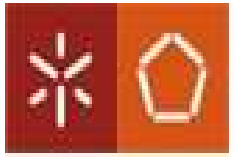
Desenvolvimento de Sistemas Software

Aula Teórica 5: Diagramas de Use Case / Especificação de Use Case

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



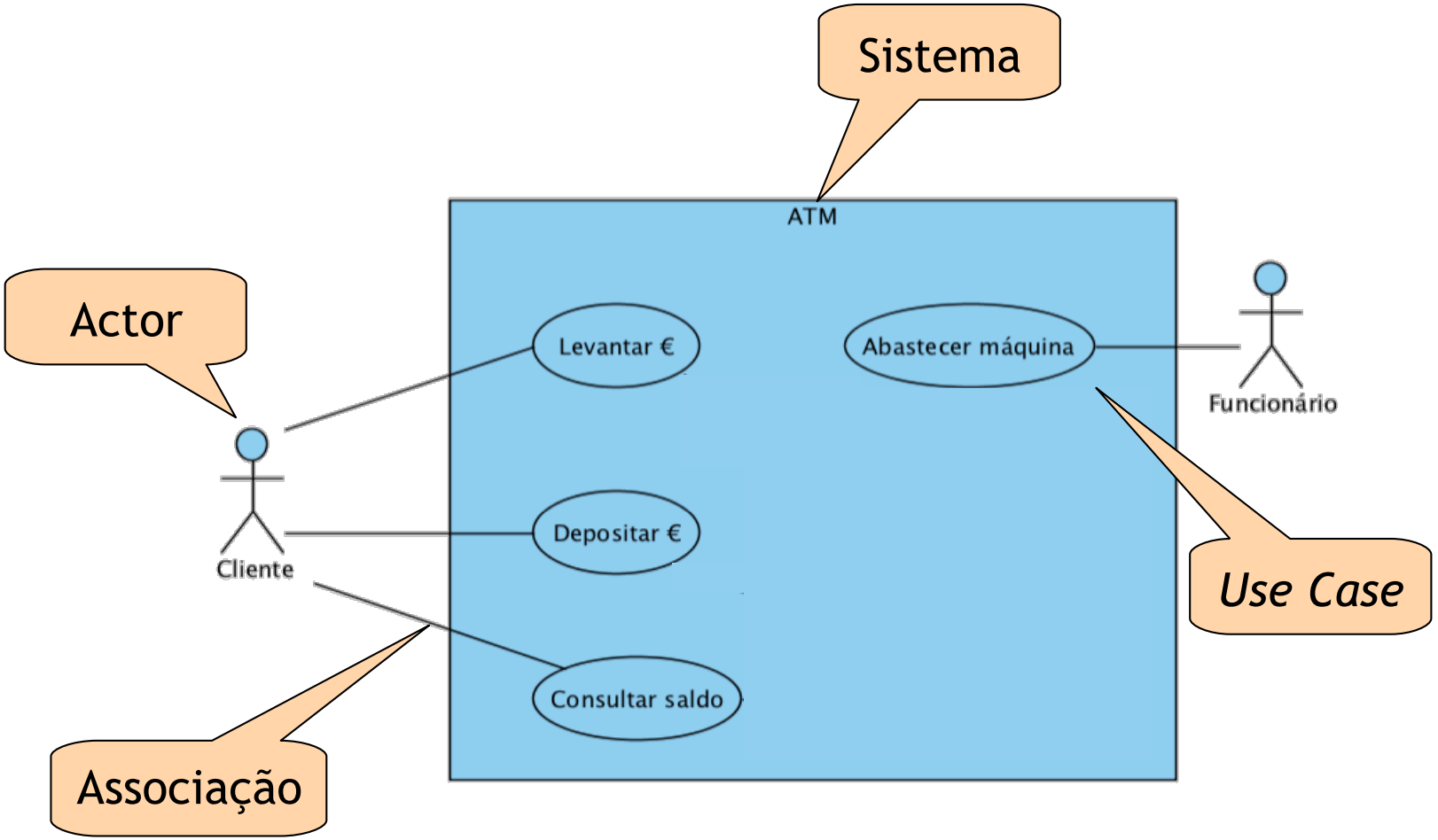
USE CASES: Importância

- 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.

To my knowledge, no other software engineering language construct as significant as use cases has been adopted so quickly and so widely among practitioners. I believe this is because use cases play a role in so many different aspects of software engineering.

Ivar Jacobson
Founding Father of UML and
Creator of Use Case Diagram

Diagrama de *Use Cases* - notação



Especificação de *Use Cases*

Use Case: Levantar €
Descrição: Cliente levanta quantia da máquina
Pré-condição: Sistema tem notas
Pós-condição: Cliente tem quantia desejada e saldo da conta foi actualizado
Comportamento normal:

1. Cliente apresenta cartão e PIN
2. Sistema valida acesso e apresenta opções
3. Cliente indica que pretende levantar dada quantia
4. Sistema processa levantamento da quantia
5. Sistema fornece quantia, talão e devolve cartão
6. Cliente retira notas, talão e cartão

Tipos de fluxos de eventos

- Em cada especificação de um Use Case podem/devem existir diferentes fluxos de controlo (sequências de eventos, comportamentos)
- Podemos caracterizá-los em três tipos:
  - Comportamento Normal (ou Fluxo Principal)

    O fluxo mais comum. Representa uma situação perfeita em que nada corre mal.
    A pós-condição é satisfeita.

  - Comportamentos/Fluxos Alternativos

    Fluxos válidos mas menos comuns.
    A pós-condição é satisfeita.

  - Comportamentos/Fluxos de Excepção

    Condições de erro suficientemente importantes para serem capturadas no modelo.
    A pós-condição NÃO é satisfeita.

Especificação de *Use Cases* - alternativas

Use Case: Levantar €
Descrição: Cliente levanta quantia da máquina
Pré-condição: Sistema tem notas
Pós-condição: Cliente tem quantia desejada e saldo da conta foi actualizado
Comportamento normal:

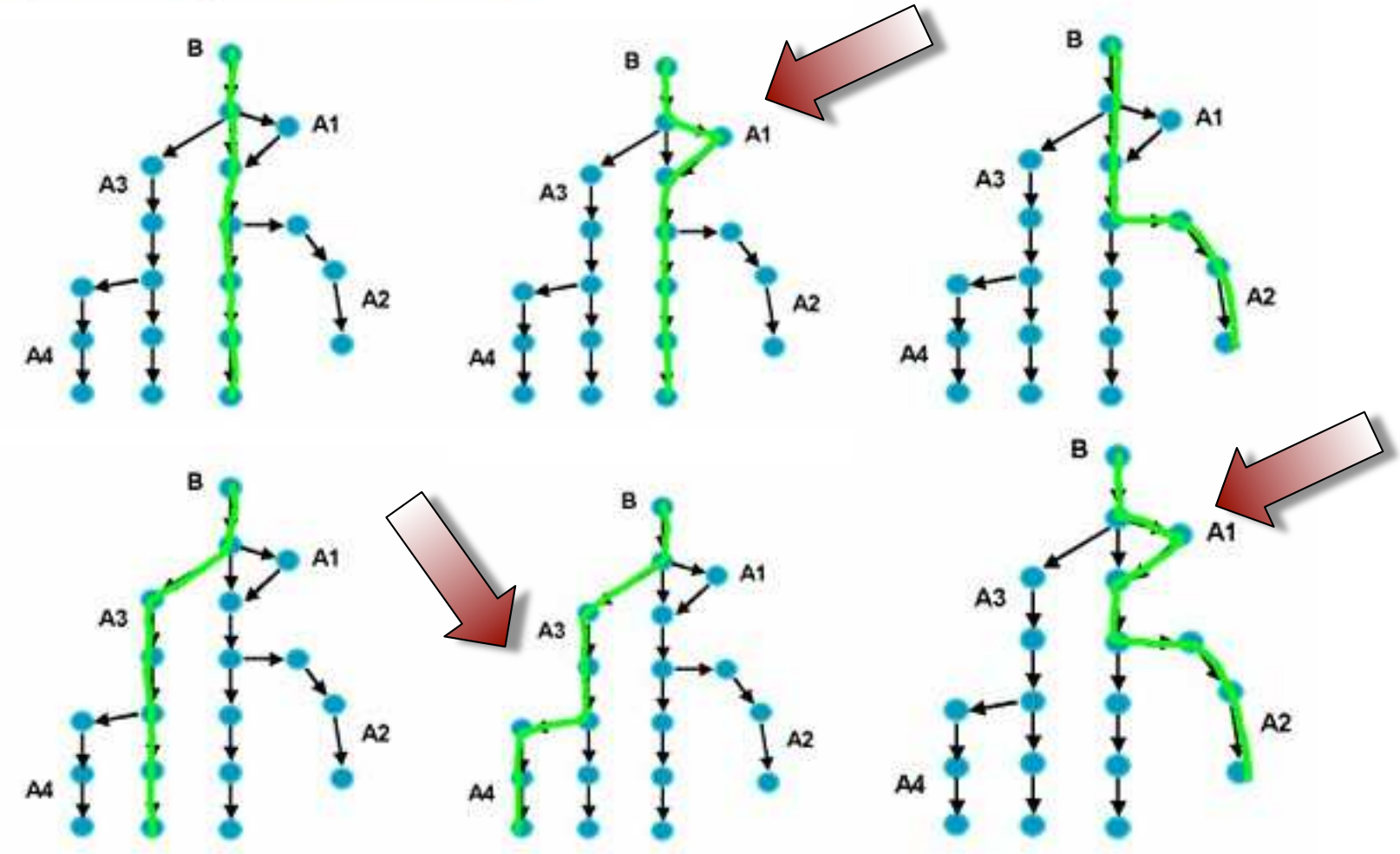
1. Cliente apresenta cartão e PIN
2. Sistema valida acesso e apresenta opções
3. Cliente indica que pretende levantar dada quantia
4. Sistema processa levantamento da quantia
5. Sistema fornece quantia, talão e devolve cartão
6. Cliente retira notas, talão e cartão

Comportamento Alternativo [sem papel]:

4.1. Sistema avisa de impossibilidade de emitir talão e pergunta se deve continuar
4.2. Actor diz que sim
4.3. Sistema processa levantamento da quantia
4.4. Sistema fornece quantia e devolve cartão
4.5. Cliente retira notas e cartão

CENÁRIOS E SEU ESTUDO

Figure 6. Finding scenarios in a use case



Especificação de *Use Cases* - alternativas

Use Case: Levantar €
Descrição: Cliente levanta quantia da máquina
Pré-condição: Sistema tem notas
Pós-condição: Cliente tem quantia desejada e saldo da conta foi actualizado
Comportamento normal:

1. Cliente apresenta cartão e PIN
2. Sistema valida acesso e apresenta opções
3. Cliente indica que pretende levantar dada quantia
4. Sistema processa levantamento da quantia
5. Sistema fornece quantia, talão e devolve cartão
6. Cliente retira notas, talão e cartão

Comportamento Alternativo [sem saldo na conta]:

4.1. Sistema avisa sobre inexistência de saldo e pergunta se deve continuar a crédito
4.2. Cliente diz que sim
4.3. Sistema processa levantamento da quantia a crédito
4.4. Regressa a 5

Outras respostas possíveis em cenários alternativos/excepções

Tipos de fluxos de eventos

- Em cada especificação de um Use Case podem/devem existir diferentes fluxos de controlo (sequências de eventos, comportamentos)
- Podemos caracterizá-los em três tipos:
  - Comportamento Normal (ou Fluxo Principal)

    O fluxo mais comum. Representa uma situação perfeita em que nada corre mal.
    A pós-condição é satisfeita.

  - Comportamentos/Fluxos Alternativos

    Fluxos válidos mas menos comuns. Úteis para capturar diferentes opções dos Actores. Úteis para capturar erros recuperáveis.
    A pós-condição é satisfeita.

  - Comportamentos/Fluxos de Excepção

    Condições de erro irrecuperável suficientemente importantes para serem capturadas no modelo.
    A pós-condição NÃO é satisfeita.

Especificação de *Use Cases* - excepções

Use Case: Levantar €
Descrição: Cliente levanta quantia da máquina
Pré-condição: Sistema tem notas
Pós-condição: Cliente tem quantia desejada e saldo da conta foi actualizado
Comportamento normal:

1. Cliente apresenta cartão e PIN
2. Sistema valida acesso e apresenta opções
3. Cliente indica que pretende levantar dada quantia
4. Sistema processa levantamento da quantia
5. Sistema fornece quantia, talão e devolve cartão
6. Cliente retira notas, talão e cartão

Comportamento Alternativo [sem papel]:

4.1. Sistema avisa de impossibilidade de emitir talão e pergunta se deve continuar
4.2. Cliente diz que sim
4.3. Sistema processa levantamento da quantia
4.4. Sistema fornece quantia e devolve cartão
4.5. Cliente retira notas e cartão

Excepção [saldo de conta insuficiente]:

4.1. Sistema avisa de impossibilidade de levantar quantia
4.2. Sistema fornece cartão
4.3. Cliente retira cartão

O que falta?

Estruturas de Controlo

- if … then … else …

a) se existe papel então Sistema fornece talão senão pergunta se deve continuar e Cliente diz Sim
b) se existe papel então Sistema fornece talão
c) Sistema fornece quantia e devolve cartão
d) Cliente retira notas, se existe papel então retira talão e retira cartão

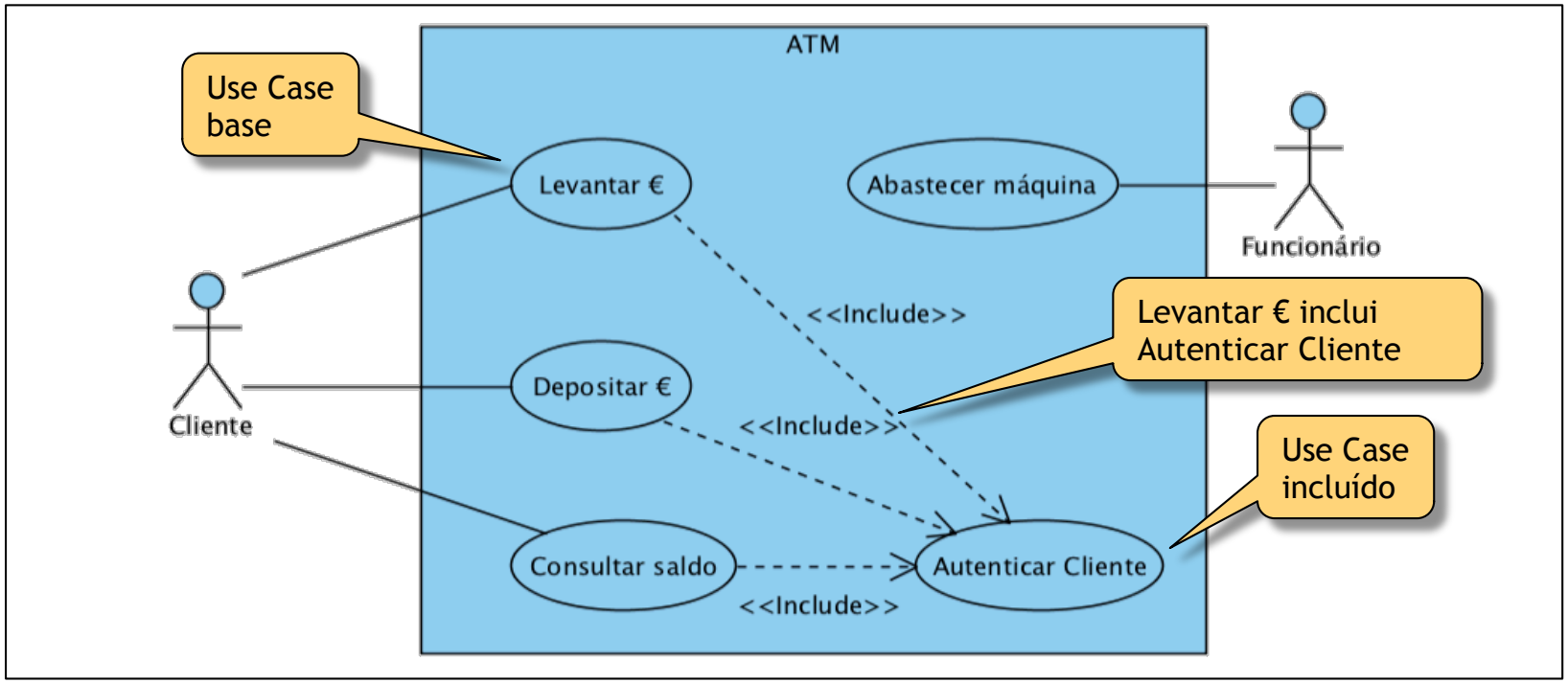
  - Vantagens
    - Estrutura de controlo familiar
    - Diminui número de fluxos alternativos
  - Inconvenientes
    - Use Cases mais difícieis de ler/compreender
    - Cenários alternativos mais difíceis de identificar
    - Use Cases mais difíceis de testar e de implementar
- go to …
  - permitem definir ciclos – evitar!

Mais sobre Diagramas de Use Case

- Dependências entre Use Cases
  - <<include>
  - <<extends>>
- Generalização

Diagramas de *Use Cases* - <<include>>

- Um estereótipo de dependência.
- Utilizado para indicar a reutilização de comportamento.



- Actores utilizam os *use case* base.
- Quando o *use case* base é executado, também o *use case* incluido o é

Use Case: Autenticar cliente

Comportamento normal:

1. Actor fornece cartão
2. Sistema pede PIN
3. Actor indica PIN
4. Sistema valida dados

Use Case: Levantar €

Comportamento normal:

1. «include» Autenticar Cliente
2. Sistema apresenta opções
3. Cliente indica que pretende levantar dada quantia
4. Sistema processa levantamento da quantia
5. Sistema fornece quantia, talão e devolve cartão
6. Cliente retira notas, talão e cartão

<<include>>

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

Vamos flexibilizar esta norma, tratando as excepções no UC base !!

Use Case: Levantar €

Comportamento normal:

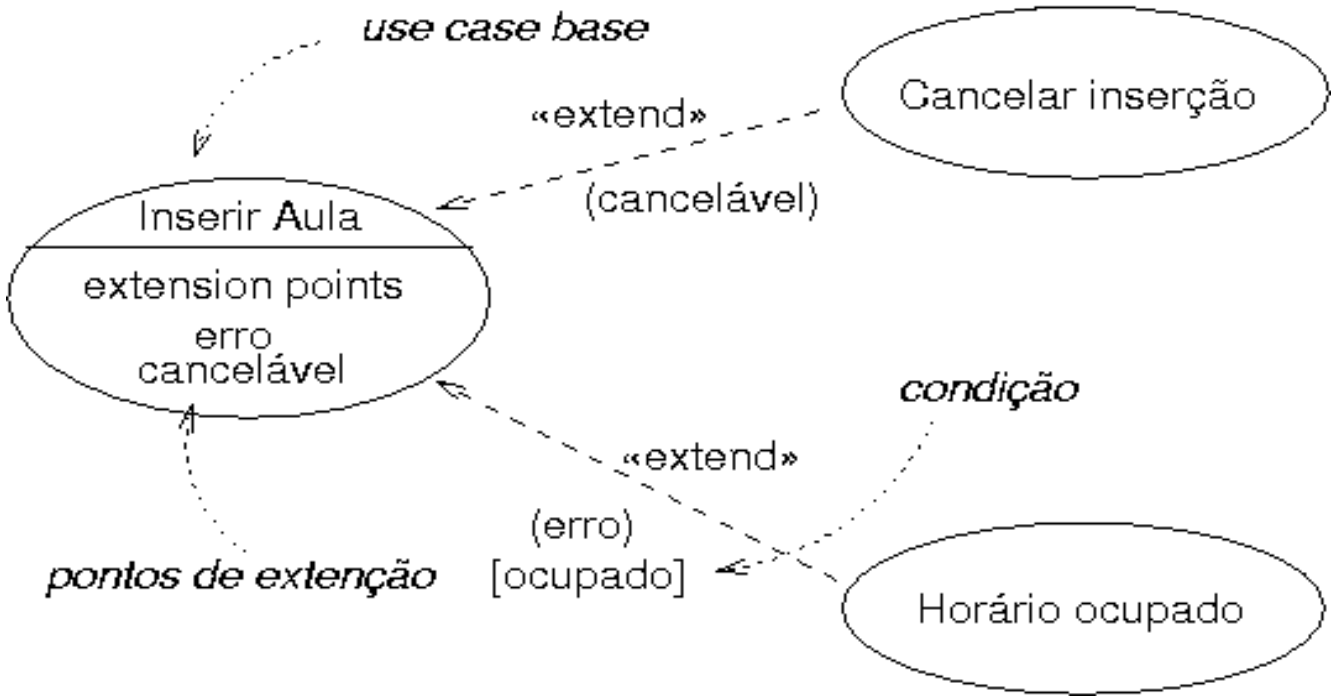
1. «include» Autenticar Cliente
2. Sistema apresenta opções
3. Cliente indica que pretende levantar dada quantia
4. Sistema processa levantamento da quantia
5. Sistema fornece quantia, talão e devolve cartão
6. Cliente retira notas, talão e cartão

Excepção [autenticação falha]

2.1. Sistema informa que autenticação falhou
2.2. Sistema devolve Cartão
2.2. Cliente recolhe Cartão

«*extend*»

- Outro estereótipo de dependência.
- Permite adicionar comportamento a um *use case* base.



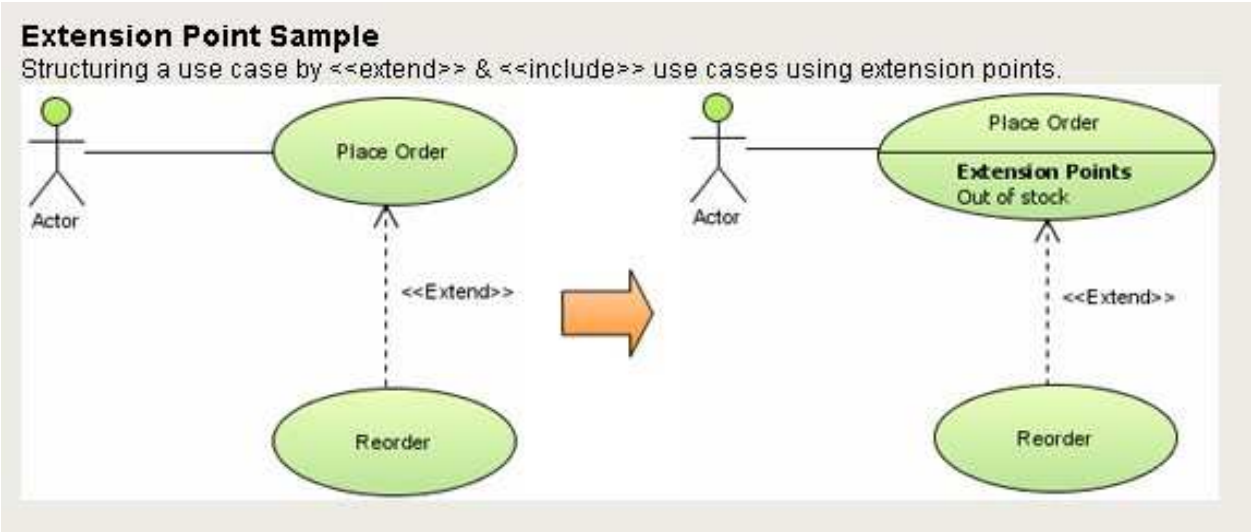
- Estratégia: escrever caso base; identificar variações; utilizar extensões para elas.
- Caso base deve ser um *use case* bem formado sem as extensões!
- Extensão pode não ser um *use case* bem formado por si só.

USE CASES: Relacionamentos

<<extend>>

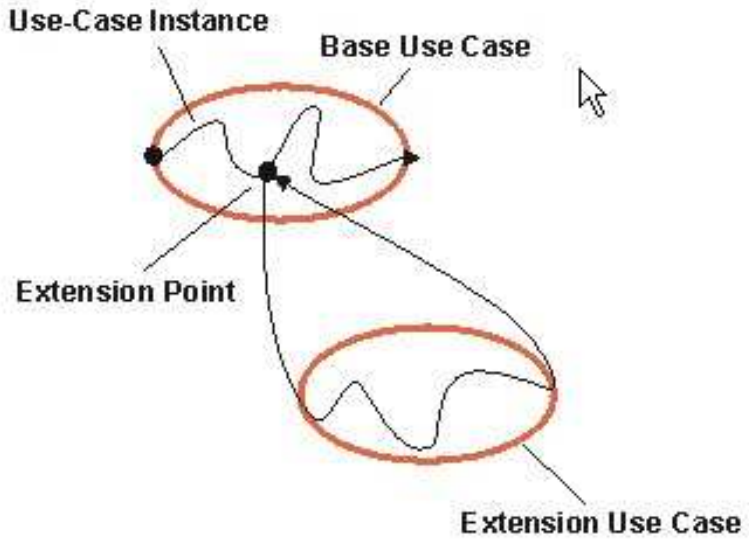
Os "extending" use cases servem para, em dadas condições, ou seja, condicionalmente, acrescentarem funcionalidade ao "use case" de base (em função das **condições de extensão**).

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: Extension Points

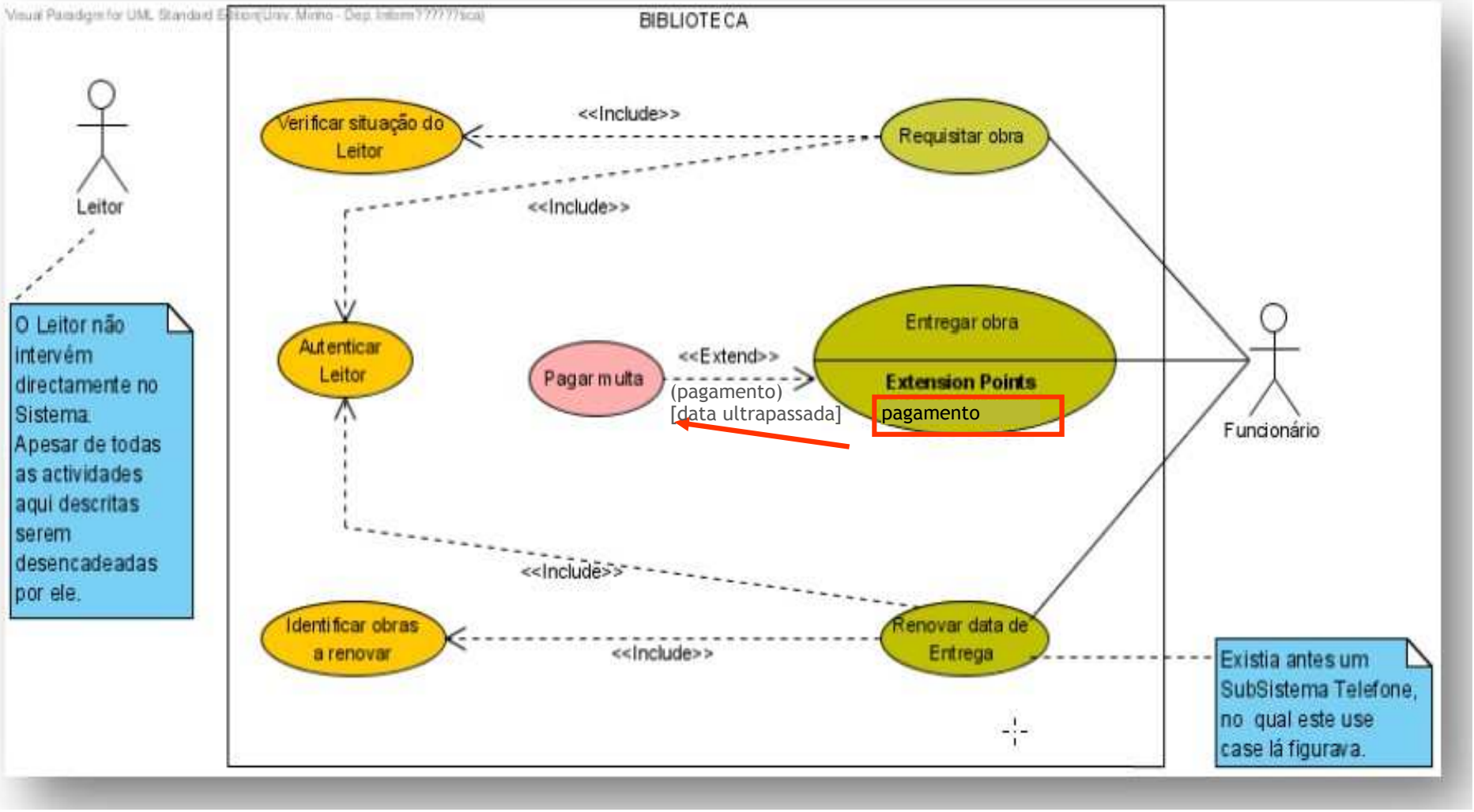
Semântica operacional correcta de <<extend>>



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.

Universidade do Minho
Departamento de Informática

USE CASES: <<extend>>



Use Case: Entregar Obra

Comportamento normal:

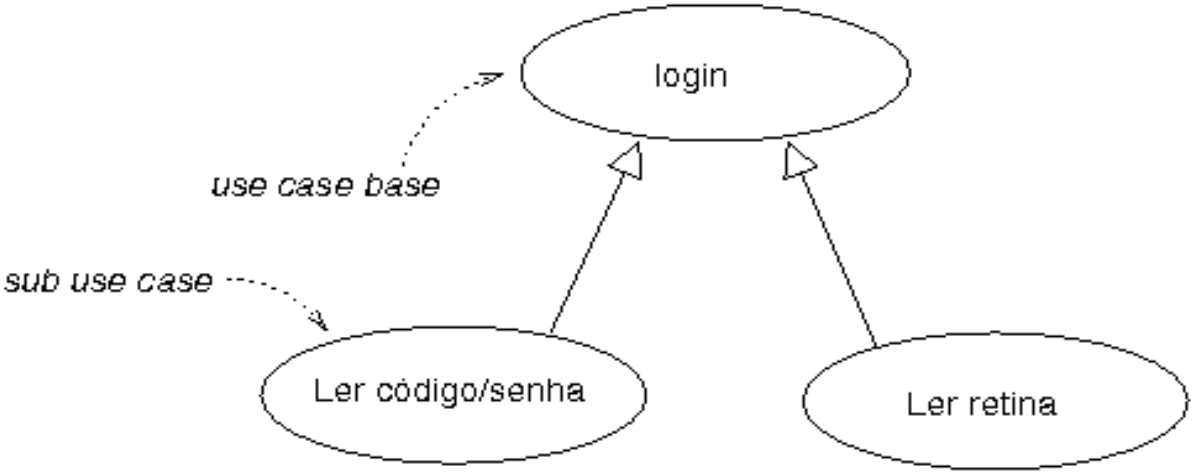
1. Actor apresenta obra
2. Sistema identifica obra
3. Sistema apresenta preço [extension point: pagamento]
4. Actor confirma realização de pagamento
5. Sistema imprime talão

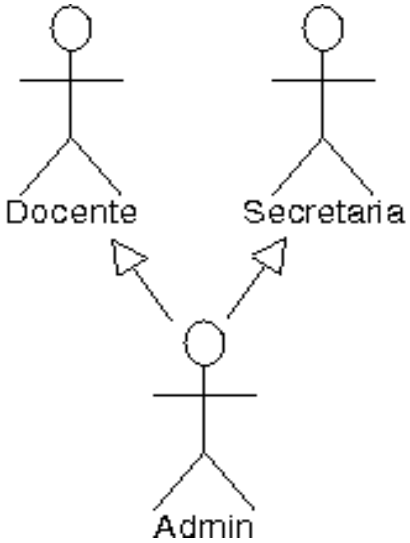
Extention point: pagamento

condição: data de entrega ultrapassada

3. **extended by**: Pagar Multa

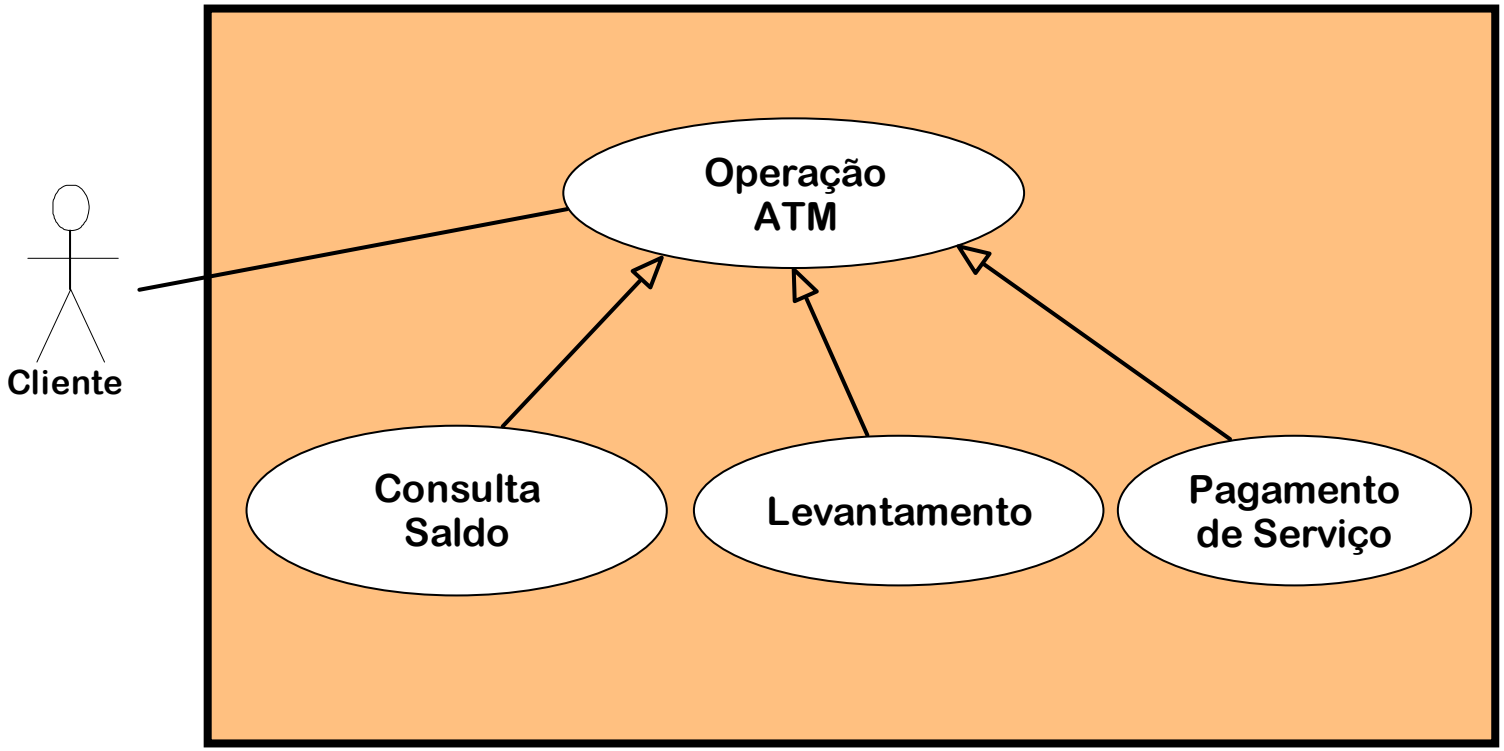
Generalização/Especialização





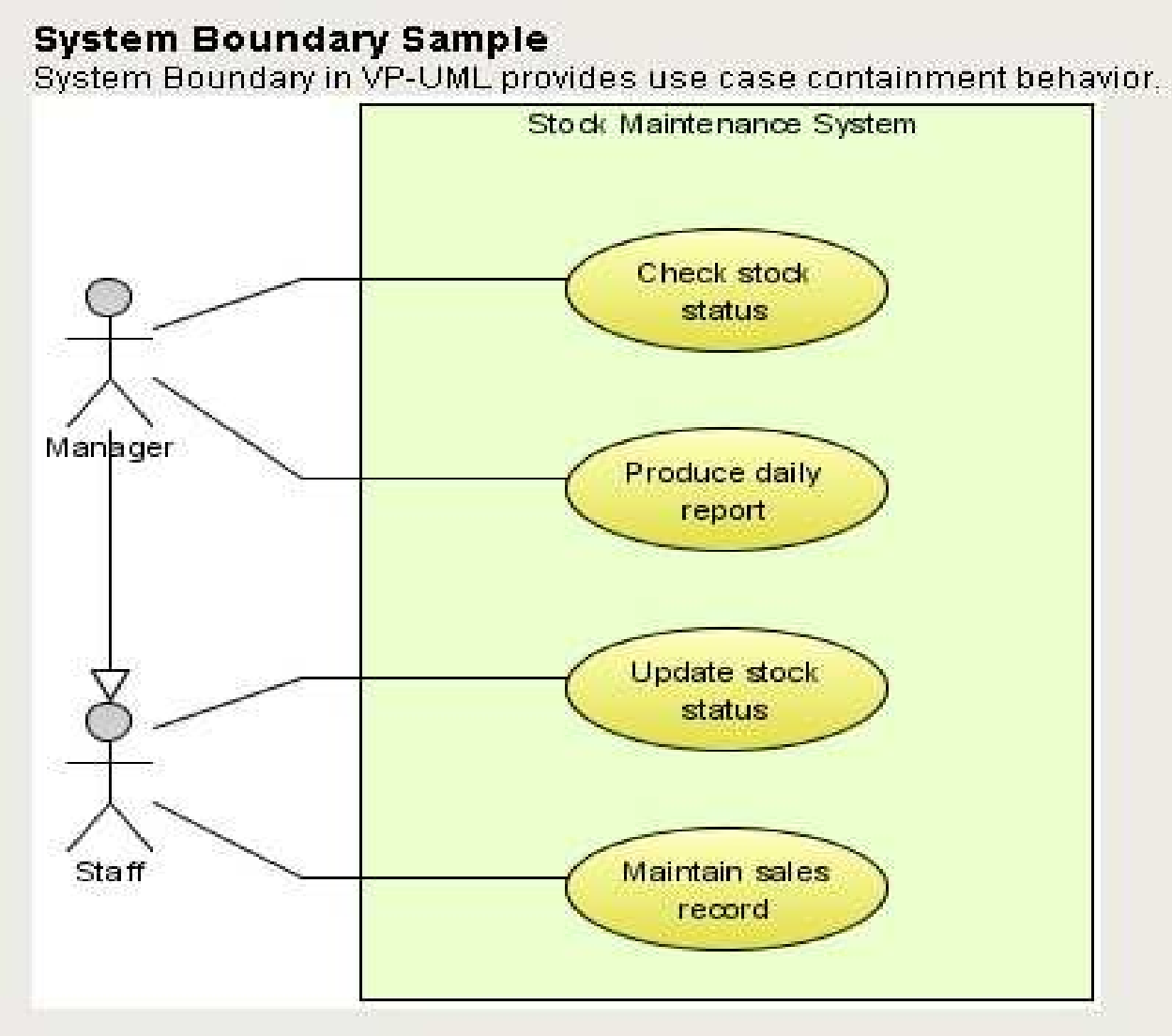
- Sub-elementos são casos particulares de super-elementos.
- Um sub-elemento pode ser utilizado onde quer que o super-elemento possa.
- Útil para user profiling (definição de níveis de acesso).
- Nos exemplos apresentados:
  - Existem duas formas de fazer login.
  - O actor Admin pode realizar todos os *use cases* de Docente e Secretaria.

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).

ACTORES: Generalização

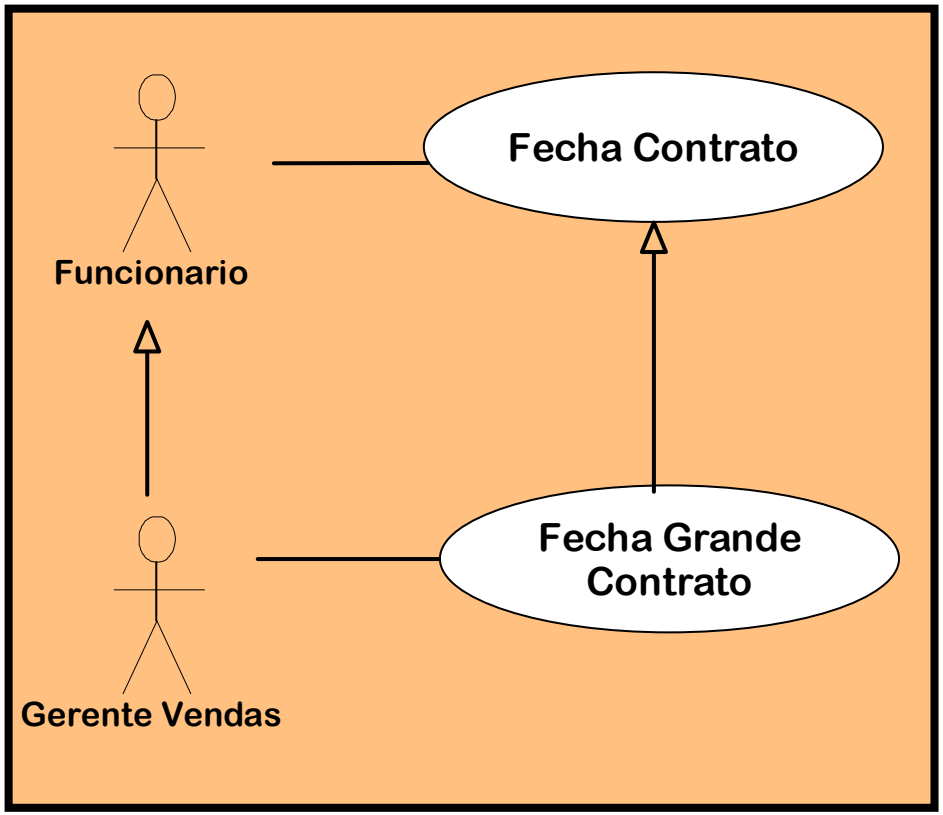


Generalização de Actores

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

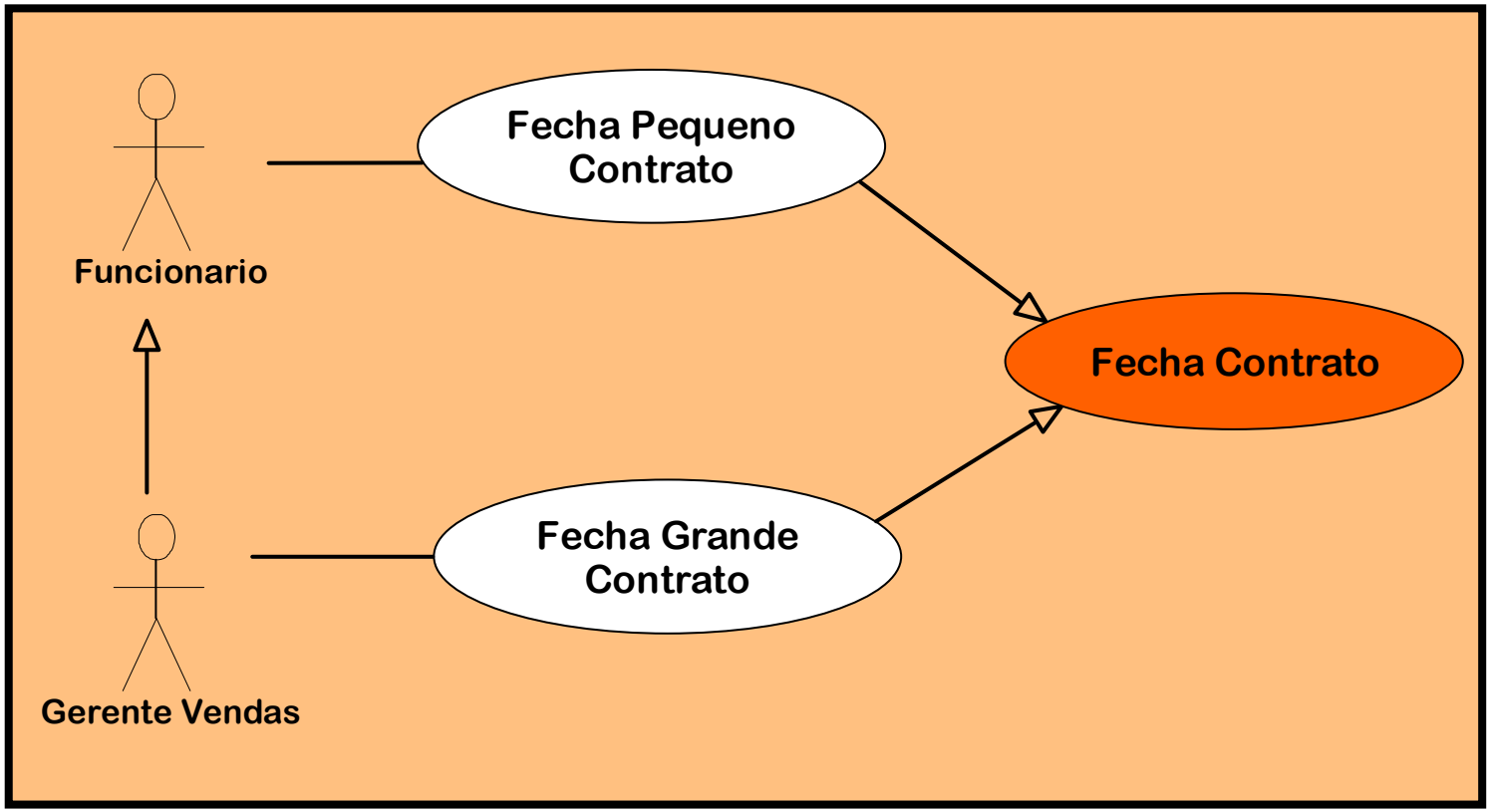
Problemas com a generalização de Actores e UCs



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

Não era certamente isto que se pretendia especificar !

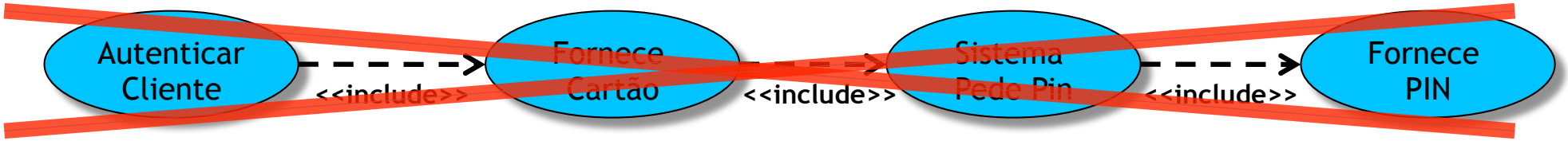
Solução (Generalização das tarefas com "profiling" adequado):



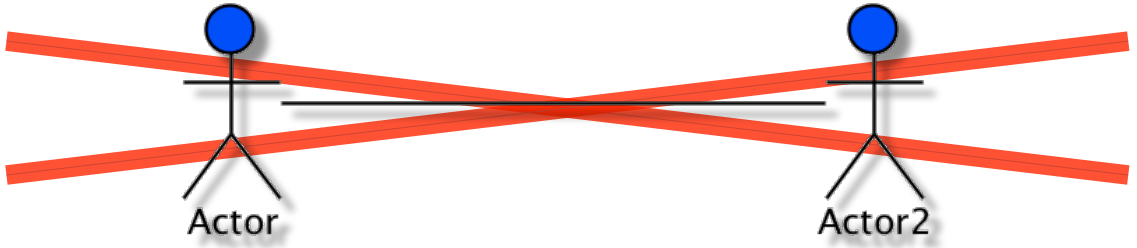
Situação pouco comum em que se justifica generalização de UC.

Outros aspectos a ter em atenção…

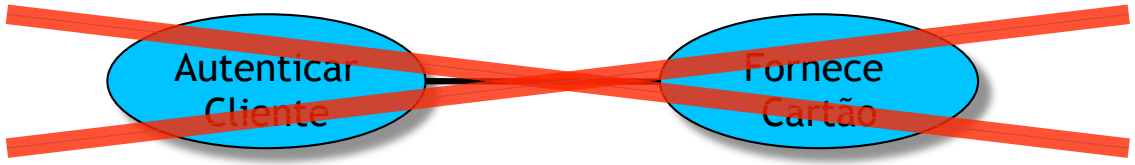
- Modelo de Use Cases não representa fluxo de dados



- Modelo de Use Cases não representa comunicação entre actores



- Modelo de Use Cases não representa comunicação entre use cases



- include, extends e generalização devem ser utilizados com moderação!

Hunting for use-case scenarios

Part I: Analyzing customer psychology

by Pan-Wei Ng

Independent Consultant

Use cases, an increasingly popular technique, are often the basis for successful projects. One reason the technique is so powerful is that use cases force both user representatives and analysts to explore system usage scenarios -- and reach consensus on how to handle them -- early in the project lifecycle. This is crucial to project success, as these same scenarios also drive the analysis, design, and test processes. However, hunting down every possible user or system behavior is not simple; it requires lateral thinking and venturing out of the box. Frequently, practitioners give up too easily and simply let use cases terminate.

(The Rational edge, Out 2003)

"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

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.

Modelação do Requisitos Funcionais

Sumário:

- Especificação de Use Cases
  - Fluxo principal; fluxos alternativos; excepções
- Estruturação dos modelos
  - Dependências entre Use cases; Generalização
- Notas finais