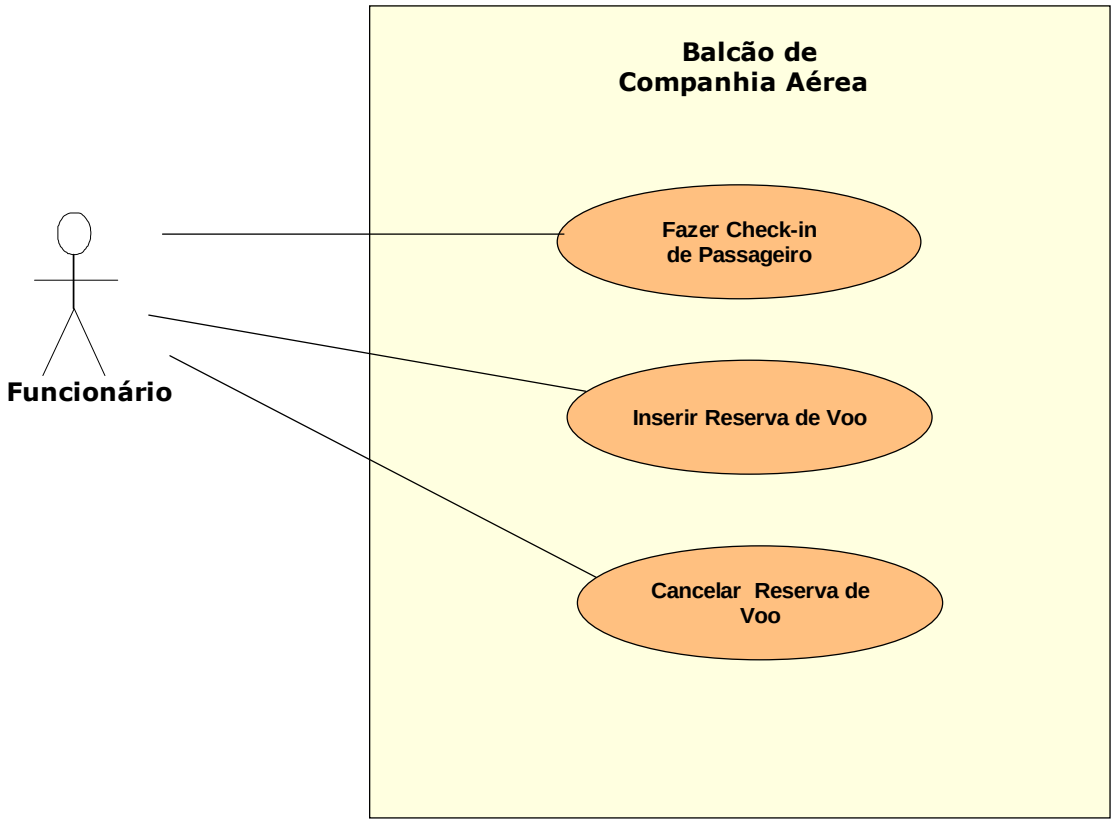
# USE CASES: continuação

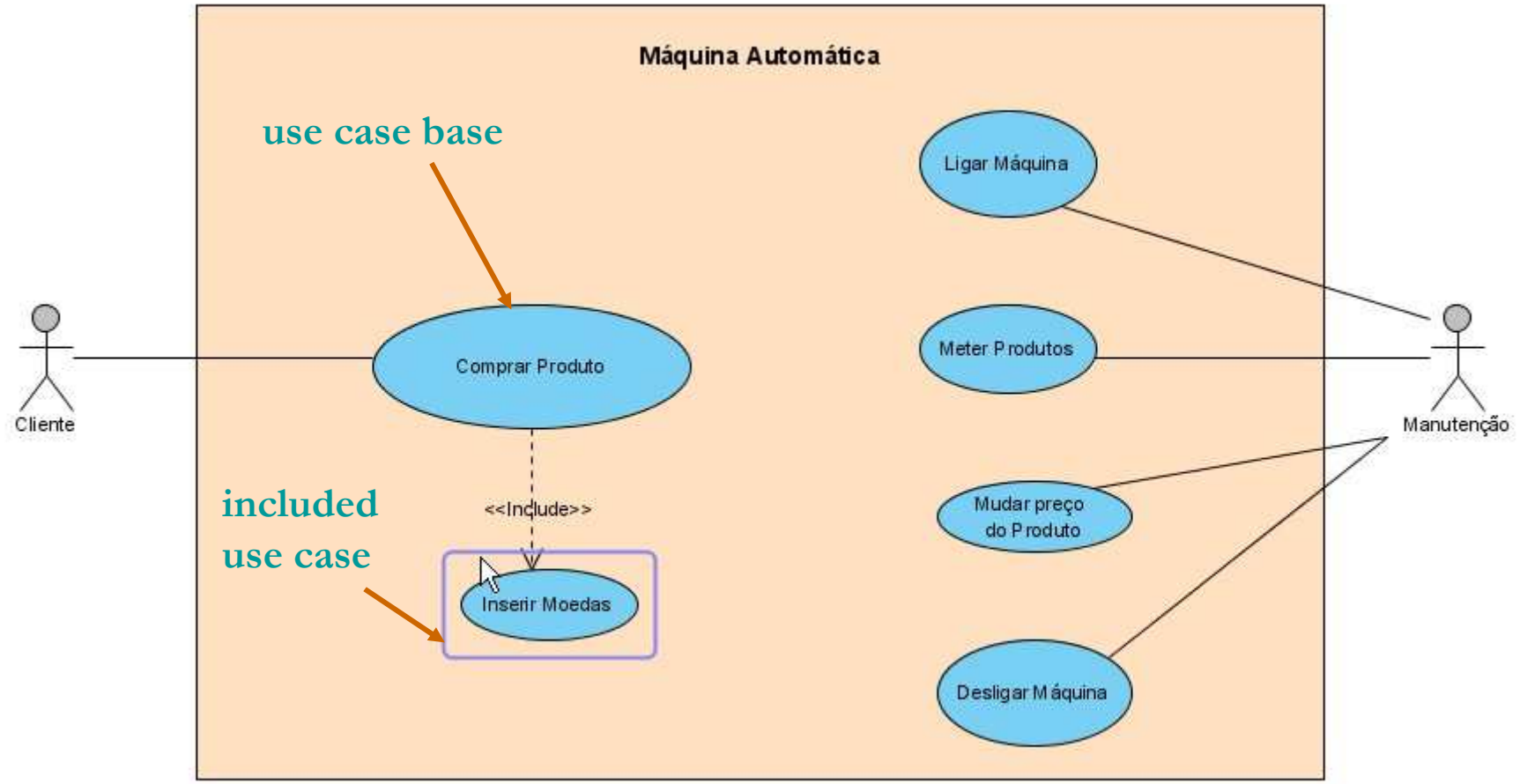



Os primeiros diagramas de Use Case (DUC) de um Sistema, descrevem apenas a funcionalidade total "esperada" do sistema, sem detalhar possíveis estruturações de use cases, nem ordem das acções, etc.

# USE CASES: Importância

- São uma forma intuitiva e sistemática de capturar requisitos funcionais (o que o sistema deve oferecer aos seus utilizadores);
- São a base (ou o centro) de todo o processo de desenvolvimento;
- Permitem identificar melhor as tarefas que são os objectivos dos utilizadores do sistema;
- Identificam o que o sistema deve fazer para cada tipo de utilizador;
- Especificam todas as possíveis utilizações do sistema;
- São instrumentos de diálogo entre clientes e projectistas;
- Permitem até desenvolver protótipos da Interface com o Utilizador.

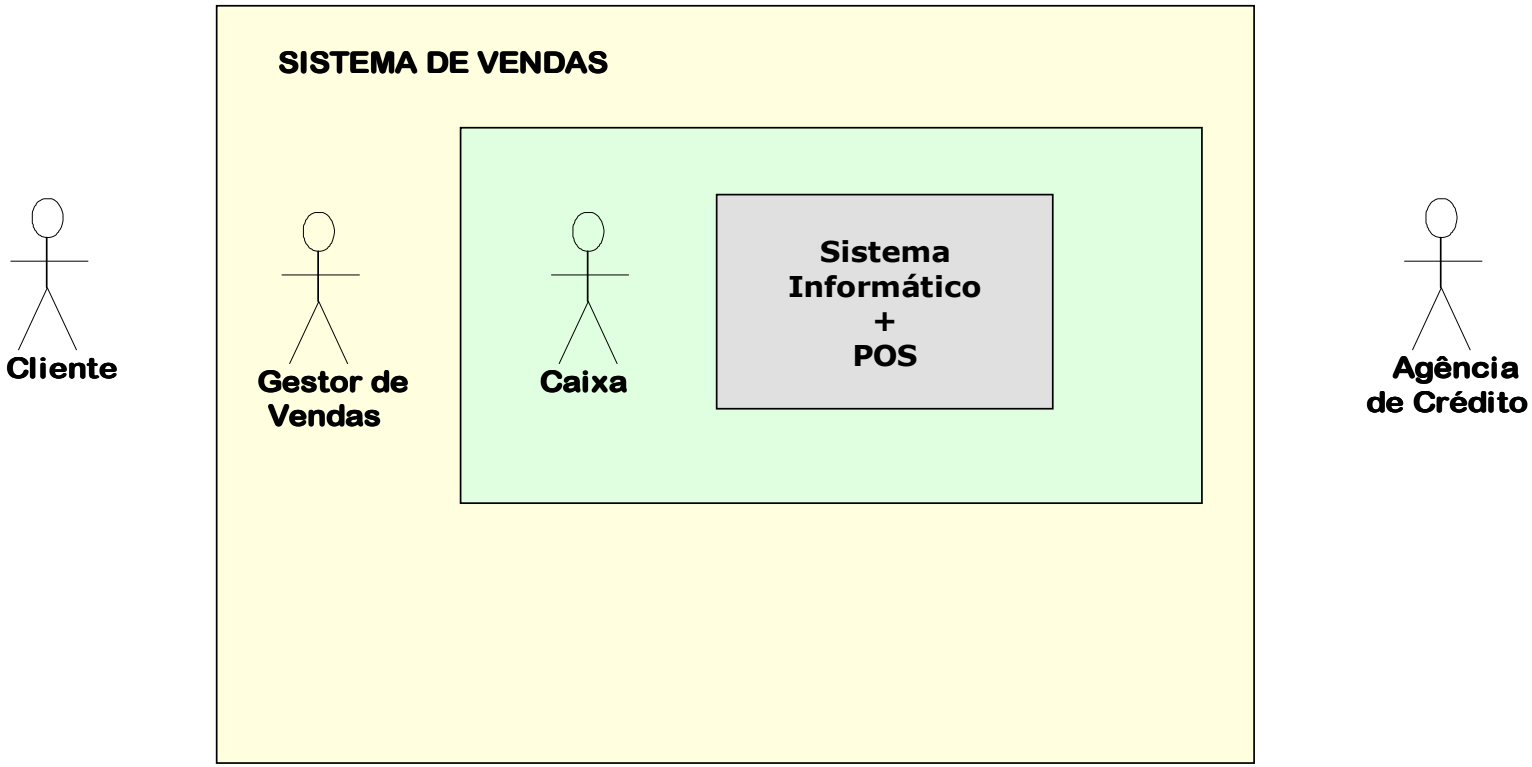
# USE CASES: cont.




Só depois, gradualmente, e em função do que se vai compreendendo e decidindo, é que se introduz mais detalhe, relacionando UCs entre si, reutilizando UCs, etc.

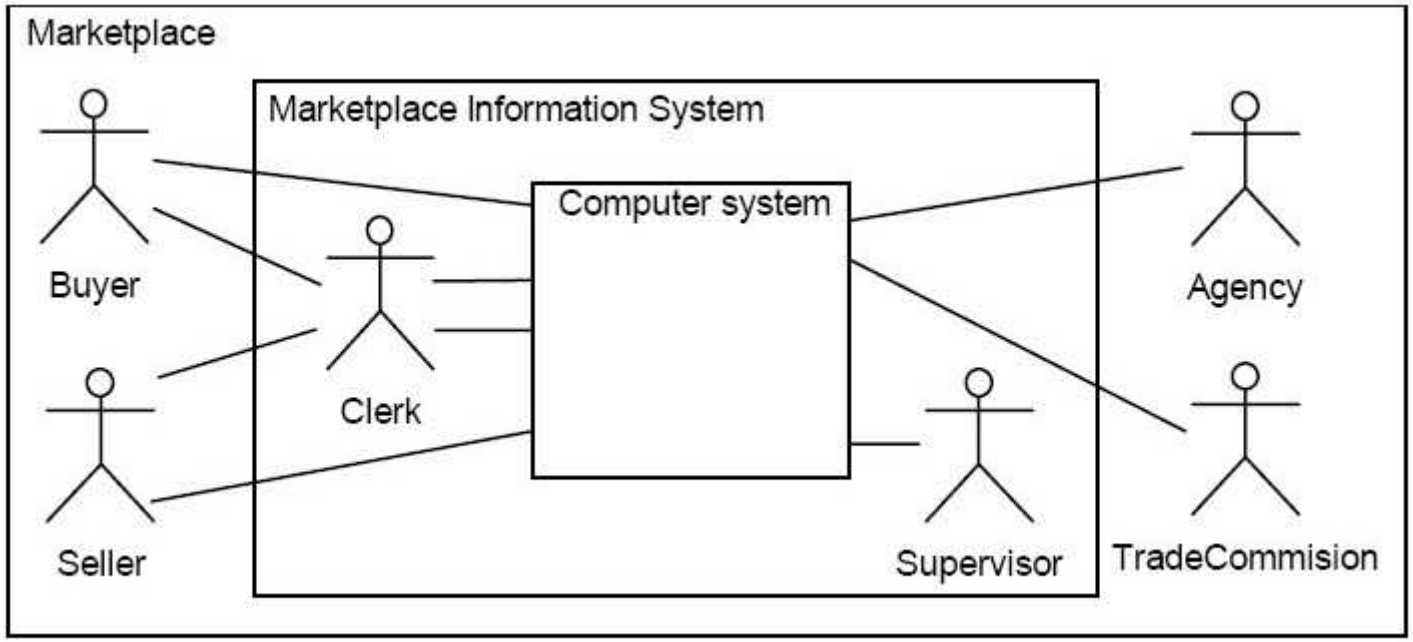
# USE CASES: continuação

## CONTEXTO



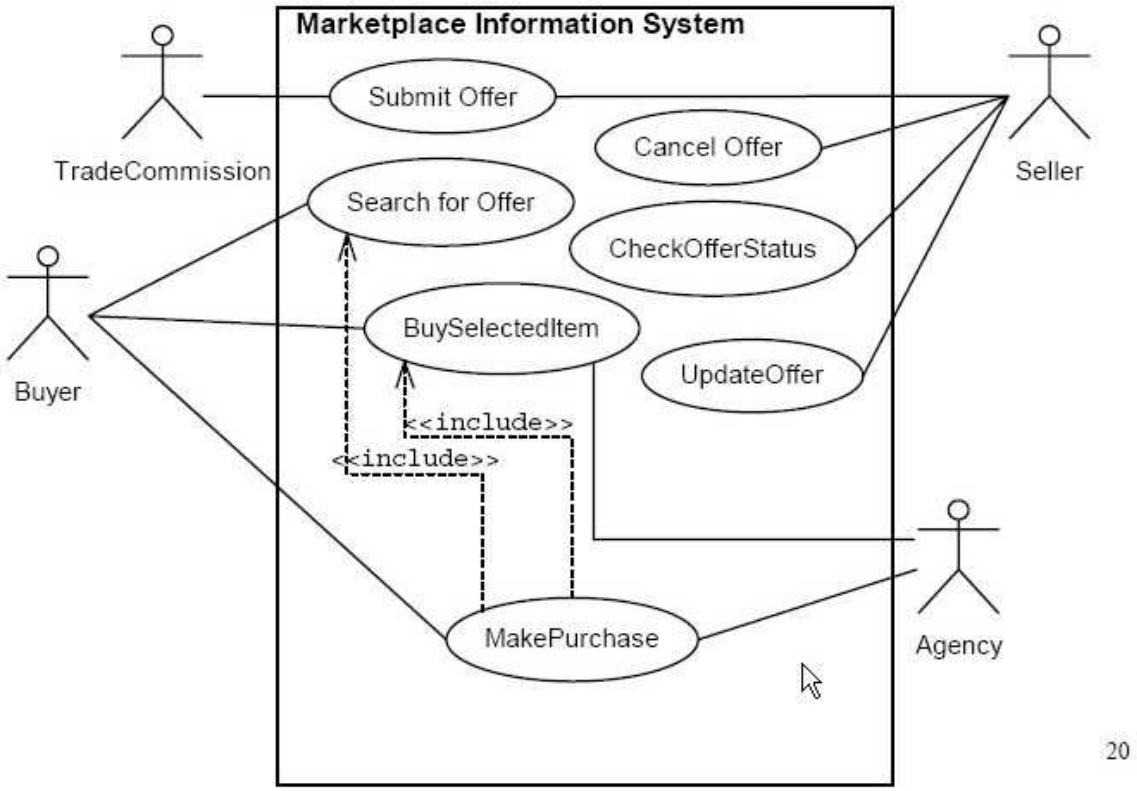

É importante ter sempre uma ideia de qual o contexto global ("scope") de funcionamento do Sistema Software que vamos desenhar e construir, e de "quem" está envolvido e porquê.

## DIAGRAMA DE USE CASES DE CONTEXTO



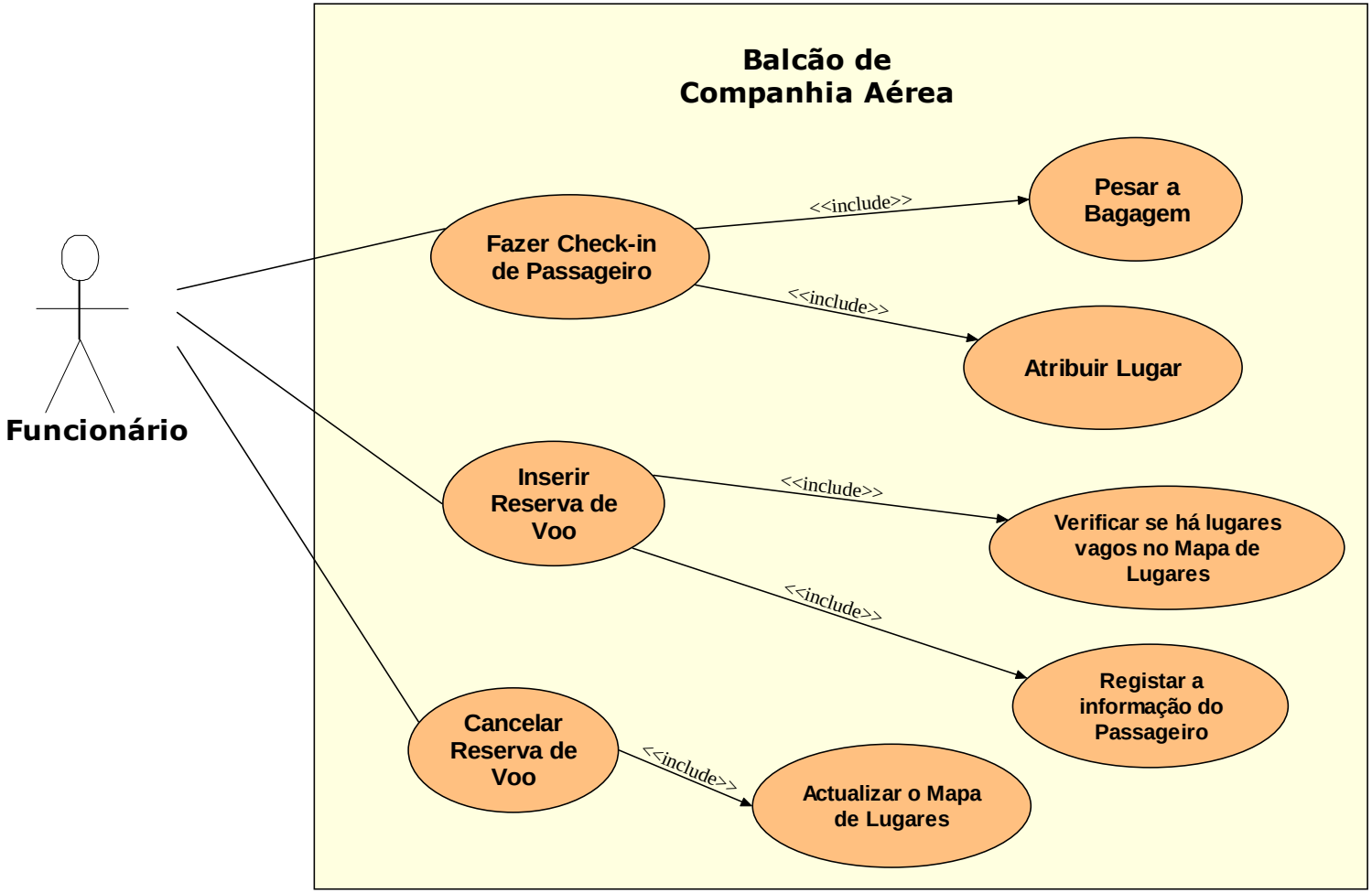

Por vezes, desenham-se DUC que envolvem várias partes da organização, para que se compreenda melhor o contexto do desenvolvimento do Sistema Sofware: Bolsa, Sistema geral de Informação da Bolsa, Sistema Sofware da Bolsa e Agências, etc.

# DIAGRAMA DE USE CASES DE CONTEXTO




**Por vezes define-se o DUC de todo o Sistema de Informação necessário para a organização funcionar, e apenas depois se decide o que irá ser informatizado (ou seja vai fazer parte do Sistema Software) !**

# USE CASES




**Precisamos de agora estudar mais formas de relacionamento entre UCs e entre Actores.**

# USE CASES: Relacionamentos

**Estereótipos pré-definidos para UCD** (extensões aos conceitos base, por vezes para tornar mais clara a sua semântica; existem pré-definidos; podem ser criados pelo modelador; são por vezes um enriquecimento semântico para o programador; palavra entre << .. >>):

**<<include>> e <<extend>> são 2 relacionamentos entre Ucs.**

---

**<<include>>**

Os "including" use cases não são opcionais, pois são sempre necessários para a correcta execução do "use case" base !

Os "including" use cases devem ter sempre sucesso, ou seja, realizar a sua tarefa.

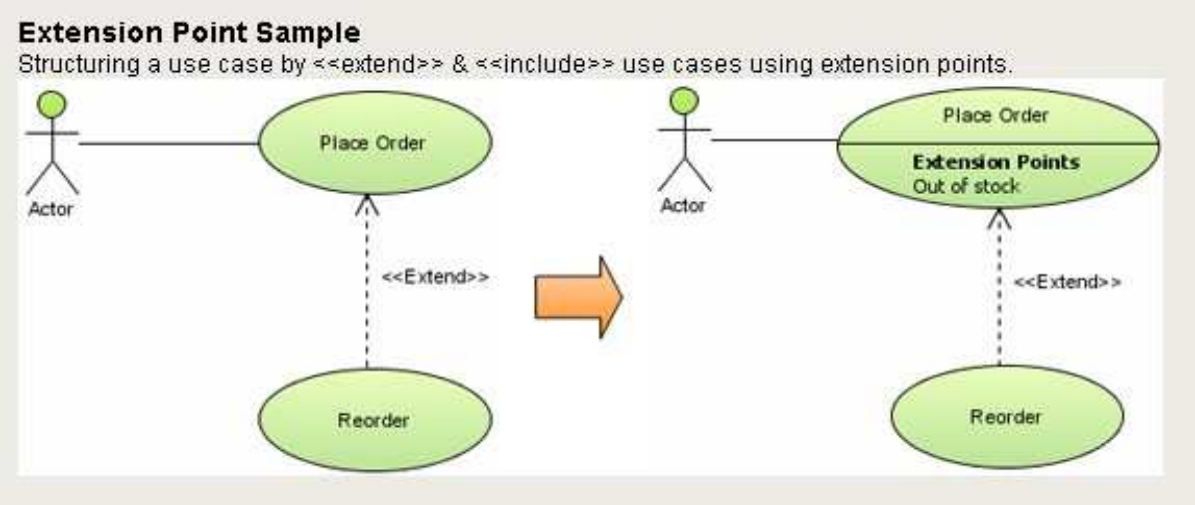
---

# USE CASES: Relacionamentos

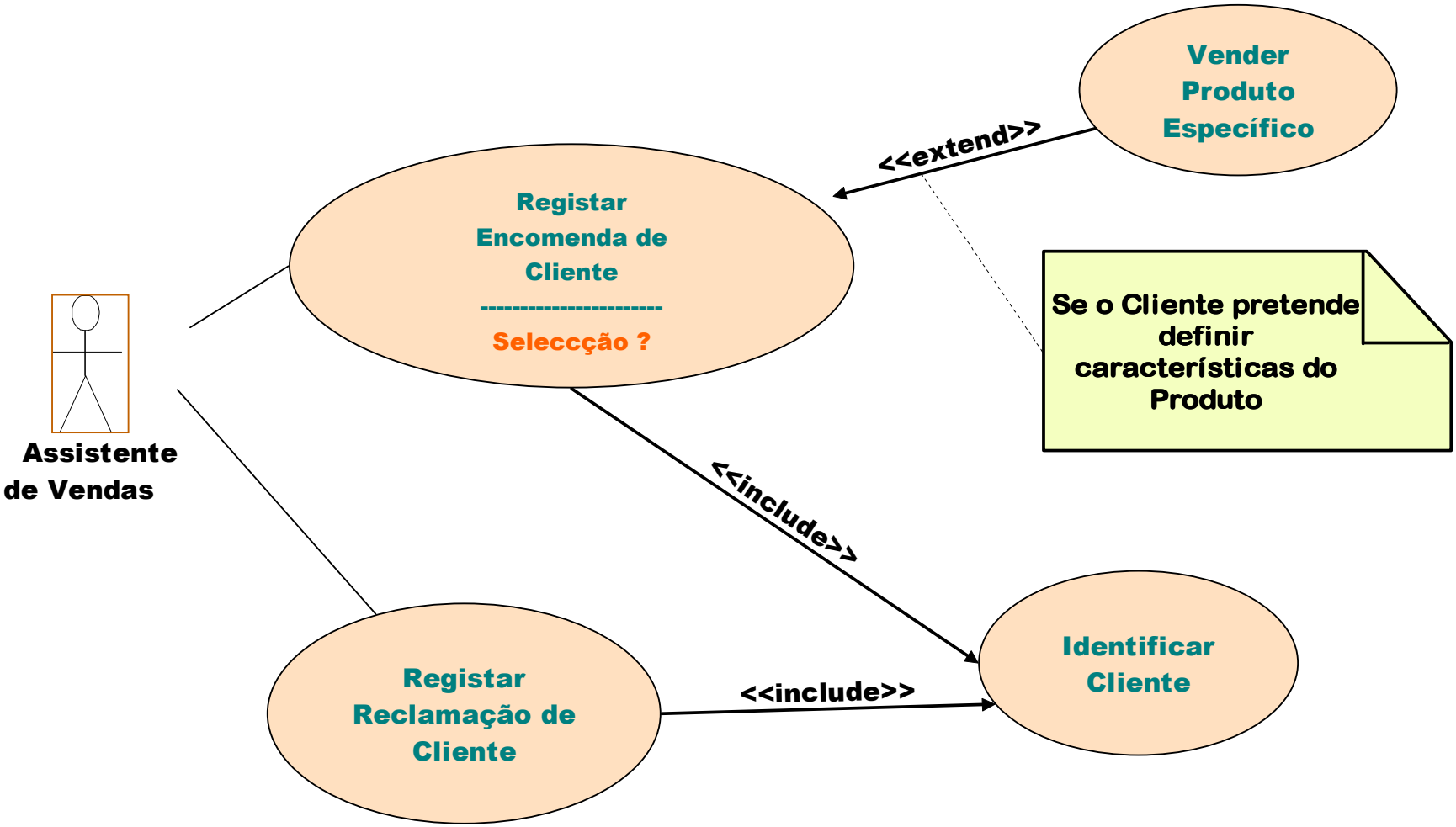
## <<extend>>

Os "extending" use cases servem para, em dadas condições, ou seja, condicionalmente, acrescentarem funcionalidade ao "use case" de base (em função das **condições de extensão**).

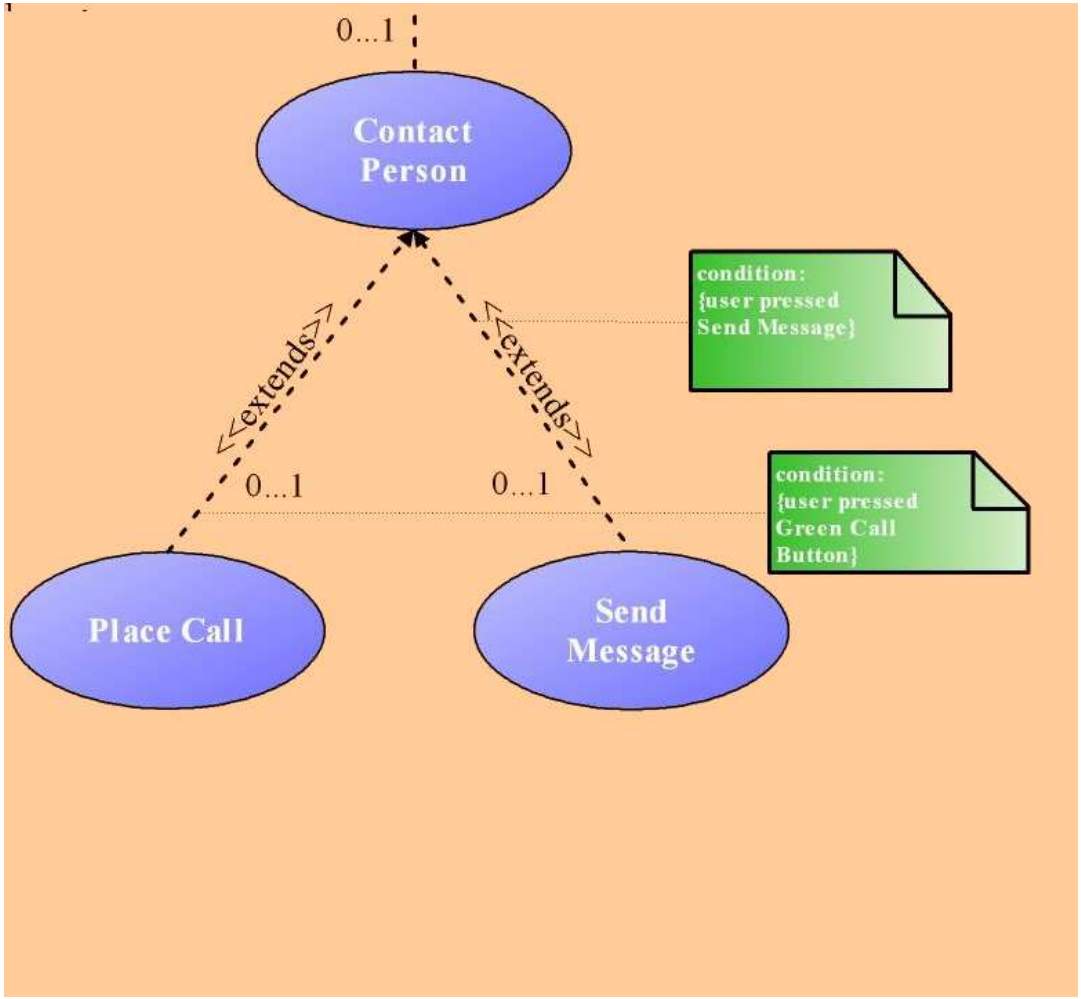
Mas, o use case de base deve possuir um comportamento que, mesmo que nenhuma extensão seja adicionada, seja um comportamento significativo e relevante, regra que nem sempre é seguida !!



# USE CASES: Relacionamentos



# USE CASES: <<extend>>

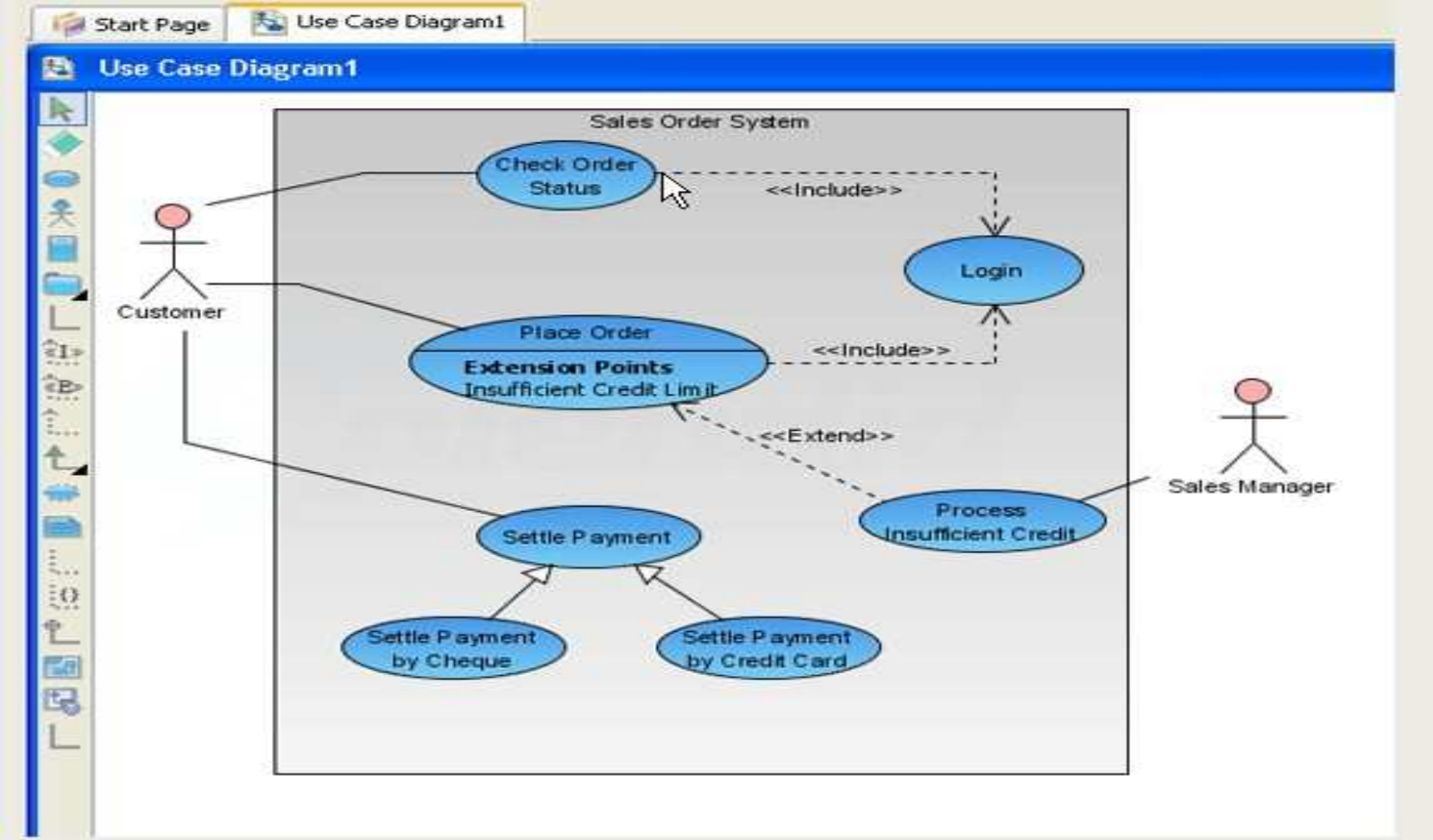



**Se as condições forem disjuntas os use cases de extensão são alternativas, podendo acontecer que nenhum seja activado, cf. 0..1**

Notar que o use case de extensão é que aponta para o use case de base !!

**O use case base que é aumentado, deverá ter uma funcionalidade própria, significativa e independente das "potenciais" extensões**

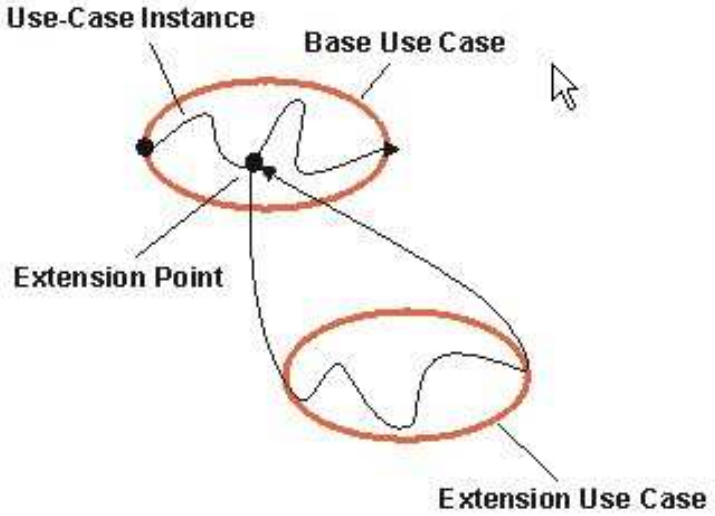
# USE CASES: Extension Points

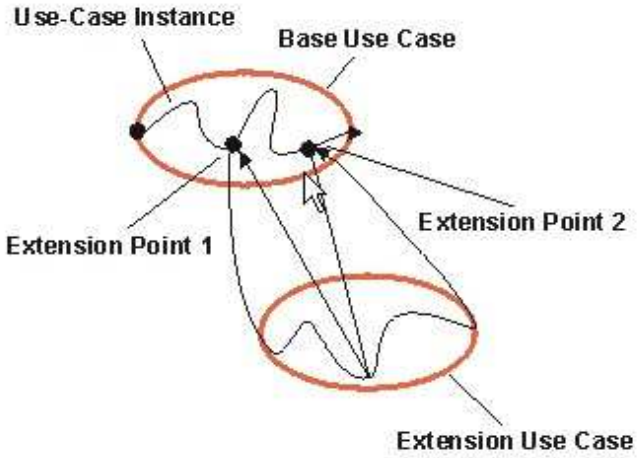



Um "extended" Use Case, pode ter o seu comportamento aumentado ou não, dependendo das condições definidas nos designados "extension points".

# USE CASES: Extension Points

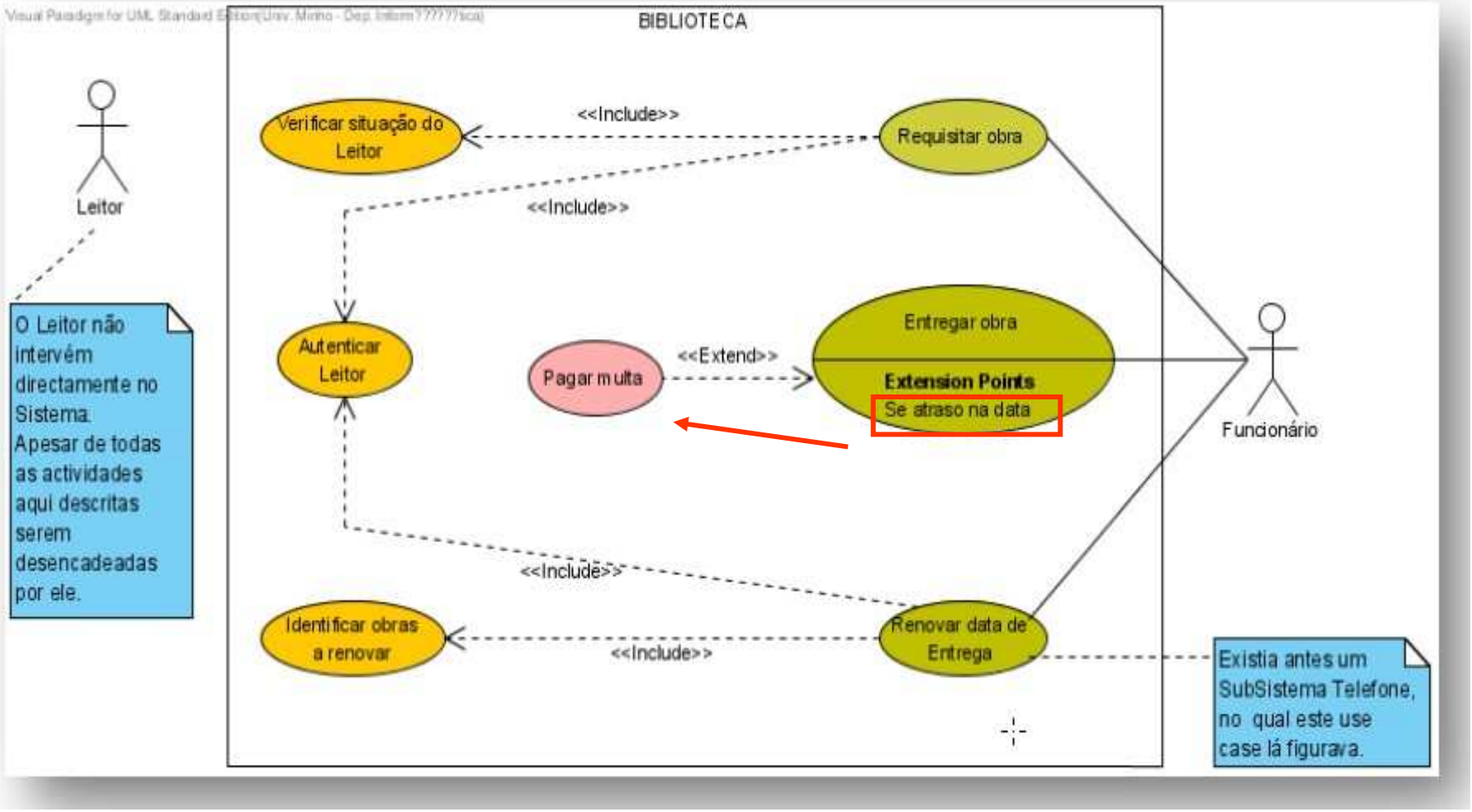
## Semântica operacional correcta de <<extend>>



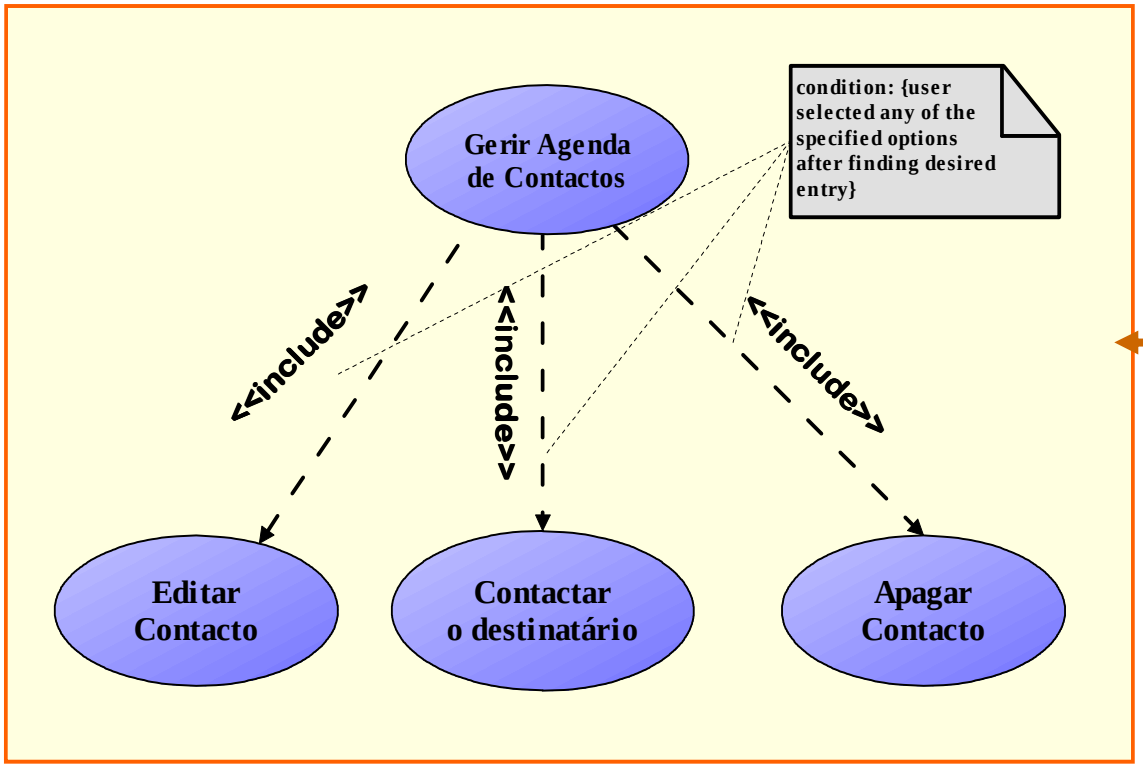




Se as condições de teste de extensão especificadas nos “extension points” forem verdadeiras, então serão realizadas as operações definidas nos respectivos use cases de extensão.

# USE CASES: <<extend>>



## Utilizações erradas de <<include>>

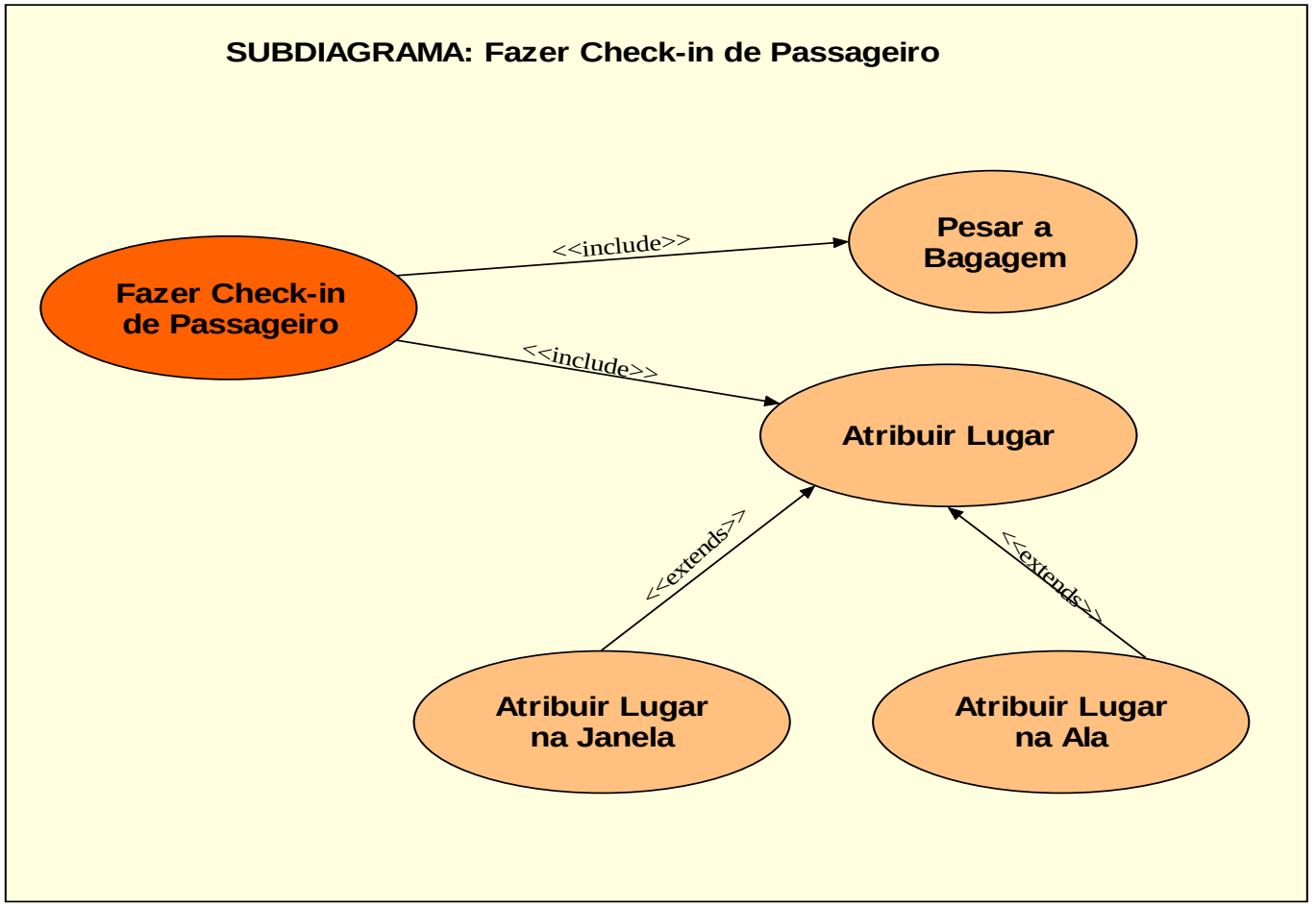



**Pretendido:**

O "use case" base pode ser aumentado por um qualquer dos sub use-cases, mas não os inclui a todos, podendo mesmo não incluir nenhum.

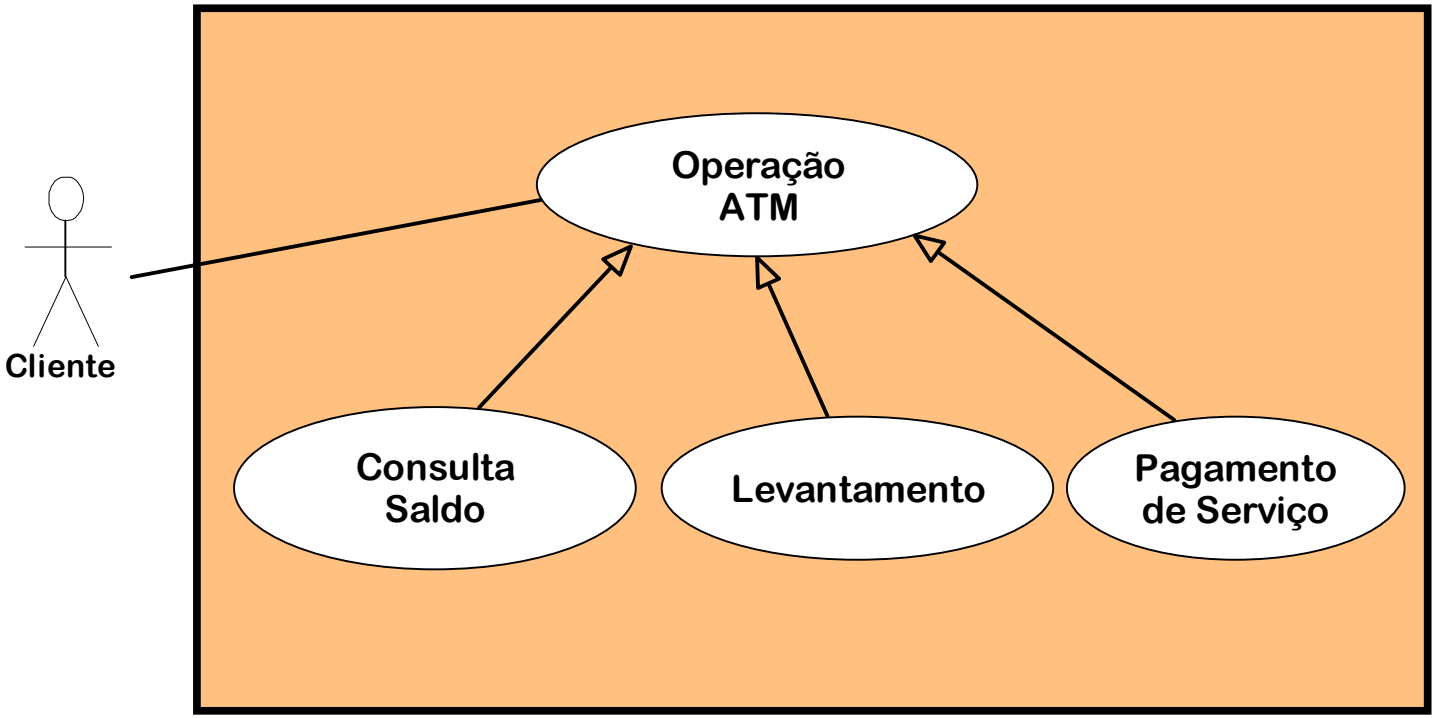
▣ O que está especificado não é isto, mas sim que o "use case base" deve incluir TODOS os sub. A "nota" e as setas estão erradas. Atenção !!

# USE CASES: Subdiagramas



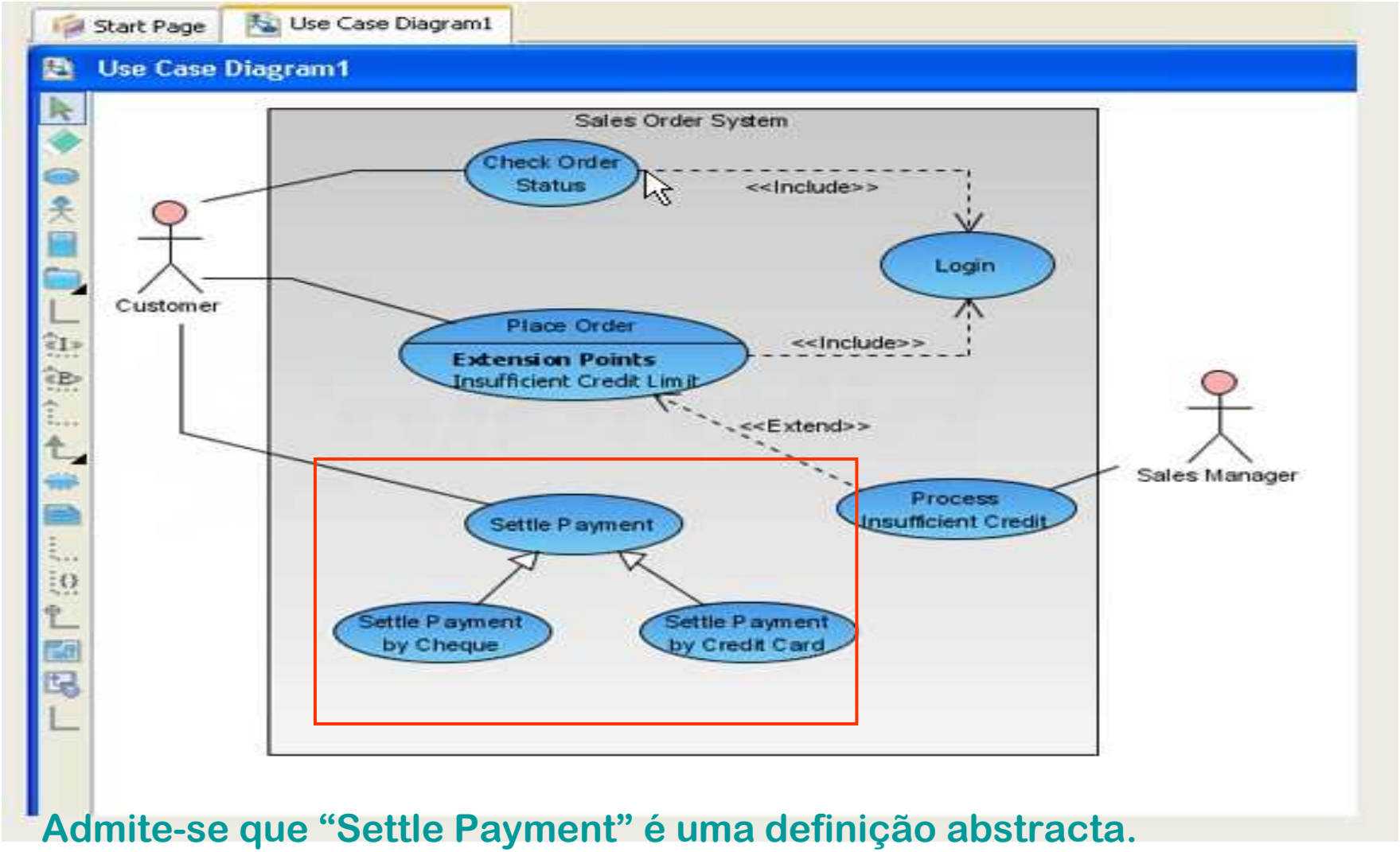

Podemos "esconder" os detalhes usando subdiagramas (modularidade e abstracção).

# USE CASES: Generalização



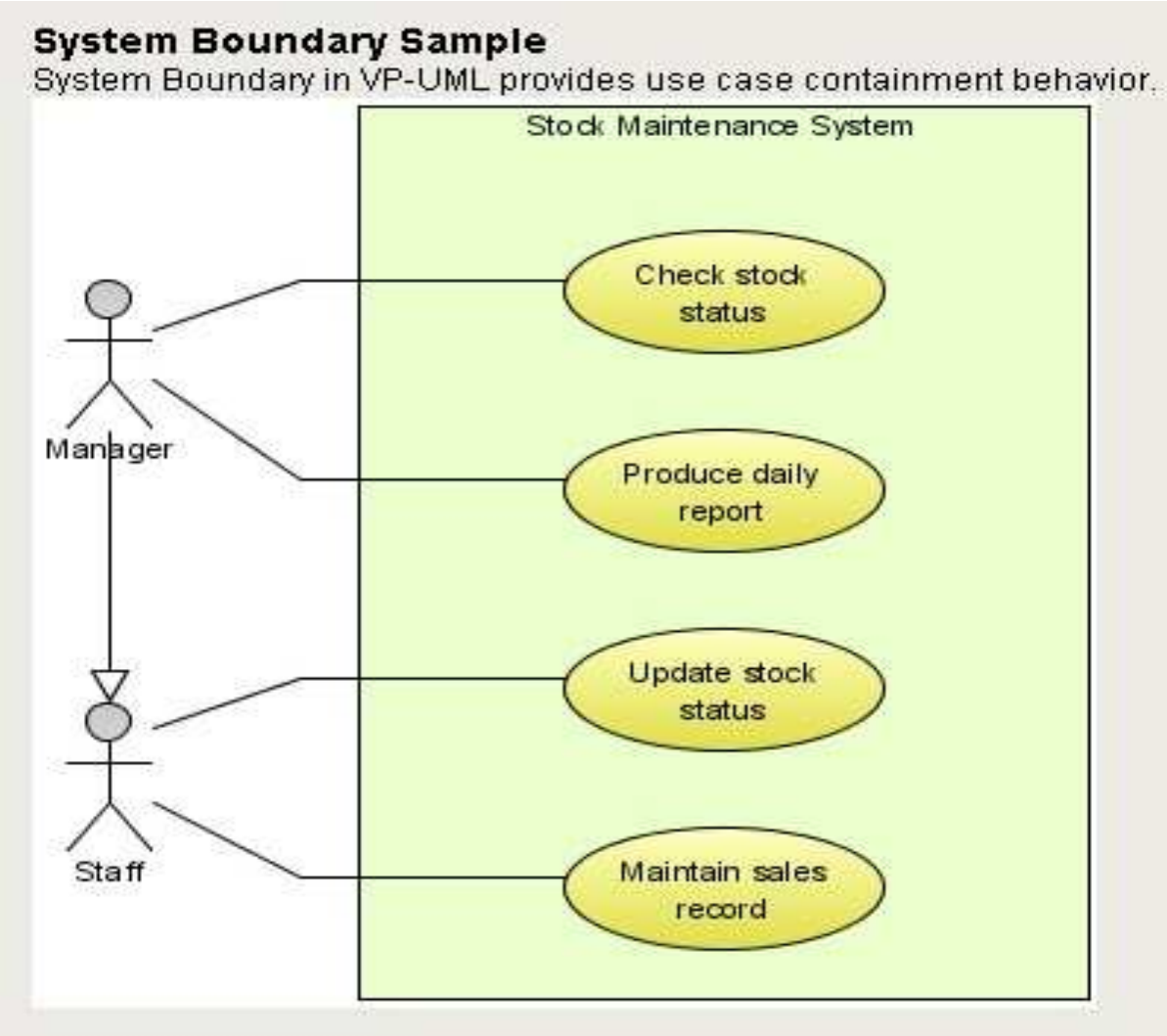

▣ **Generalização** permite expressar relacionamento **is-a**, subclassificação, herança e polimorfismo. O use case base pode assumir uma qualquer das formas (comportamentos) expressas nos sub-use cases (tal como indica o princípio da substituição em OO).

# USE CASES: Generalização




**Admite-se que "Settle Payment" é uma definição abstracta.**
**Mas o polimorfismo de Use Cases é DESACONSELHADO !!**

# ACTORES: Generalização




## Generalização de Actores

Corresponde a uma clarificação de papéis e responsabilidades perante o sistema ou subsistema.

Assim, quem assume certo "perfil" tem acesso a e tem responsabilidade sobre certas actividades

# ACTORES: Generalização

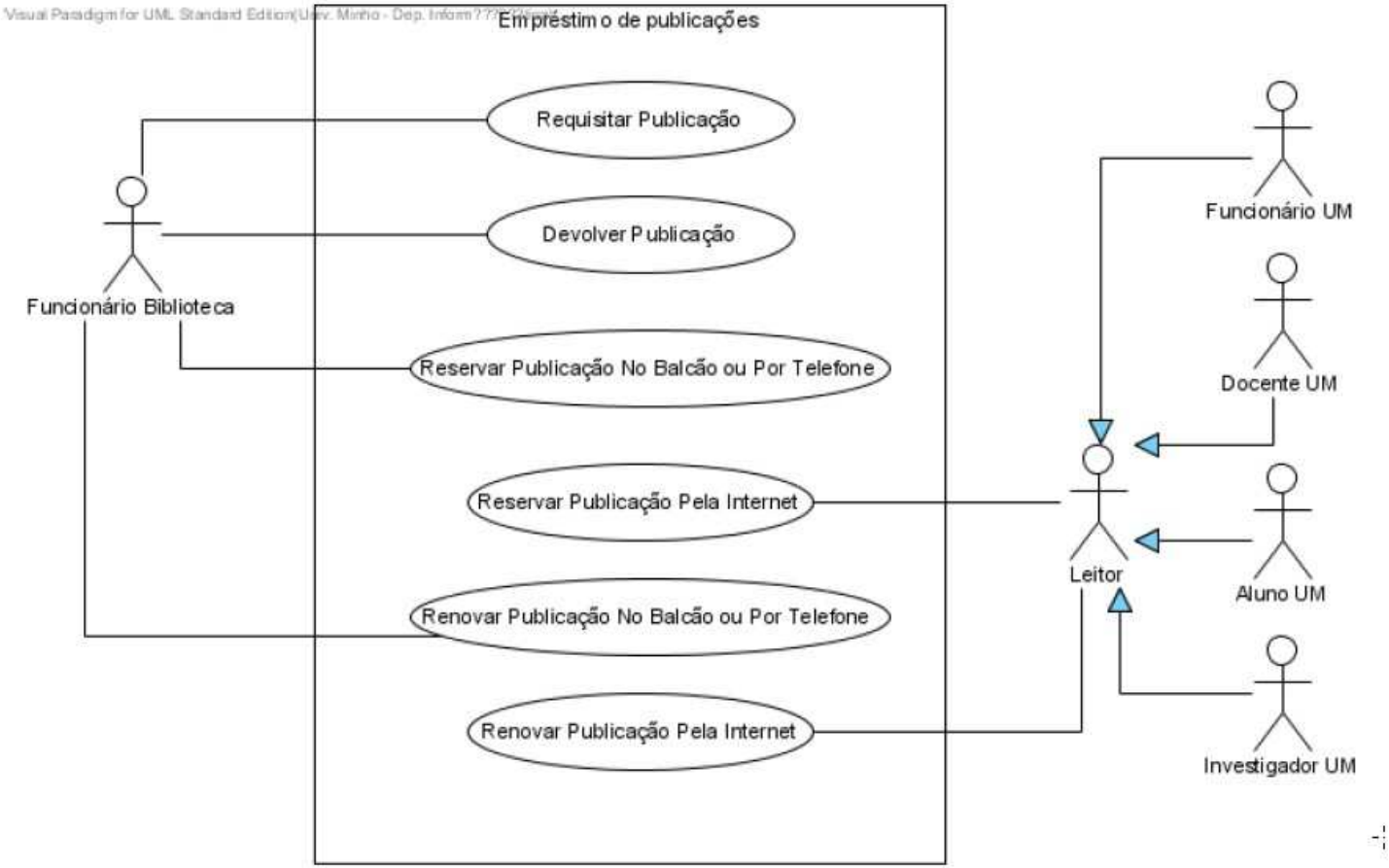
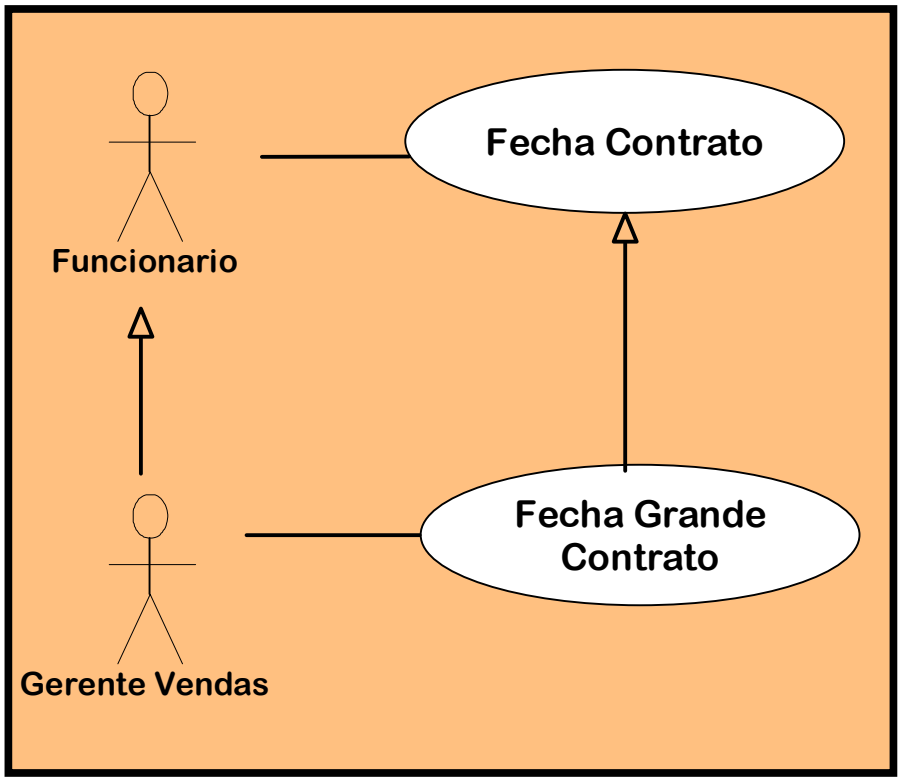



***Figura 1:*** Diagrama de Use Cases do Sistema de Empréstimo de publicações dos SDUM

**Diversos TIPOS de LEITOR, mas com os mesmos objectivos.**

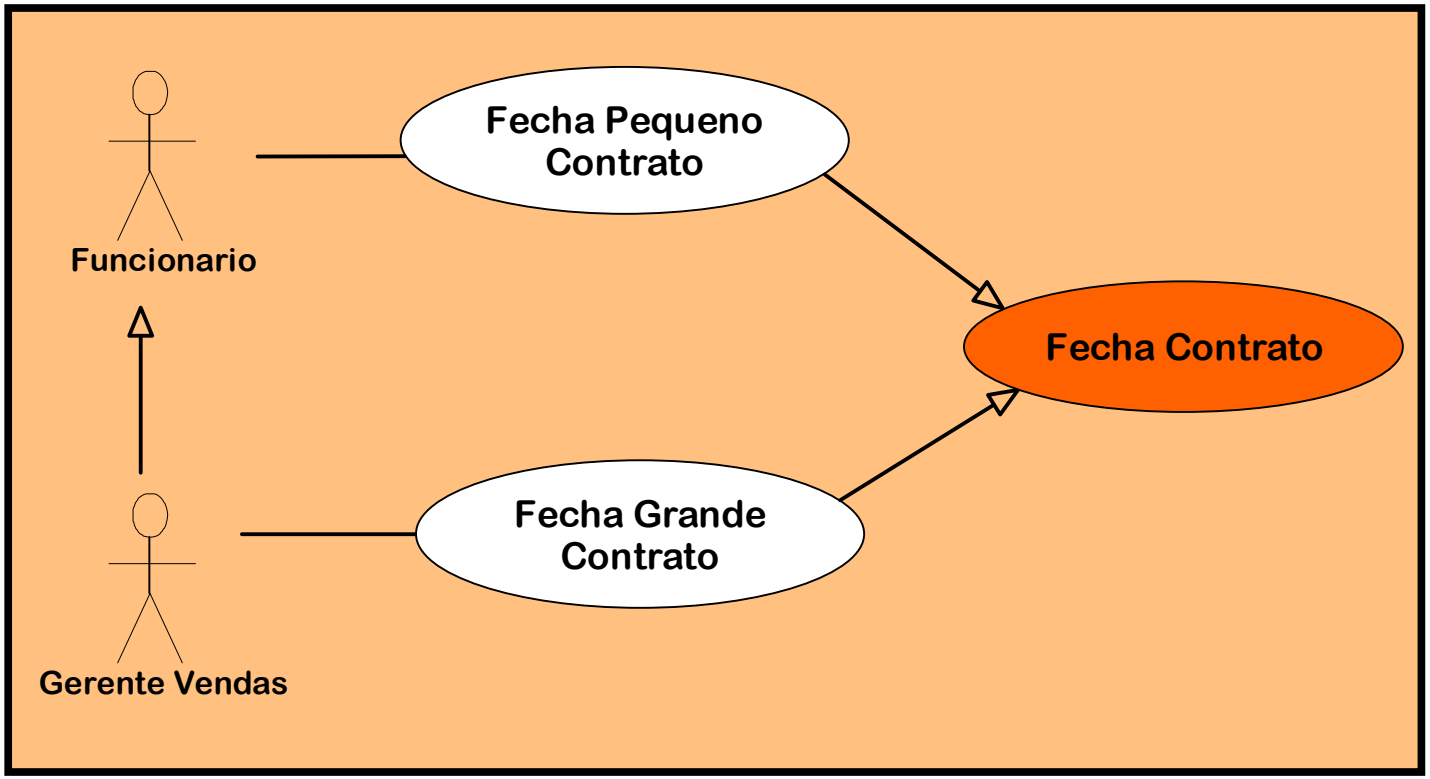
## Problemas com a generalização de Actores e UCs




Herança => Polimorfismo, pelo que "Fecha Contrato" é substituível por "Fecha Grande Contrato". Assim, o simples Funcioário pode fechar grandes contratos, tal como o Gerente.

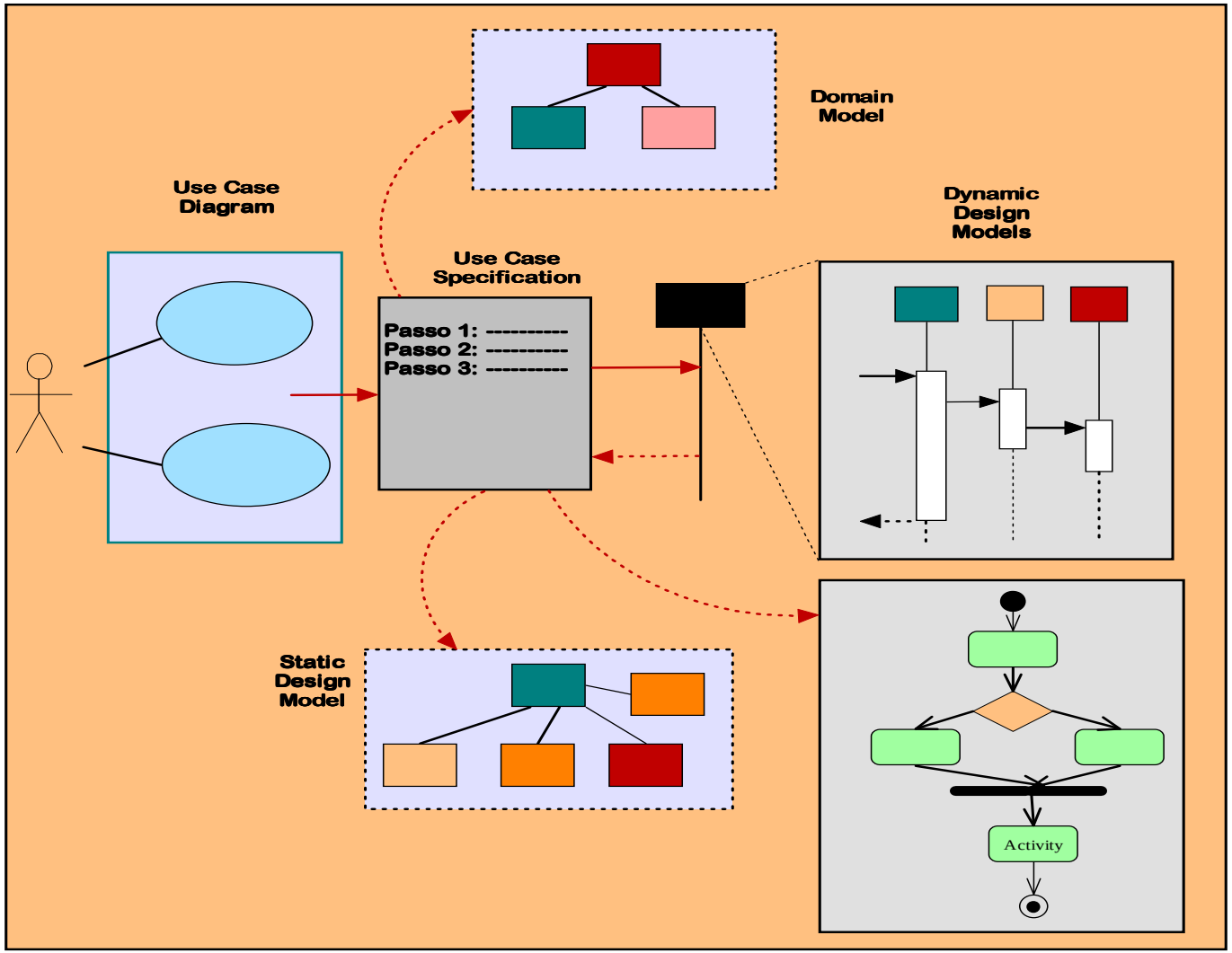
Não era certamente isto que se pretendia especificar !

## Solução (Generalização das tarefas com "profiling" adequado):



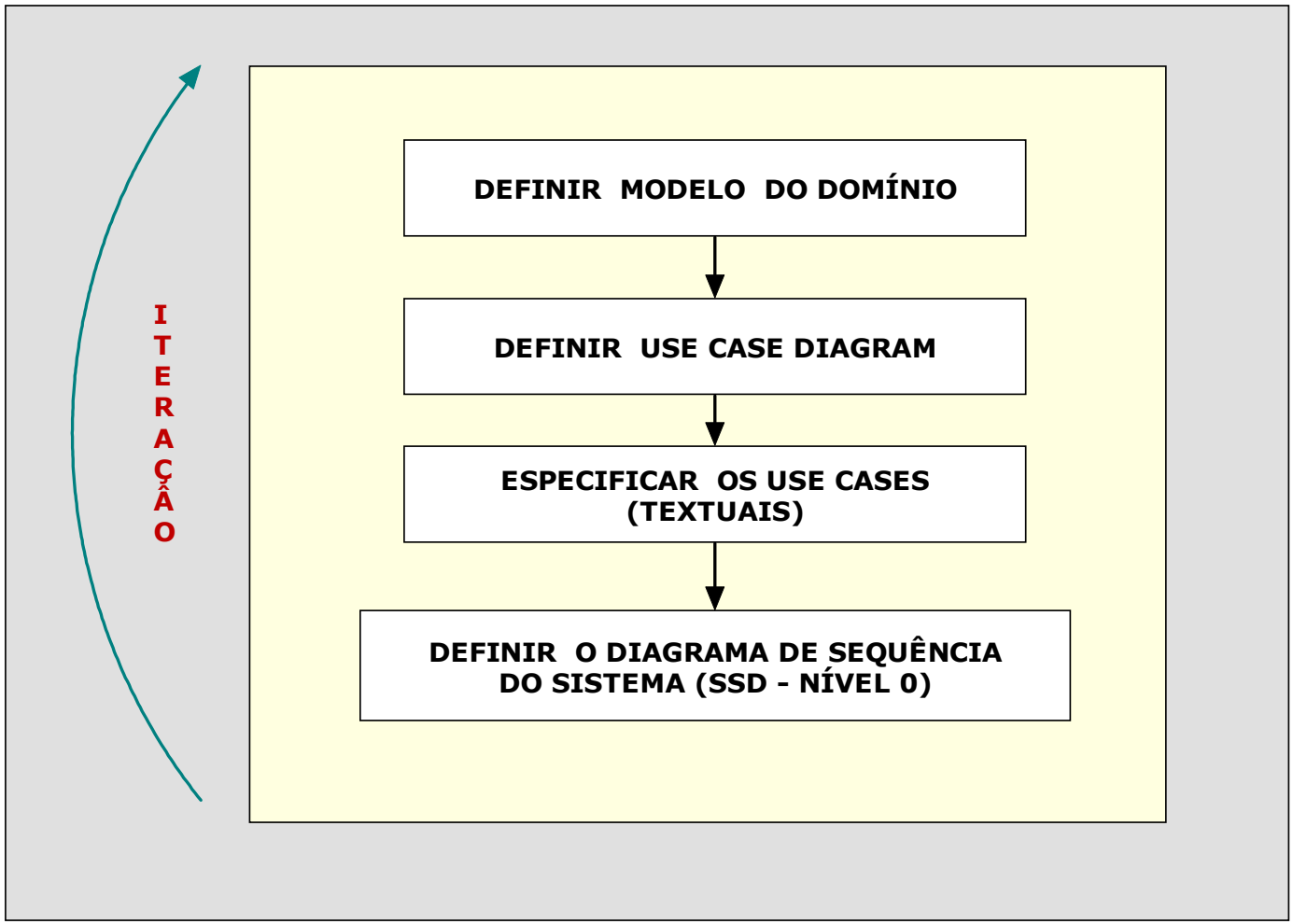

Situação pouco comum em que se justifica generalização de UC.

# METODOLOGIA




Use Cases são vistos como os elementos centrais do processo de análise (cf. Jacobson e RUP)

# Requisitos funcionais



# Modelos do Domínio

## O que é um Domínio ?

O termo **Domínio** denota, em Engenharia Informática, mas não só, um conjunto de sistemas ou áreas funcionais, dos vários sectores organizacionais de actividade, ou da sociedade em geral, onde a existência de uma terminologia (**sintaxe ou termo**) deve estar inequivocamente associada a certos conceitos, o que em muito simplifica a compreensão e, em consequência, a correcta comunicação entre quem tem que "definir contractos de informatização" e quem tem que "realizar tais contractos" sno âmbito (domínio) de tais sistemas, áreas, etc.

## Alguns Domínios em que a Engª Informática tem produzido "produtos":

Quase todos ou todos. Quem indica excepções ?

## Exemplos de Domínios típicos do âmbito da Engª Informática:

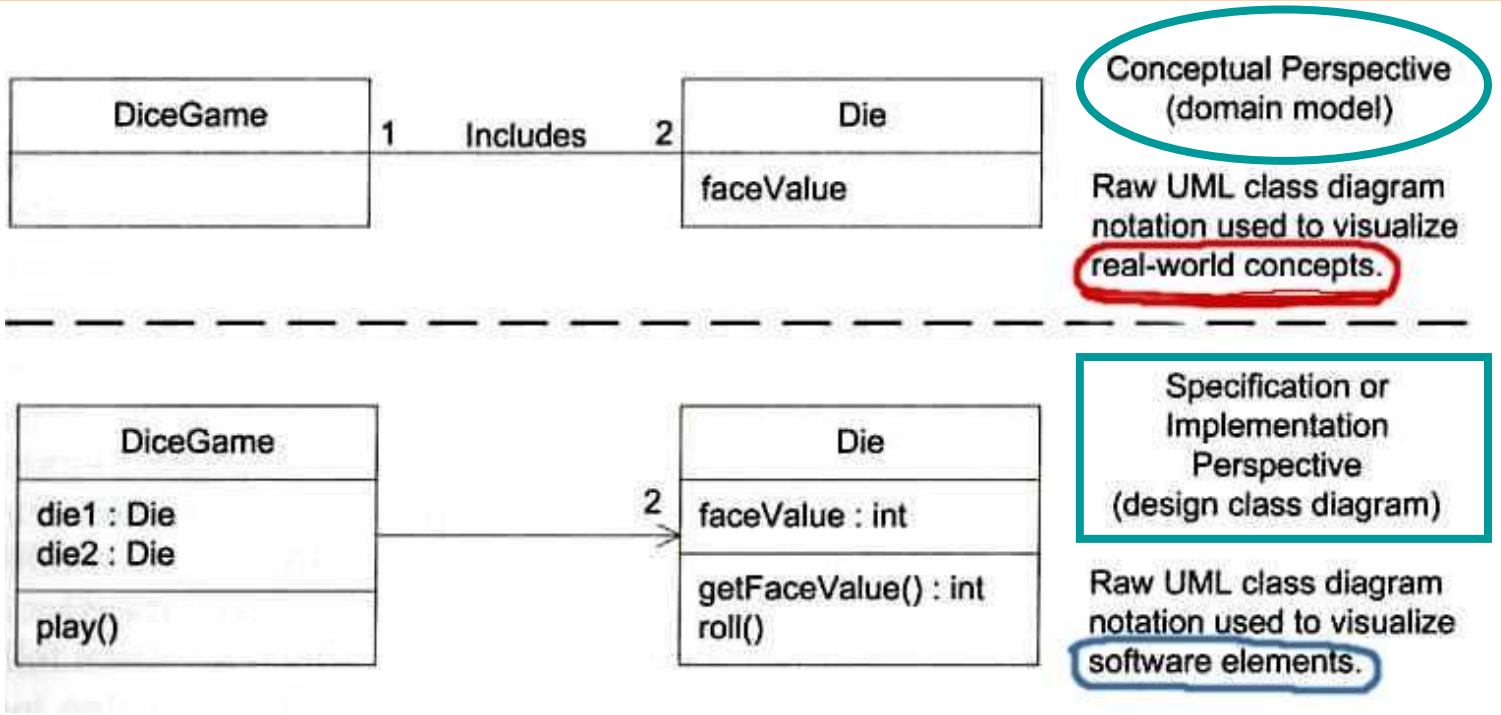
Bancário; Aviónica; Construção Civil; Administração Pública; Telecomunicações; Medicina e Clínica Médica; Biotecnologia; Segurança, enfim, todas as outras Engenharias e, actualmente, todas as outras áreas, cf. Arqueologia, Museus, Documentação, Administrativo, Gestão, etc.

- **Como modelar um Domínio numa perspectiva OO ?**

1) **Organizando e definindo todo o vocabulário "fundamental" do domínio do problema, em especial o que sobressai da análise de requisitos com os interlocutores numa tabela semântica, ou seja, um dicionário terminológico aceite e assinado por ambas as partes;**

2) **Organizando e relacionando termos que estão definidos num glossário ou num dicionário de dados firmado, se a complexidade do problema o justificar, definindo as classes que representam as entidades d**o estado interno persistente e partilhado do sistema, como sendo as entidades (e eventos) do negócio, e suas respectivas ligações semânticas a outras entidades, bem como as classes **que modelam a estrutura de documentos (que são dados estruturados) trocados entre o sistema e o seu ambiente;**

# Modelos do Domínio

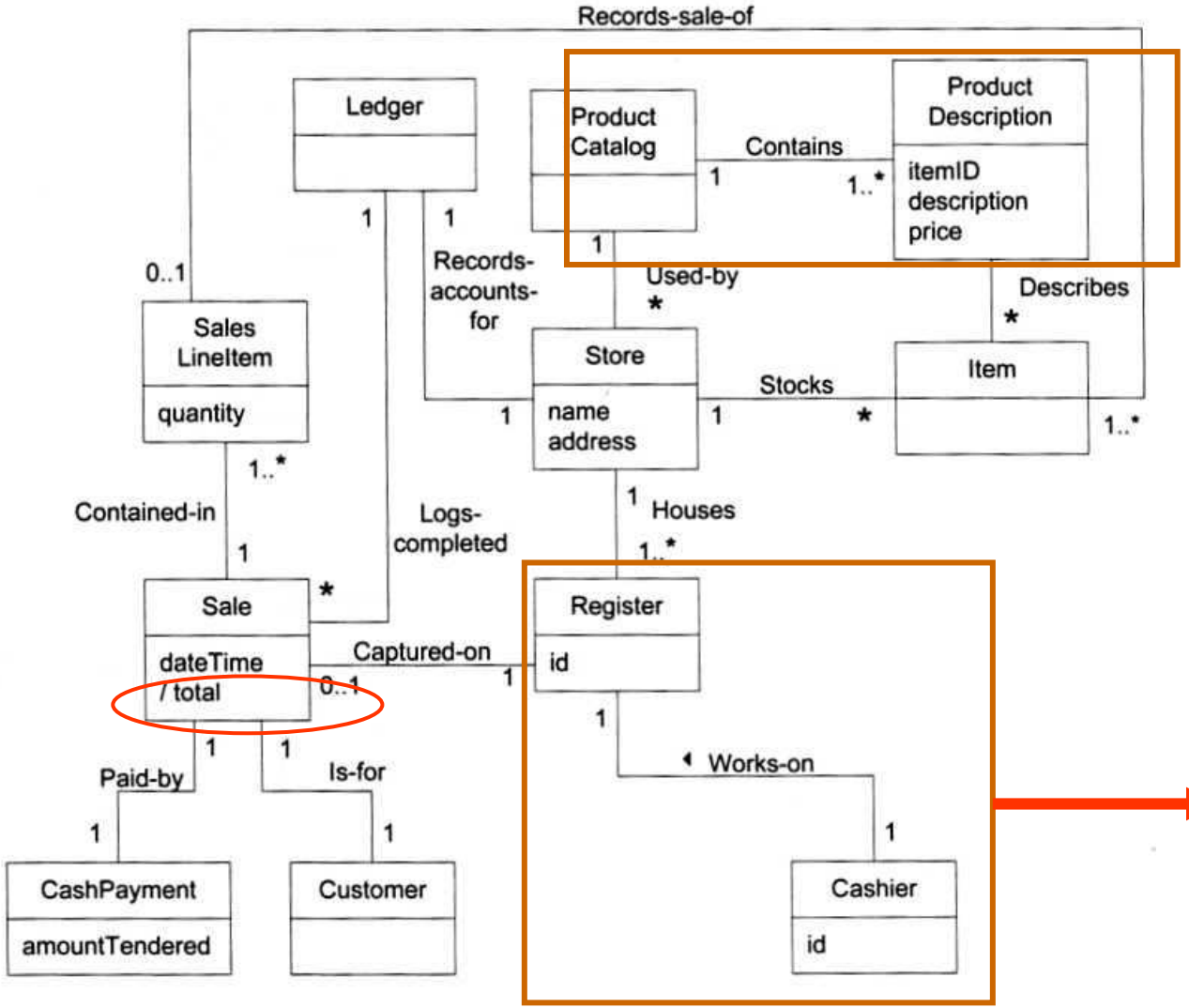



**Jogo de Dados**

- Classes, atributos e relacionamentos servem em UML para modelar o domínio do sistema, tal como, com mais detalhe, servem para modelar os elementos sofware.
- Porém, **Modelos de Domínio** são Conceptuais e **Diagramas de Classe** são já especificações/implementações com vista à codificação!

Atenção !

# Modelos do Domínio: POS




Associações entre classes podem possuir:

Nome

Contains, Paid-by, etc.

Multiplicidade:

1 – um, 0 - zero

* - mais de 1

1..* - 1 ou mais

Navegabilidade, ou seja, orientação:

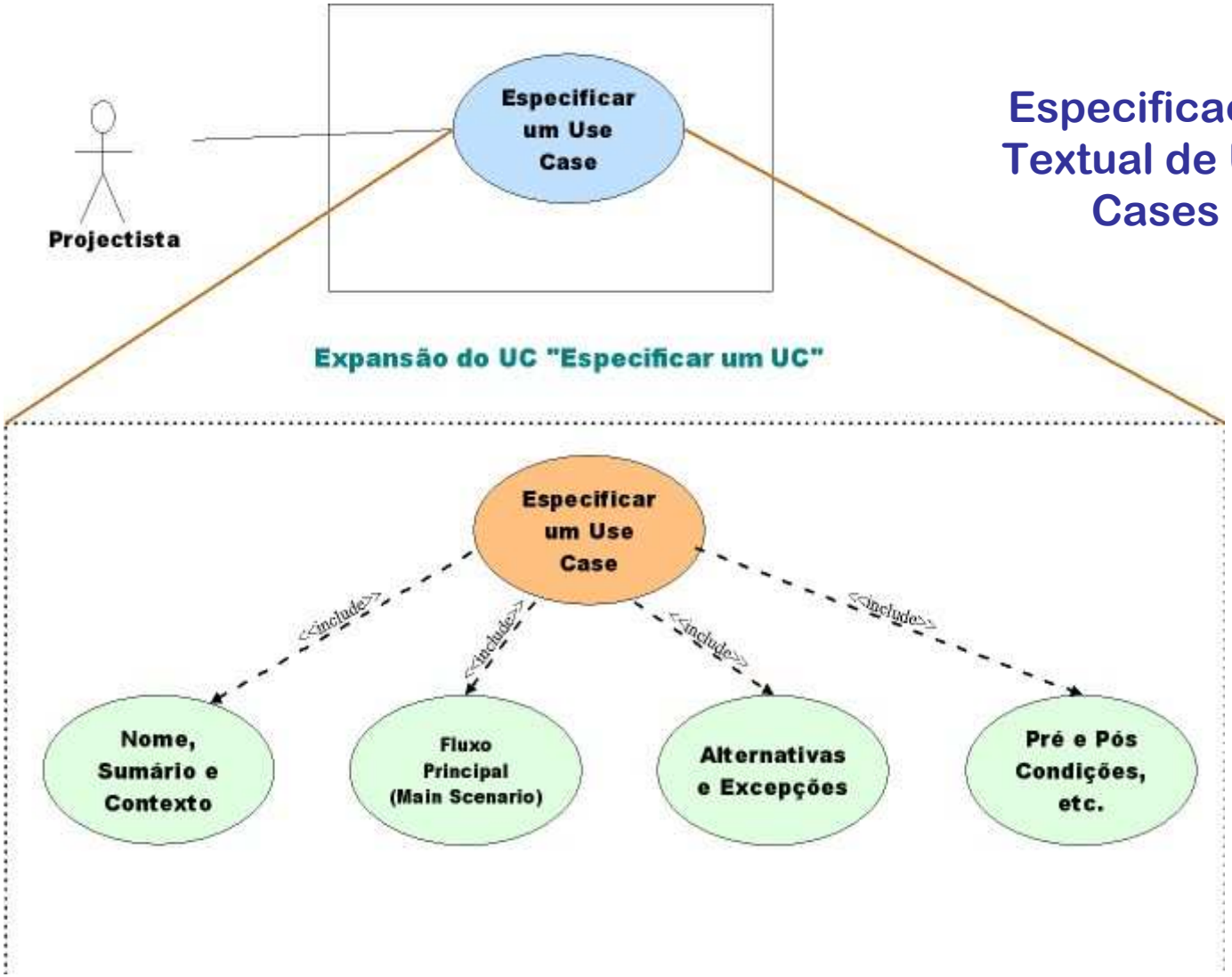
"Cada funcionário de caixa trabalha, a cada momento, numa máquina com identificação única".

**UMA ENORME, E POR ISSO INTERESSANTE REFLEXÃO SOBRE A DIFICULDADE EM ENCONTRAR DISPONÍVEIS, ONDE QUER QUE SEJA, NAS EMPRESAS DE SW, NA INTERNET, ETC., MODELOS DE DOMÍNIOS ANALISÁVEIS OU, PELO MENOS, DADOS COMO EXEMPLOS QUE POSSAM SER APRESENTADOS NUMA AULA DE ENGª INFORMÁTICA, É SINTOMÁTICA DE QUE A MAIORIA DAS ORGANIZAÇÕES NÃO TÊM NEM MODELOS DE DOMÍNIO NEM DE NEGÓCIO.**

**PORQUE SERÁ ??**

**PORTANTO, TEMOS QUE SER NÓS, ENGENHEIROS DE SW, A INFERI-LOS, ANALISÁ-LOS E DEPOIS, SE POSSÍVEL, A MELHORÁ-LOS E A IMPLEMENTÁ-LOS.**

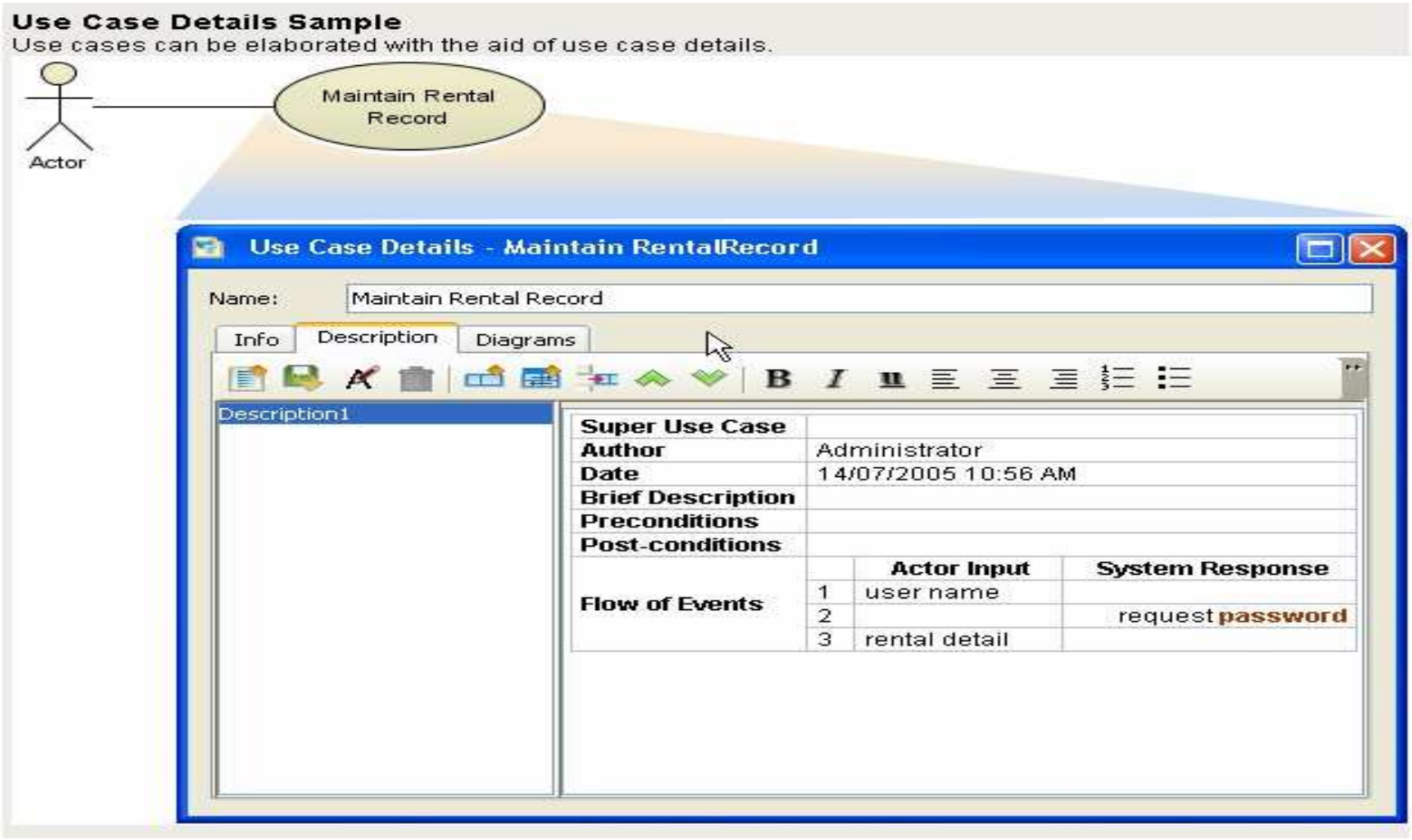
# Como especificar UC




## Especificação Textual de Use Cases

# Especificação Textual de UC

## O formato é livre, cf. UML 2.0.
## Nós usaremos o template do Visual Paradigm, mas com algumas modificações.



# A (single) Use Case: textual form

**Use Case: #1 Seller submits an offer**
**Scope:** Marketplace
**SuD:** Marketplace Information System
**Level:** Primary Task
**Primary Actor:** Seller
**Supporting Actor:** Trade Commission
**Main success scenario specification:**

1. Seller submits information describing an item
2. System validates the description.
3. Seller adjusts/enters price and enters contact and billing information.
4. System validates the seller's contact information.
5. System verifies the seller's history to permit the seller to operate
6. System validates the whole offer with the Trade Commission
7. System lists the offer in published offers.
8. System responds with an uniquely identified authorization number.

**Extensions:** Excepções

2a Item not valid
  2a1 Use case aborted
5a Seller's history inappropriate
  5a1 Use case aborted
6a Trade commission rejects the offer
  6a1 Use case aborted

**Sub-variations:** Alternatives

2b Price assessment available
  2b1 System provides the seller with a price assessment.

Vladimír Mencl
From Textual Use Cases to Behavior Specifications, Berlin, Apr 19, 2004

**Apenas 1 exemplo ...**

# Especificação Textual de UC

| | | |
|---|---|---|
| **USE CASE #** | | *Identificador: Frase contendo o verbo* |
| Objectivo no contexto | | *Objectivo do use case por palavras* |
| Âmbito e Nível | | *Que sistema é tomado como "black box"?* |
| | <nível> | <Geral; Tarefa; Subfunção> |
| **PRÉ-CONDIÇÃO** | | *Estado do sistema antes do início do use case* |
| ACTORES | <prim> | *Nome do papel do actor primário (o que inicia o UC)* |
| | <sec> | *Outros sistemas envolvidos* |
| | | |
| **NORMAL (Sucesso)** | **PASSO** | **ACÇÃO** |
| | 1 | |
| | 2 | *passos do cenário desde o início até se atingir o objectivo,* |
| | 3 | *podendo conter <<include>> de use cases, pontos de* |
| | 4 | extensão e |
| | 5 | |
| | .... | |
| **EXCEPÇÕES** | **PASSO** | **ACÇÃO DE EXCEPÇÃO** |
| | x.a | *Condição causadora da ramificação* |
| | | *Acção ou nome do sub-usecase* |
| | | |
| | | |
| **ALTERNATIVAS** | **PASSO** | **ACÇÃO** |
| | x.a | *Condição causadora da ramificação* |
| | | *Acção ou nome do sub-usecase* |
| | | |
| **PÓS-CONDIÇÃO** | pc-suc | *Estado do sistema após execução com sucesso do use case* |
| | pc-erro | *Pode conduzir a estados de erro ? Quais ?* |
| Use Cases Superiores | | |
| Sub-use cases | | |
| | | |
| **OUTRA INFORMAÇÃO** | | |
| Frequência (dia/mês) | | |
| Canais dos Actores | | *Telefone, e-mail, carta, ficheiro, ..* |
| Data Limite de Conclusão | | |
| Data Actual | | |
| Versão actual | | |
| Autores | | |

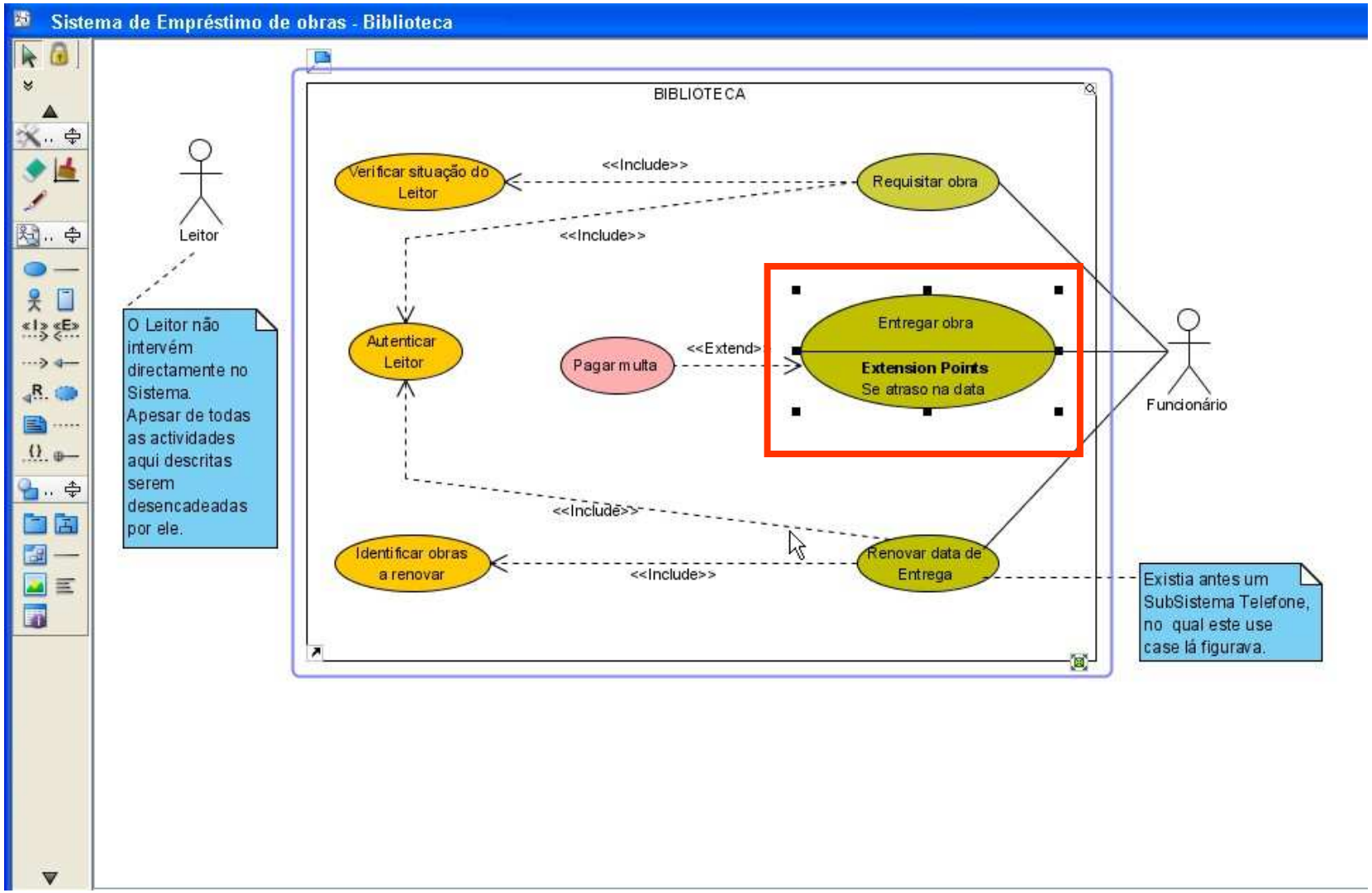
**O NOSSO MODELO**

**Especificação do Fluxo Principal**

**Especificação de fluxos de mau comportamento ou erro, eventualmente recuperáveis.**

**Especificação de Fluxos de alternativos de sucesso.**

# Especificação Textual de UC



# Especificação Textual de UC

**Entregar obra Details**

Name: Entregar obra

Info | Description | Diagrams

Main

| | | | |
|---|---|---|---|
| **Super Use Case** | | | |
| **Author** | Pedro | | |
| **Date** | 27/Dez/2007 21:08:33 | | |
| **Actors** | Funcionário | | |
| **Brief Description** | O Leitor pretende entregar obra requisitada | | |
| **Preconditions** | | | |
| **Post-conditions** | **Sucesso:** O Sistema actualizou registo da obra e ficha do Leitor.<br>**Insucesso:** O Sistema NÃO actualizou registo da obra e ficha do Leitor. | | |
| **Normal Flow of Events** | | **Actor Input** | **System Response** |
| | 1 | Funcionário insere código da obra | |
| | 2 | | O Sistema valida código da obra. |
| | 3 | | O Sistema bloqueia segurança da obra. |
| | 4 | | O sistema verifica data de entrega. |
| | 5 | | O Sistema actualiza Ficha do Leitor. |
| | 6 | | O Sistema actualiza Registo da Obra. |
| | 7 | | O Sistema arquiva requisição. |
| | 8 | | O Sistema emite talão comprovativo da entrega. |
| **Alternative Flow of Events** | | **Actor Input** | **System Response** |
| 2a - O código da obra é inválido. | 1 | Voltamos ao **passo 1**. | |
| **Alternative Flow of Events** | | **Actor Input** | **System Response** |
| 4a - data de entrega ultrapassada. | 1 | **extends:** Pagar Multa. | |
| | 2 | Voltamos ao **passo 5**. | |

# Especificação Textual de UC

## Próxima aula:

- Normalização da especificação textual de Use Cases;
- O que são Diagramas de Sequência em UML;
- Passagem sistemática dos Use Cases para os Diagramas de Sequência.