguro de saúde]
3.1. Sistema pede confirmação de dados do seguro
3.2. Funcionário confirma dados
3.3. regressa a 3

Alternativa [dados inválidos]
3.2.1. Funcionário não confirma dados
3.2.2. Sistema propõe preço normal
3.2.3. Funcionário aceita
3.2.4. regressa a 3.3

Excepção [não aceita preço normal]
3.2.3.1. Funcionário não aceita peço normal

Use Case: Pagar óculos
Descrição: Funcionário cobra um par de óculos
Pré-condição: Existe papel para imprimir talões
Pós-condição: Óculos ficam pagos
Comportamento normal:
1. Funcionários indica número de talão de óculos a pagar
2. Sistema determina dívidas do cliente
3. Sistema indica valor total a cobrar (dívidas + óculos)
4. Funcionário confirma pagamento
5. Sistema regista pagamento e imprime talão

Excepção
4.1. Funcionário não confirma pagamento
4.2. Sistema regista dívida na ficha de cliente

Alternativa [cliente tem seguro de saúde]
3.1. Sistema pede confirmação de dados do seguro
3.2. Funcionário confirma dados
3.3. regressa a 3

Alternativa [dados inválidos]
3.2.1. Funcionário não confirma dados
3.2.2. Sistema propõe preço normal
3.2.3. Funcionário aceita
3.2.4. regressa a 3.3

Excepção [não aceita preço normal]
3.2.3.1. Funcionário não aceita peço normal

Use Case: Realizar consulta de optometria
Descrição: Funcionário cobra a realização de uma consulta
Pré-condição: Existe papel para imprimir talões
Pós-condição: Consulta fica paga
Comportamento normal:
1. <<include>> identificar cliente
2. Sistema determina dívidas do cliente
3. Sistema indica valor total a cobrar (dívidas + consulta)
   [ponto de extensão: valor a pagar]
4. Funcionário confirma pagamento
5. Sistema regista reserva e imprime talão

Excepção
4.1. Funcionário não confirma pagamento
4.2. Sistema regista dívida na ficha de cliente

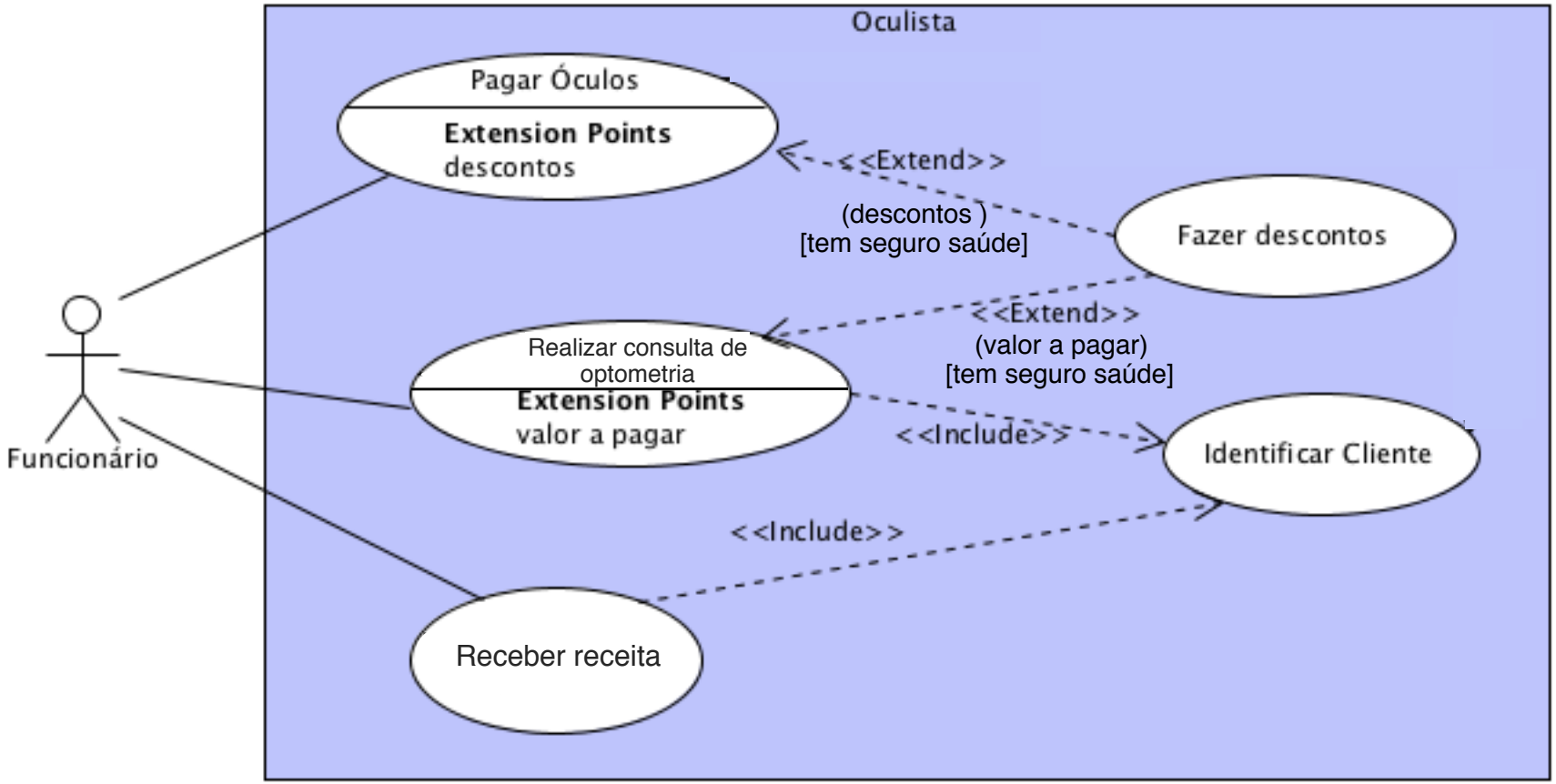
Use Case: Pagar óculos
Descrição: Funcionário cobra um par de óculos
Pré-condição: Existe papel para imprimir talões
Pós-condição: Óculos ficam pagos
Comportamento normal:
1. Funcionários indica número de talão de óculos a pagar
2. Sistema determina dívidas do cliente
3. Sistema indica valor total a cobrar (dívidas + óculos)
   [ponto de extensão: descontos]
4. Funcionário confirma pagamento
5. Sistema regista pagamento e imprime talão

Excepção
4.1. Funcionário não confirma pagamento
4.2. Sistema regista dívida na ficha de cliente

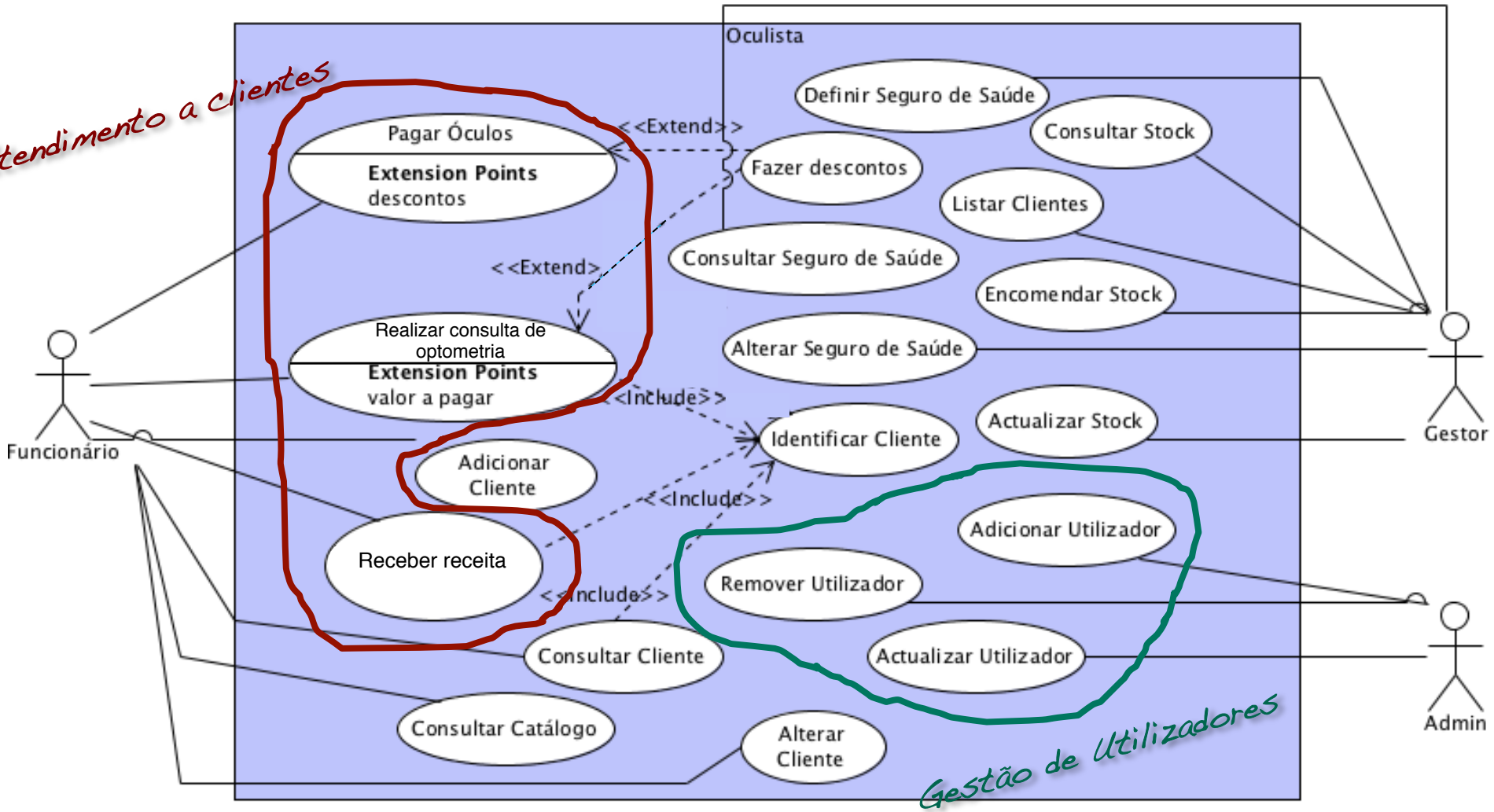
Use Case: Fazer desconto
Descrição: Funcionário cobra a realização de uma consulta
Pré-condição:
Pós-condição: preço fica calculado com desconto
Comportamento normal:
1. Sistema pede confirmação de dados do seguro
2. Funcionário confirma dados

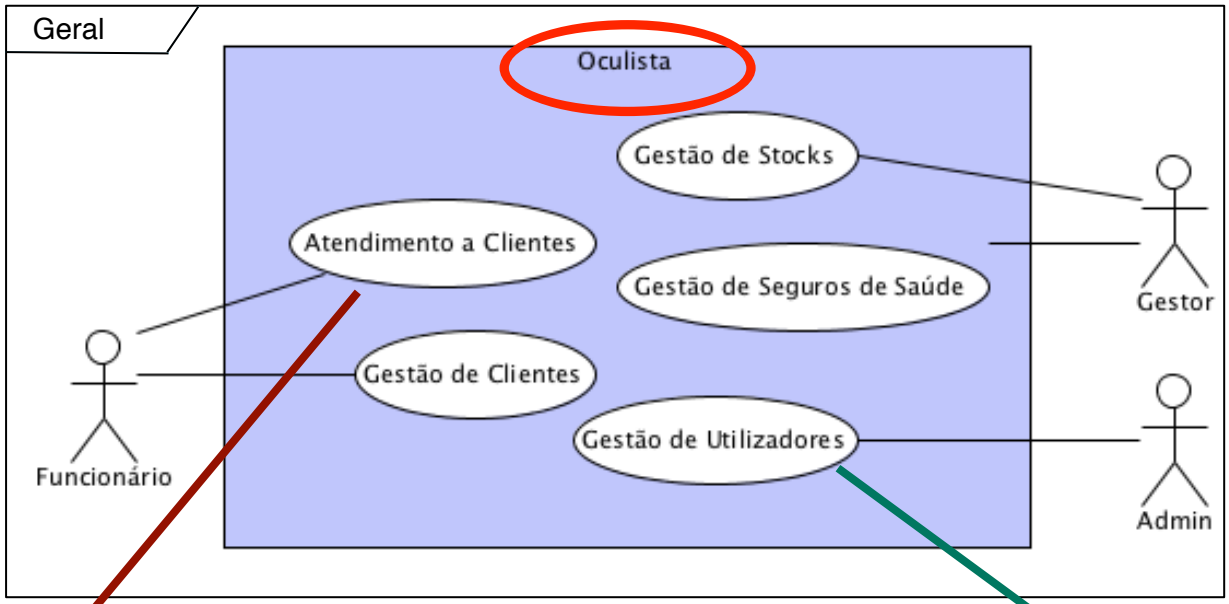
Alternativa [dados inválidos]
2.1. Funcionário não confirma dados
2.2. Sistema propõe preço normal
2.3. Funcionário aceita

Excepção [não aceita preço normal]
2.3.1. Funcionário não aceita peço normal

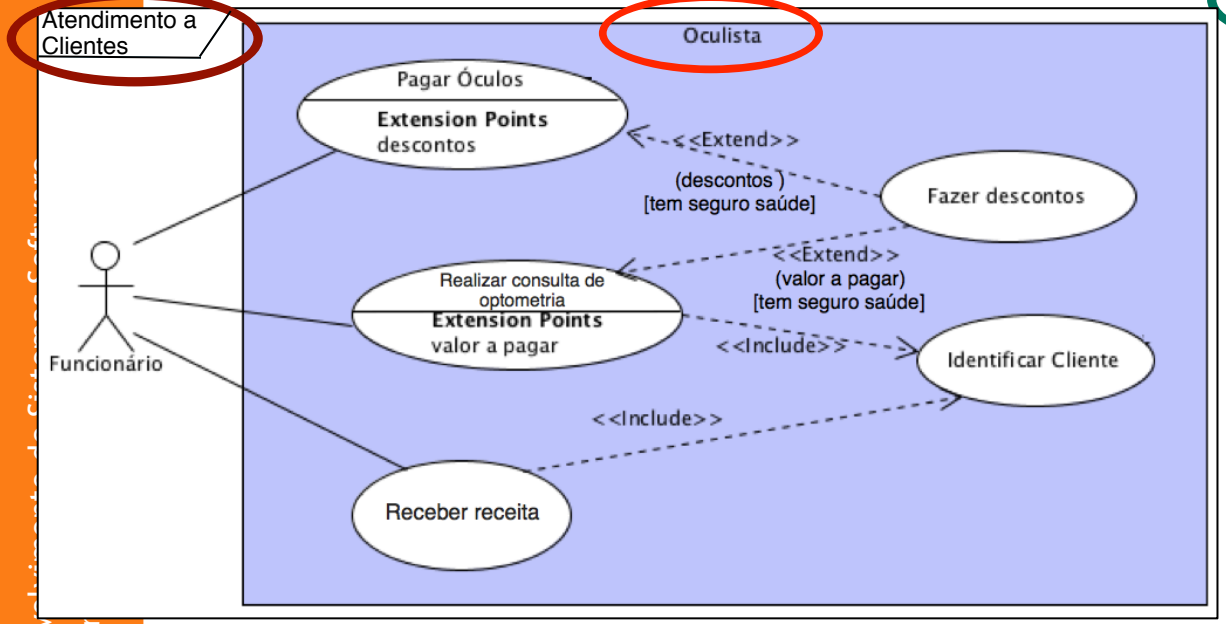

Oculista
Pagar Óculos
Extension Points
descontos
<<Extend>>
(descontos)
[tem seguro saúde]
Fazer descontos
<<Extend>>
(valor a pagar)
[tem seguro saúde]
Realizar consulta de optometria
Extension Points
valor a pagar
<<Include>>
Identificar Cliente
<<Include>>
Receber receita
Funcionário

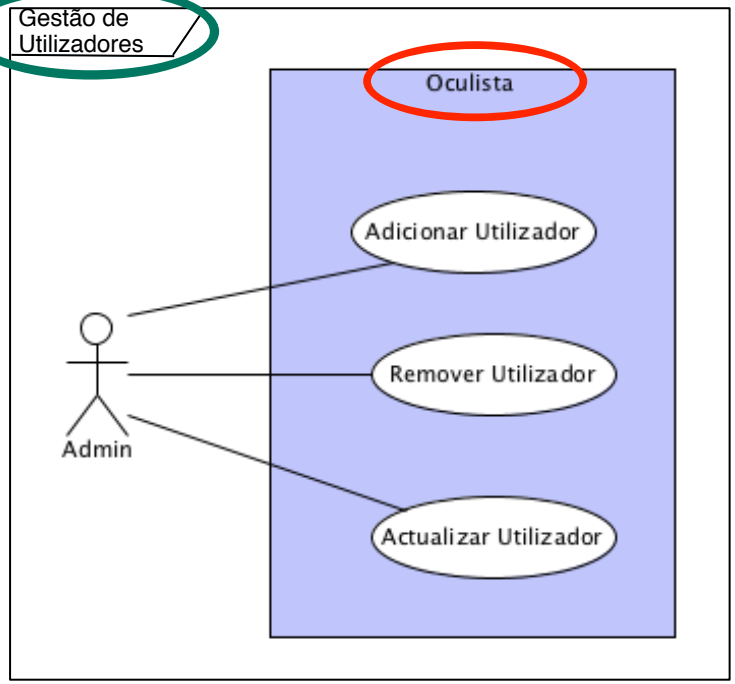
Estruturação de modelos?





Três diagramas, um modelo!





a) Um use case descreve as sequências de interacções entre actores externos e o sistema em projecto, sendo este visto como uma *black box*, para que um dado objectivo ou tarefa se realize.

b) Devem ser especificados num use case quer os cenários de sucesso (tarefa completada com êxito) quer os de insucesso (tarefa não completada);

c) Cada passo de interacção actor-sistema descrito num use case designa-se por *evento*, *acção* ou *operação*, e devem distinguir-se quanto à sua origem (actor ou sistema);

d) Um use case descreve um fluxo principal de eventos/operações, designado *fluxo principal* ou *cenário principal*, bem como outros possíveis fluxos ou caminhos designados *fluxos alternativos* ou *cenários alternativos* bem ainda como fluxos que, sendo alternativos, conduzem a situações de insucesso, a que chamaremos *fluxos de excepção* ou apenas *excepções*;

e) Use cases são multi-nível, ou seja, use cases podem ser especificados usando a funcionalidade especificada noutros use cases através de relações definidas em UML, designadamente, inclusão, extensão e generalização;

f) A generalização é também aplicável aos actores, desta forma sendo possível representar o relacionamento entre actores/papéis perante o sistema;

g) Use cases devem ser simples e legíveis, não devem conter detalhes sobre a interface com o utilizador e devem ter o nível de detalhe necessário a cada iteração de requisitos (são refináveis);

h) Use cases relacionados com actores devem ser identificados por verbos no infinitivo e sujeitos, deixando claro qual a tarefa que o sistema deve fornecer ao actor.

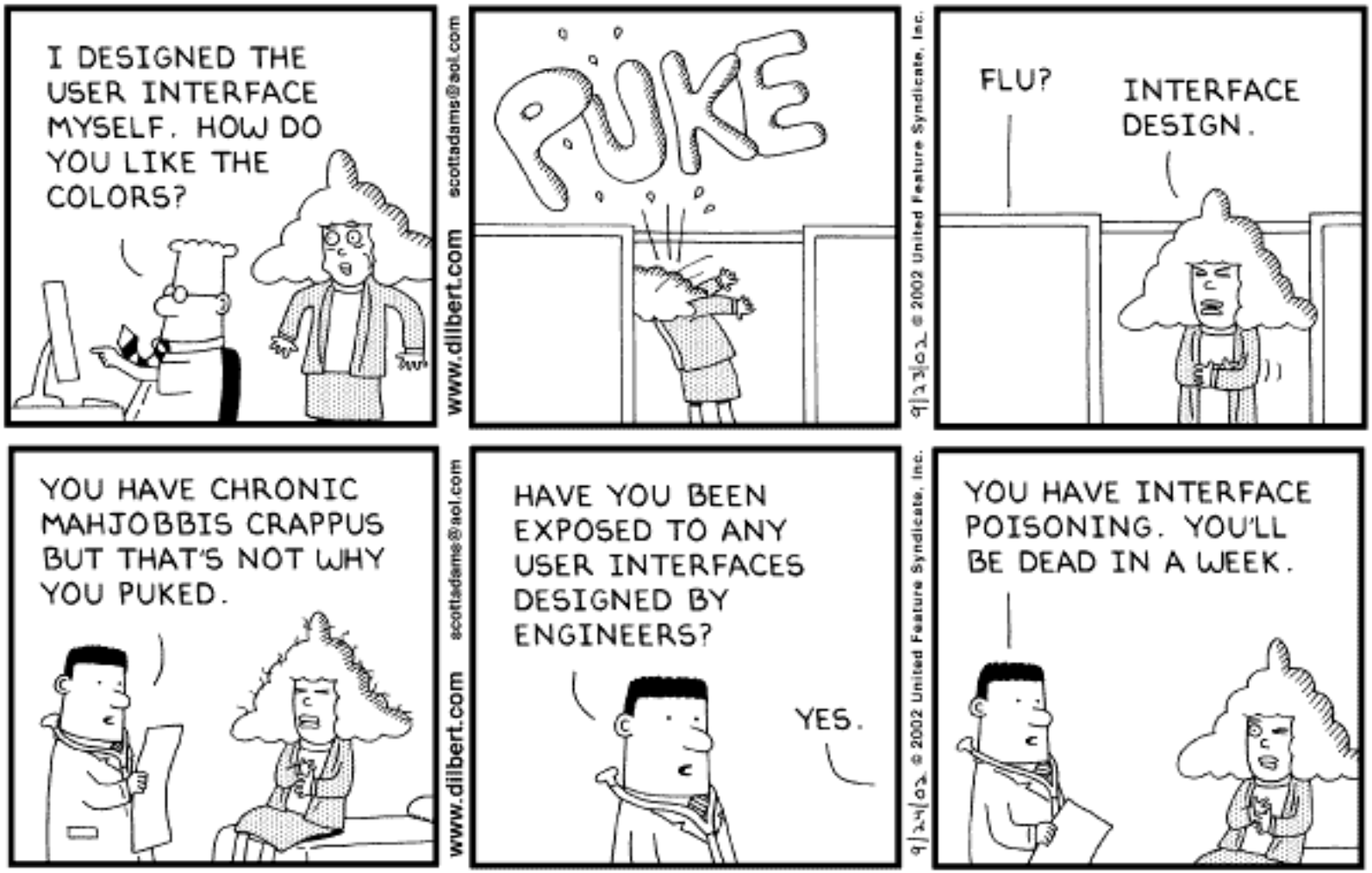
Requisitos funcionais

Requisitos funcionais permitem definir o que o sistema deverá fazer.

Após análise de requisitos, UP propõe a concepção do sistema começando pela arquitectura.

O que falta!

Interface com o utilizador?



Trabalho

Enunciado do Trabalho Prático

Desenvolvimento de Sistemas Software

Licenciatura em Engenharia Informática

2013/2014

Introdução

Este documento apresenta o enunciado do trabalho prático da Unidade Curricular de Desenvolvimento de Sistemas Software para o ano lectivo 2013/2014. **Leia**-o **com atenção**.

Objectivos

Pretende-se com este trabalho que desenvolvam um sistema de suporte à gestão da actividade de uma Associação de Escolas de Futebol. Tome como modelo a APEF (Associação Portuguesa de Escolas de Futebol[1]). O sistema deverá permitir gerir as escolas associadas e as equipas de cada escola, bem como o calendário de torneios e os jogos em cada torneio.

O Regulamento Técnico para os torneios da APEF pode ser consultado *online*[2]. Note, no entanto, que o referido regulamento é omisso em relação à forma como se organizam os torneios. Assim o sistema a implementar deverá ser suficientemente flexível para permitir diferentes tipos de torneio. Por exemplo, uma primeira fase de grupos seguida de eliminatórias para apurar a classificação final; ou uma primeira fase de grupos, seguida de uma segunda fase de dois grupos (o dos melhores e o dos piores). O sistema deverá permitir trocar facilmente o algoritmo, bem como adaptar-se a diferentes números de equipas e campos por torneio.

[1] Ver: http://www.apefescolasdefutebol.net/
[2] Ver: http://www.apefescolasdefutebol.net/?page_id=172

Para além dos torneios, pretende-se também prever a possibilidade de ser organizado um campeonato ao longo do ano. O campeonato deverá seguir as regras definidas no Regulamento das Competições organizadas pela Liga Portuguesa de Futebol para a II Liga 2013/2014[3]. Enquanto no campeonato existe um conjunto fixo de equipas em competição ao longo da temporada, nos torneios as equipas participantes são definidas torneio a torneio. É necessário ter em atenção que o campeonato e torneios não entrem em conflito (quer em termos de equipas, quer em termos de campos desportivos).

O sistema a construir deverá prever uma interface de gestão, uma interface para registo de resultados durante as provas e uma interface de consulta (para o público), em que será possível consultar o calendários das provas, bem assim como os resultados e classificações até à data.

As soluções metodológicas e tecnológicas a adoptar no desenvolvimento das aplicações terão que obedecer ao seguinte conjunto de requisitos:

- A aplicação deverá ser desenvolvida utilizando tecnologias orientadas a objetos (idealmente desenvolvida em Java).
- A análise de requisitos deverá produzir um Modelo de Domínio, um modelo de *Use Case* e uma proposta de interface com o utilizador.
- A concepção e desenvolvimento da aplicação deverá seguir uma abordagem suportada por UML, de acordo com o processo descrito nas aulas.

Avaliação

O trabalho será realizado em duas fases. Em cada fase deverá ser entregue um relatório detalhando os objectivos da fase, a abordagem seguida para os atingir, os resultados obtidos (os modelos) e uma análise critica dos mesmos.

- Fase 1: relatório de análise de requisitos, a entregar até 10 de Novembro

[3] Ver: http://www.ligaportugal.pt/media/6639/Regulamento-das-Competicoes-2013-2014-.pdf

Trabalho

- Fase 2: relatório final de modelação e desenvolvimento e software produzido, a entregar até 5 de Janeiro

Cada fase terá uma avaliação independente. Os pesos de cada fase são indicados abaixo.

- Fase 1: 35% da classificação final
- Fase 2: 65% da classificação final

A apresentação e discussão final do trabalho será realizada a partir da semana de 07 de Janeiro 2014.

Grupos de Trabalho

Os grupos de trabalho deverão ter entre 3 e 5 elementos. A definição dos grupos de trabalho será efectuada com recurso ao sistema de *elearning*, em moldes a definir nas aulas.

José Creissac Campos

António Nestor Ribeiro

Diagramas de Use Case

Sumário:

- Valor dos Use Case no processo de desenvolvimento
- Dependências entre Use Cases (<<include>> / <<extend>>)
- Estruturação de modelos de Use Case