Desenvolvimento de Sistemas Software

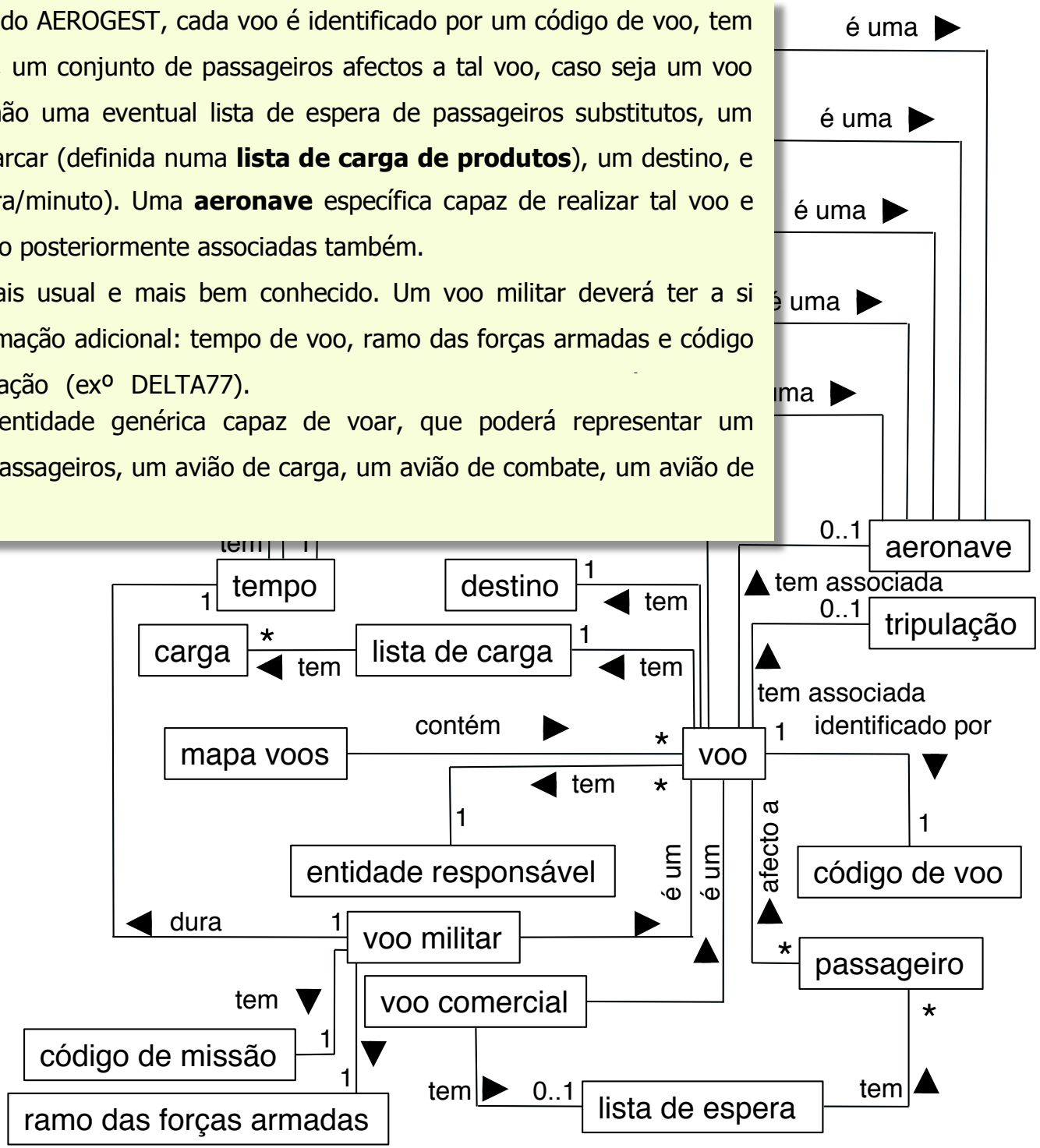
Aula Teórica 5: Modelação do Requisitos Funcionais (Diagramas de Use Case)

Desenvolvimento de Sistemas Software
António Nestor Ribeiro, José Creissac Campos, F. Mário Martins

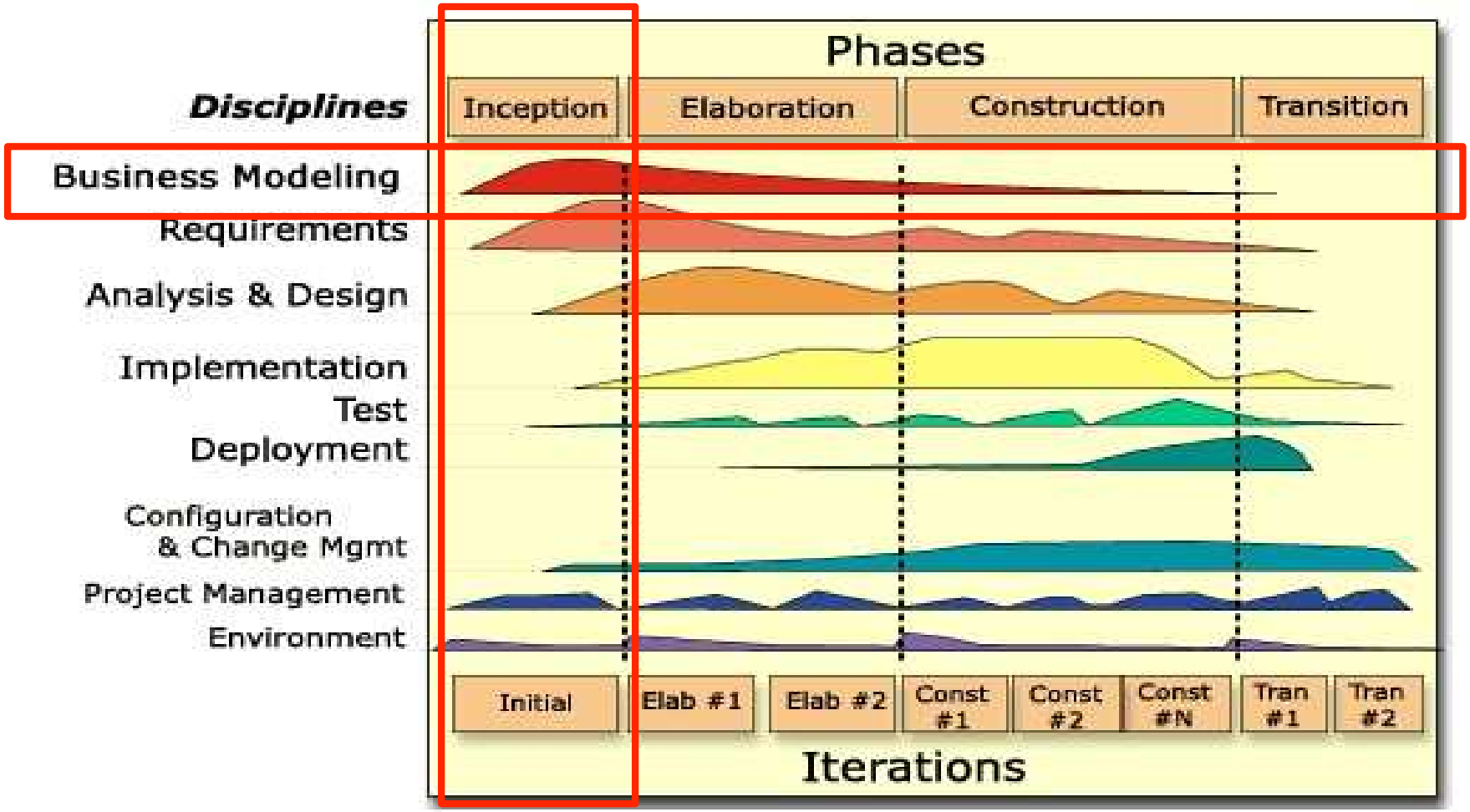
No **Mapa de Voos** do dia do AEROGEST, cada voo é identificado por um código de voo, tem uma entidade responsável, um conjunto de passageiros afectos a tal voo, caso seja um voo comercial poderá ter ou não uma eventual lista de espera de passageiros substitutos, um conjunto de cargas a embarcar (definida numa **lista de carga de produtos**), um destino, e um tempo de partida (hora/minuto). Uma **aeronave** específica capaz de realizar tal voo e uma **tripulação**, ser-lhe-ão posteriormente associadas também.

Um voo comercial é o mais usual e mais bem conhecido. Um voo militar deverá ter a si associada a seguinte informação adicional: tempo de voo, ramo das forças armadas e código de missão para comunicação (exº DELTA77).

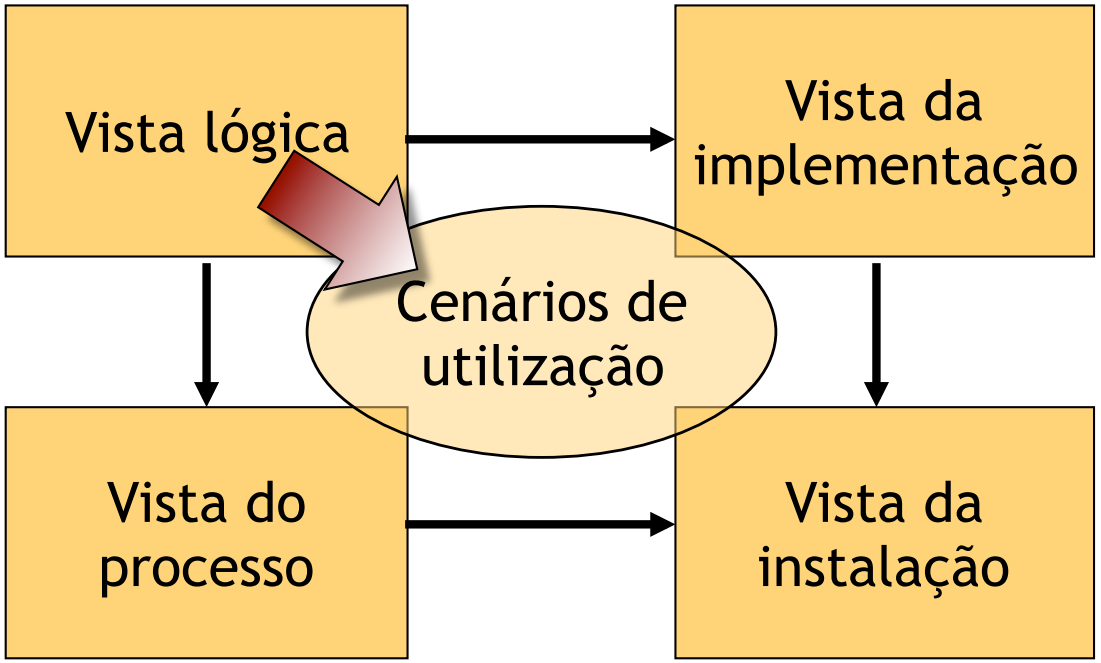
Uma **Aeronave** é uma entidade genérica capaz de voar, que poderá representar um helicóptero, um avião de passageiros, um avião de carga, um avião de combate, um avião de incêndios, etc.



Próximos passos…



Onde estamos…



(Kruchten, 1995)

Definição de requisitos funcionais

Definição de requisitos do sistema, duas abordagens possíveis:

- Visão estrutural – interna
- Visão orientada aos *use case* – externa

Visão Estrutural (OO)

- Definir classes;
- Definir métodos das classes;
- Definir interface com o utilizador (comportamento do sistemas face ao utilizador);

Problemas: O que interessa ao utilizador é o comportamento do sistema, no entanto a interface com o utilizador só é definida no final do processo.

- Perigo de o sistema não fornecer toda a funcionalidade pretendida;
- Perigo de o sistema fornecer funcionalidade não pretendida

  (= despedício de trabalho).

Definição de requisitos funcionais

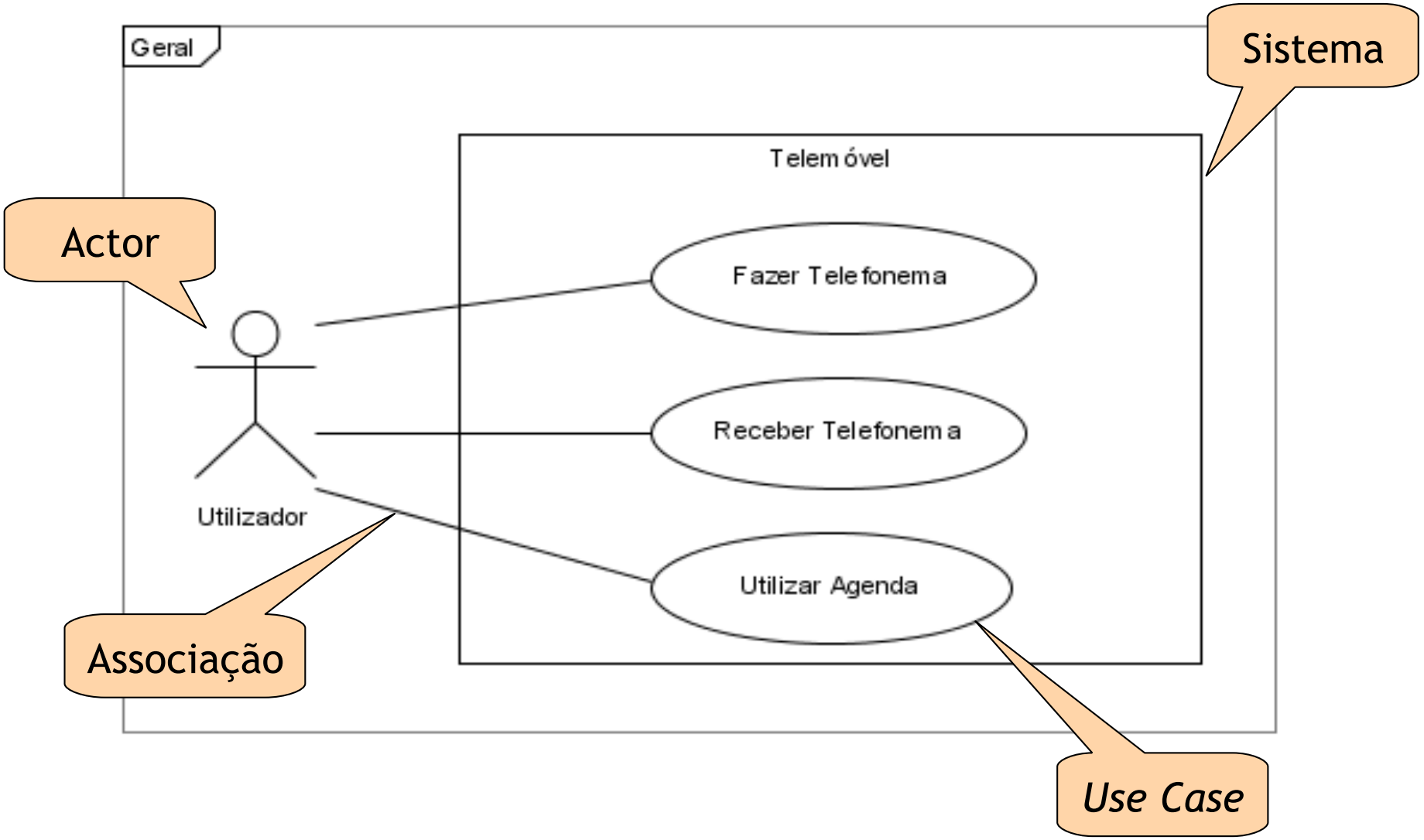
Visão orientada aos *Use Case*

- Identificar actores – quem vai interagir com o sistema?
- Identificar *Use Case* – o que se pretende do sistema?
- Identificar classes de suporte à realização dos *use case* – como vai a funcionalidade necessária ser implementada?

Vantagens:

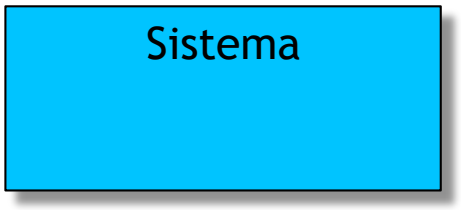
- Não há trabalho desnecessário;
- O Sistema de Informação suporta as tarefas do cliente.

Diagrama de *Use Cases*



Sistema

- define as fronteiras da solução a desenvolver



Actor

- uma abstracção para uma entidade fora do sistema
- um actor modela um propósito (alguém que tem um interesse específico no sistema) – pode não mapear 1 para 1 com entidades no mundo real
- um actor não é necessariamente um humano – pode ser um computador, outro sistema, etc.
- cada actor representa um papel ("role") que "alguém" ou qualquer "coisa" externa ao sistema pode assumir
- o conjunto de todos os actores definem todas as formas de interacção com o sistema

Associação

- representa comunicação entre o actor e o sistema – através de *use cases*
- pode ser bi-direccional ou uni-direccional

Definição de *Use Case*

- Uma unidade coerente de funcionalidade – um serviço
- define um comportamento do sistema sem revelar a estrutura interna – apenas mostra a comunicação entre sistema e actores
- o conjunto de todos os *use case* define a funcionalidade do sistema
- deve incluir o comportamento normal, bem como variações (erros, etc.)
  - vamos definir o comportamento com texto estruturado (para já);
  - vamos também definir as pré-condições e pós-condições de cada use case (cf. *design by contract*).

Design by contract

- *Design by contract* (DBC) baseia-se na noção de um contrato entre um cliente e um fornecedor para a realização de um serviço.
- O conceito central do DBC é a asserção (uma asserção é uma expressão booleana que nunca deverá ser falsa).
- Tipicamente as asserções são automaticamente testadas durante a fase de debug.
- O DBC identifica três tipos de asserções:
  - pré-condições – condições que se devem verificar para a invocação de um dado serviço ser válida;
  - pós-condições – condições que se devem verificar após a execução de um serviço;
  - invariantes – asserções que se devem verificar durante o tempo de vida da entidade a que se aplicam.
- A partir da versão 1.4 o Java passou a ter *asserts* que podem ser utilizados para definir pré- e pós-condições – no entanto não suporta invariantes .

USE CASES: Especificação

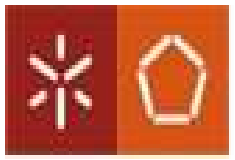
- Um UC é uma sequência de acções (um **fluxo de eventos**) que descreve a interacção entre um actor e um sistema, identificando entradas do actor e comportamentos do sistema (funcionalidade), que têm por objectivo máximo que o actor obtenha do sistema um resultado de valor (?), ou seja, realize com sucesso a tarefa pretendida.

- Um UC deve especificar um fluxo principal de sucesso, designado em geral por Main Flow ou Main Success Scenario, bem como todos os outros **fluxos alternativos** a este que conduzam ao sucesso do UC (designados Alternative Flows);

- Finalmente, um UC deve igualmente especificar fluxos de insucesso, e eventuais tentativas para recuperar tais fluxos, etc. Porém, e de forma clara, todos os fluxos de insucesso devem ser especificados como **fluxos de excepção** (cf. Exceptions).

USE CASES: cont.

- A UML não especifica, de facto, formatos particulares para a descrição textual dos UC, pelo que o formato deve ser definido pelas organizações.
- Assim, usaremos nas nossas definições textuais de UC escritas em Visual Paradigm, para além do **Main Flow**, os fluxos **Alternative** e de **Exception**;
- Embora o texto possa ser muito informal, é muito importante que a maior parte das entidades importantes façam parte do **Modelo de Domínio** e sejam referidas por identificadores coincidentes com os definidos em tal modelo (anteriormente desenvolvido ou a desenvolver em paralelo).



USE CASES

USE CASES TEXTUAIS:

- São textos simples que registam decisões conjuntas
- Fáceis de ler
- Idealmente não devem ter mais do que 10 passos no Main Flow
- Referem-se a entidades que fazem parte do Modelo do Domínio
- Situam-se ao nível dos objectivos do utilizador do sistema
- Não devem incluir formatos de dados
- Não especificam a Interface com o Utilizador

Resultam de tomadas de decisão conjuntas entre clientes, utilizadores e analistas, sendo documentos que fixam responsabilidades funcionais do sistema a desenvolver e servem para planear o projecto (tempo, $$), sendo ainda úteis na concepção e até na implementação.

O *use case* para fazer um telefonema:

Use Case: Fazer Telefonema
Pré-condição: Telefone ligado e em descanso
Comportamento normal:

1. Utilizador marca número e pressiona OK
2. Telefone transmite sinal de chamada
3. Utilizador aguarda
4. Telefone estabelece ligação
5. Utilizador fala
6. Utilizador pressiona tecla C
7. Telefone desliga chamada

Pós-condição: Telefone ligado e em descanso

Identificação de *Use Cases*

- Podemos identificar os *Use Case* do sistema a partir da identificação de cenários de utilização.
- Um cenário descreve um contexto concreto de interacção entre o utilizador e o sistema. Por Exemplo:

  Durante o semestre o Prof. Faísca foi enviando os sumários com breves resumos da matéria leccionada, via email, para o sistema Fly2. Após o fim das aulas, o Prof. Faísca utilizou a interface web do sistema para actualizar cada um dos sumários com descrições mais completas das matérias leccionadas. Finda essa actualização, imprimiu os sumários e enviou-os à Secretaria.

- A partir dos cenários podemos identificar os Use Cases (serviços) necessários à correcta disponibilização da funcionalidade requerida pelo mesmo.

Identificação de *Use Cases (II)*

No cenário anterior podemos identificar os seguintes *Use Case:*

1. Enviar sumários via email
2. Actualizar sumários via web
3. Imprimir sumários (via web? / via e-mail?)
4. Enviar sumários à secretaria – deverá este *use case* ser considerado?

   No cenário descrito o envio é feito em papel. Não se trata, portanto, de um serviço fornecido pelo sistema. No entanto, podemos discutir a possibilidade de o envio passar a ser feito electronicamente – estariamos a alterar o modo de trabalho inicialmente previsto/actual!

   Durante o semestre o Prof. Faísca **(1.) foi enviando os sumários** com breves resumos da matéria leccionada, **via email**, para o sistema Fly2. Após o fim das aulas, o Prof. Faísca **(2.) utilizou a interface web** do sistema **para actualizar cada um dos sumários** com descrições mais completas das matérias leccionadas. Finda essa actualização, **(3.) imprimiu os sumários** e **(4.)** enviou-os à Secretaria.

Diagramas de *Use Cases* – *revisão de conceitos*

- Modelam o contexto geral do sistema. Quais os actores que com ele se relacionam e que use case deve suportar.
- A concepção do sistema é guiada pelo modelo de *use case*:
  - Utilizam-se *use cases* para capturar os requisitos funcionais do sistema de uma forma sistemática;
  - O modelo de *use case* captura toda a funcionalidade requerida pelos utilizadores;
- A implementação do sistema é guiada pelo modelo de *use case*:
  - cada *use case* é implementado sucessivamente:
  - quando todos os *use cases* estiverem implementados obtém-se o sistema final;
  - fica facilitada a manutenção do sistema sempre que os requisitos sejam alterados;
- O modelo de *use case* é utilizado para o planeamento de testes:
  - Após a definição do modelo de *use case*: planear *black-box testing*.
  - Após a implementação dos *use cases*: planear *white-box testing*.

Black-box testing

- Utilizado para verificar se o sistema implementa toda a funcionalidade pretendida.
- Permite detectar erros de “omissão” (funcionalidade não implementada).

White-box testing

- Utilizado para verificar se o sistema implementa a funcionalidade de forma correcta.
- Permite detectar erros na implementação da funcionalidade pretendida.

Identificação de Use Cases

Etapas a cumprir (com o auxílio de cenários de utilização do sistema):

1. Identificar actores
2. Identificar *use cases*
3. Identificar associações

Identificar actores

- Quem vai utilizar o sistema?
- Neste caso: Docente, Secretaria, Servidor Email?, WebApp?

Identificar use cases

- Objectivos dos utilizadores/actores?
- Resposta a estimulos externos.

Identificar associações

- Que actores utilizam que *use cases*?
- Nem sempre é imediatamente evidente se a comunicação entre o sistema em análise e sistemas externos deve ser representada. Quatro abordagens podem ser identificadas:
  - ✗ mostrar todas as associações;
  - ✗ mostrar apenas as associações relativas a interacção iniciada por sistemas externos;
  - ✔ mostrar apenas as associações relativas a interacções em que é o sistema externo o interessado no *use case*;
  - ✗ não mostrar associações com sistemas externos.

Todas as associações

- Todos os sistemas externos que interagem com o sistema em análise são apresentados como actores e todas as interacções são representadas nos diagramas.
- Demasiado abrangente, em muitos casos existem interacções com outros sistemas apenas por razões de implementação e não por se tratarem de requisitos do sistema.

Apenas as associações relativas a interacção iniciada por sistemas externos

- Só são representados como actores os sistemas externos que iniciem diálogo com o sistema em análise.
- Mesmo assim muito abrangente.

Apenas as associações em que é o sistema externo o interessado

- Neste caso só são apresentados como actores os sistemas externos que necessitam de funcionalidade fornecida pelo sistema em análise.
- Usalmente esta é uma solução equilibrada.

Não mostrar associações com sistemas externos

- Apenas os utilizadores são actores, neste caso quando existem sistemas externos apresentam-se os seus actores em diálogo directo com o sistema a ser modelado.
- De uma outra forma esta solução também é demasiado abrangente e pode levar a confusão sobre quem está realmente a utilizar o sistema.

Diagramas de *Use Case* - Resumo

- Os diagramas de *Use Case* permitem definir os requisitos funcionais de um sistema:
  - que serviços deve fornecer;
  - a quem os deve fornecer.
- Notação diagramática facilita o diálogo (com os clientes e dentro da quipa de desenvolvimento).
- Especificação textual é essencial (explorar alternativas e variações).
- Utilizando diagramas de *use case*, clientes e equipa de desenvolvimento podem chegar a um acordo sobre qual o sistema a desenvolver.
- A resolução de alterações nos requisitos funcionais fica facilitada.

No entanto:

- Os diagramas de *use case* não suportam a captura de requisitos não funcionais.

Quando utilizar diagramas de Use Case?

- Sempre que se estiverem a analisar requisitos (funcionais)!

Modelação do Requisitos Funcionais

Sumário:

- Requisitos funcionais vs. requisitos não funcionais
- Definição de requisitos funcionais
- Diagramas de Use Case: notação básica
- Definição de Use Case
- Representação textual de Use Cases
- Identificação de Use Cases e Actores