Universidade do Minho
Departamento de Informática

ARQUITECTURAS DE SOFTWARE

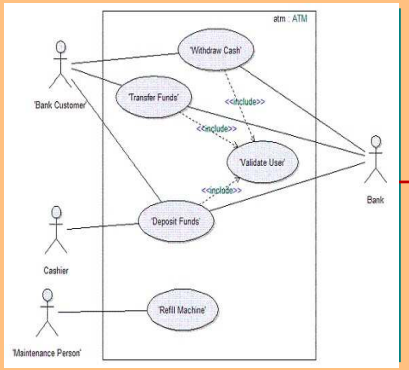
AULAS Nº 8 e 9

7-21/12/2007

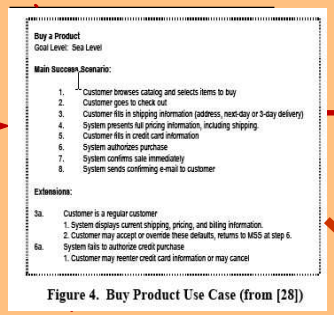
F. Mário Martins

Case Studies: Ligação das partes

Use Case Diagram



Use Case Specification



Buy a Product
Goal Level: Sea Level

Main Success Scenario:

1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping information (address, next-day or 3-day delivery)
4. System presents full pricing information, including shipping.
5. Customer fills in credit card information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming e-mail to customer

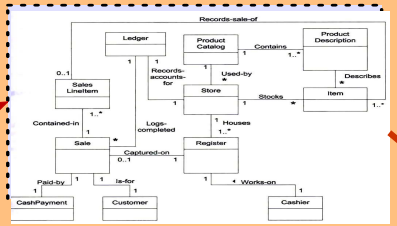
Extensions:

3a. Customer is a regular customer
1. System displays current shipping, pricing, and billing information.
2. Customer may accept or override these defaults, returns to MSS at step 6.

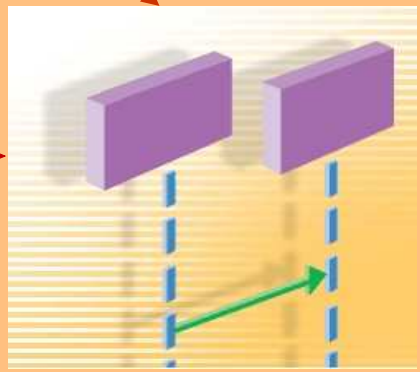
6a. System fails to authorize credit purchase
1. Customer may reenter credit card information or may cancel

Figure 4. Buy Product Use Case (from [28])

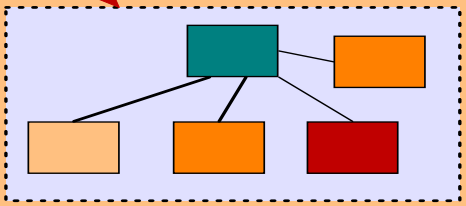
Domain Model

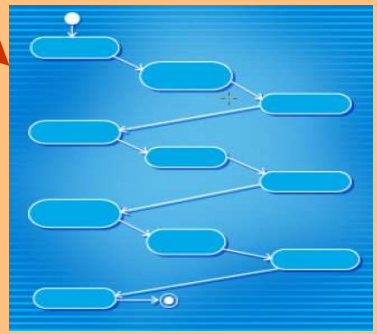


Dynamic Design Models



Static Design Model





A definição coerente de todos estes modelos corresponde à determinação das características funcionais e de estrutura do sistema

(só análise para já)

2 CASE STUDIES

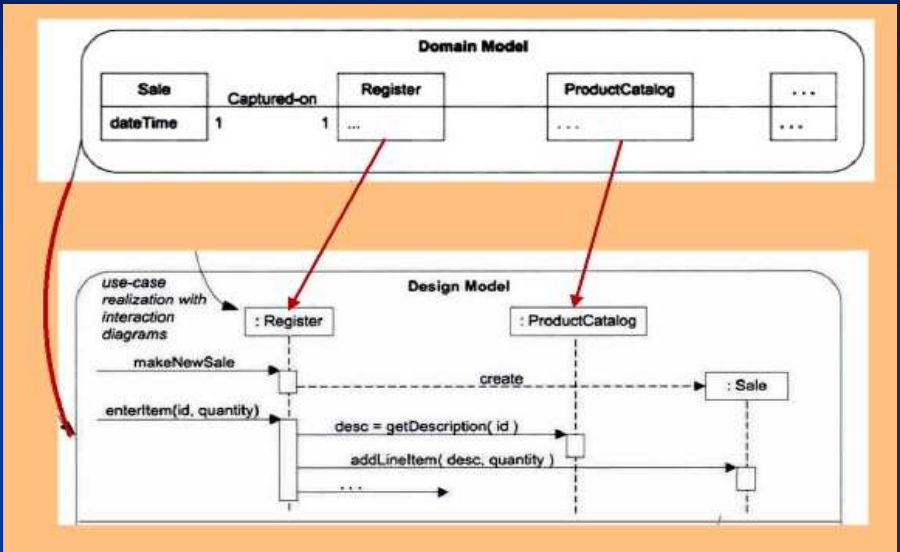
Máquina de Venda de X

- É um automatismo mecânico-electrónico. Tem características muito especiais. Implementa uma única tarefa (use case). Quando muito, e se tiver algum software, é um "embedded-system".

- É um bom exemplo para se compreender que a modelação e concepção de sistemas reactivos, electro-mecânicos, etc., baseados em sinais electrónicos ou mecânicos que se associam a comportamentos, bem como a especificação de temporizações, pode ser feita em UML2.0 mas exige um outro tipo de abordagem e metodologia de modelação.

- Estes sistemas são caracterizados pelos seus componentes (estrutura funcional) e como tais componentes cooperam entre si em resposta a dados eventos e sinais. Os seus modelos de domínio são em geral muito simples e o seu comportamento é, em geral, melhor especificado sob a forma de máquinas de estado.

Máquina de Venda de X



Domínios simples não implicam sempre modelos mais simples, mas implicam estruturas mais simples, ainda que possam por vezes possuir comportamento complexo

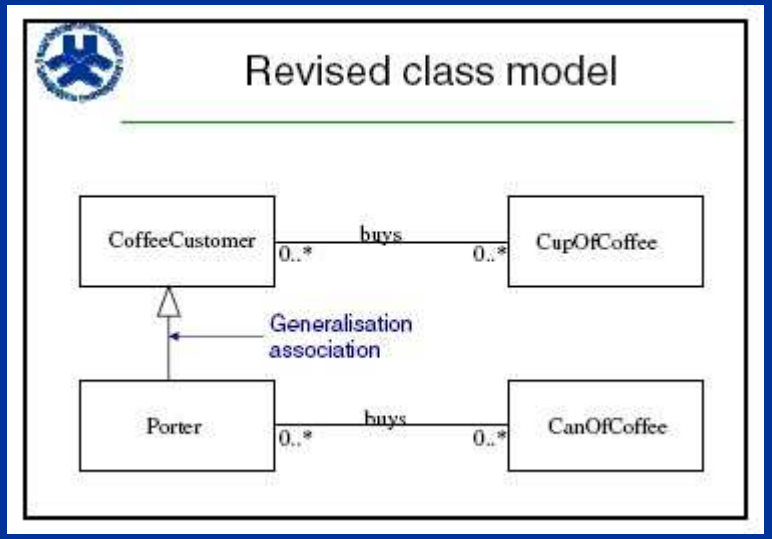
A complexidade global de um sistema de informação é resultante da complexidade estrutural (domínio e relacionamentos) da sua informação, que, por seu lado, só é complexa porque tem que servir de suporte a uma realidade operacional complexa (tarefas a realizar) inerente ao contexto da sua aplicação e funcionamento (goals).

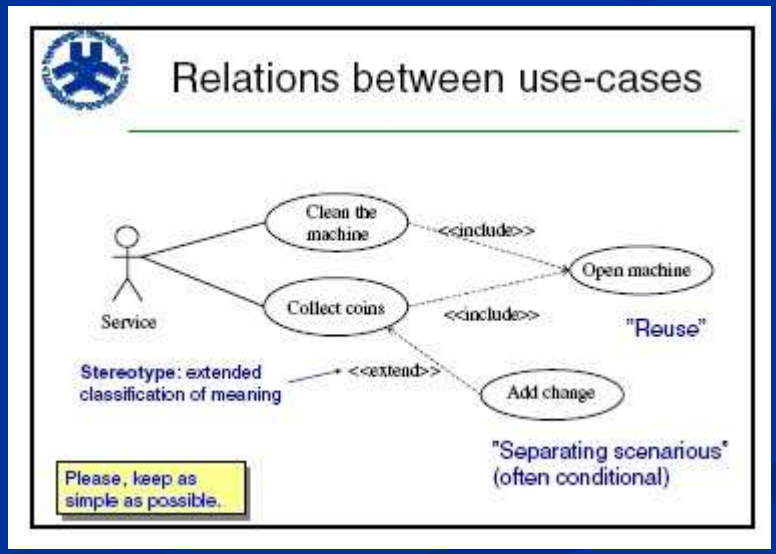
Nestes, para além das tarefas de negócio (mais-valias), temos que ter definidas todas as tarefas gestão da própria informação (!!). Distinguir ?

Máquina de Venda de X

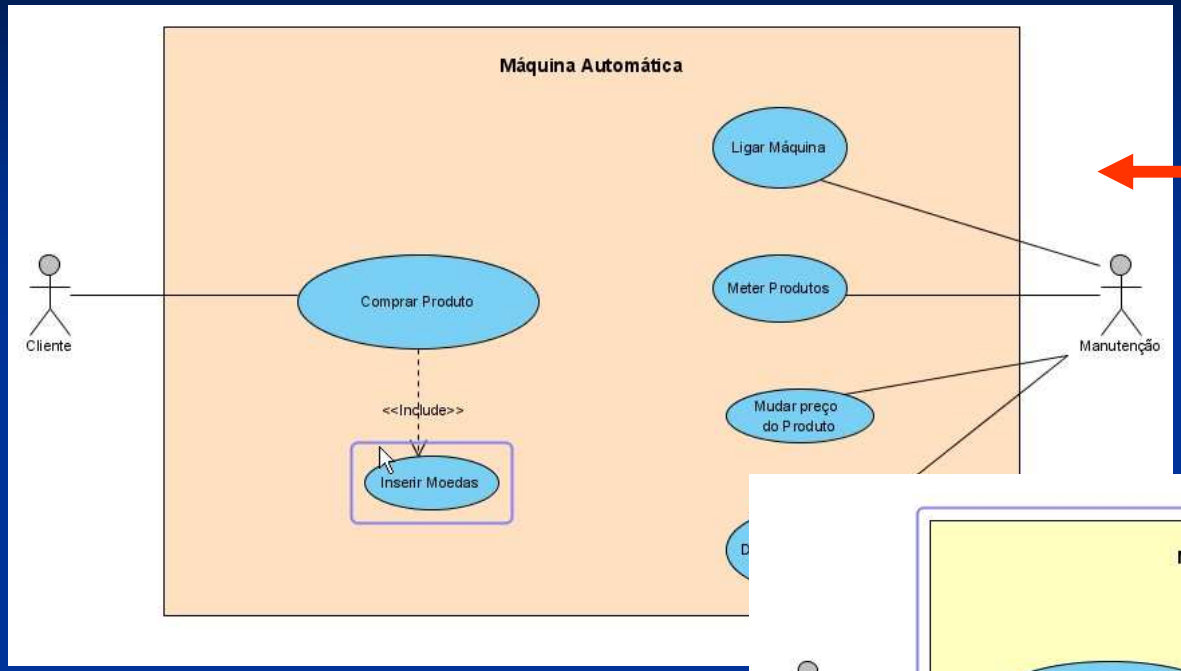
- Qualquer que seja a perspectiva e o nível de detalhe da abordagem, o modelo do domínio é muito simples.

- Tal como num Sistema de Informação sem grande utilidade, os use cases modelam as operações de "manutenção", não os verdadeiros "serviços".



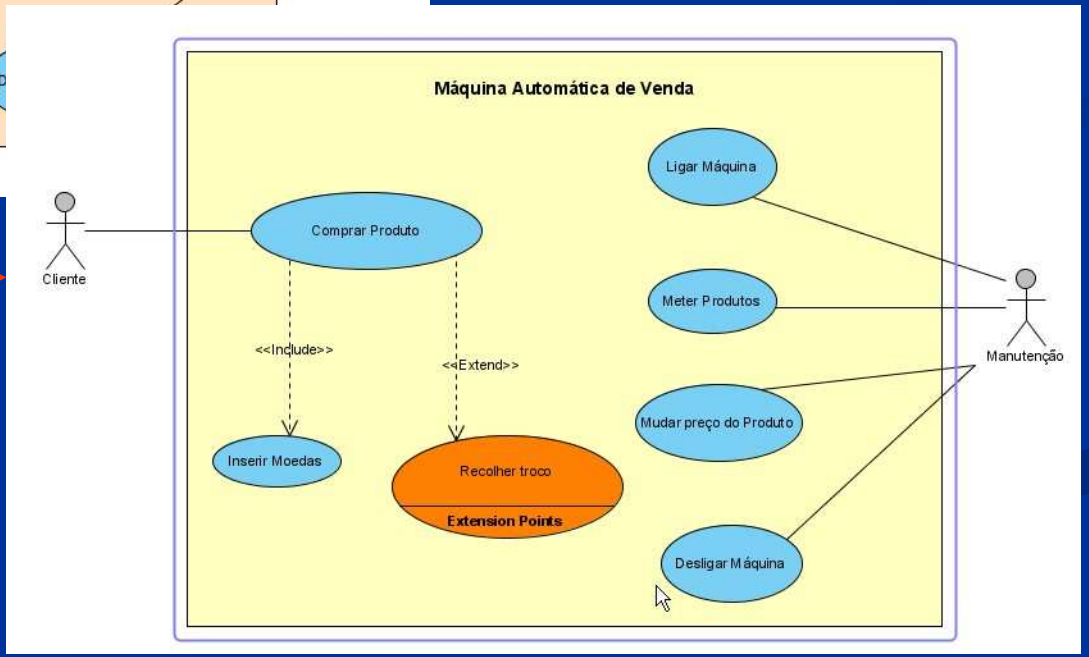


Máquina de Venda de X



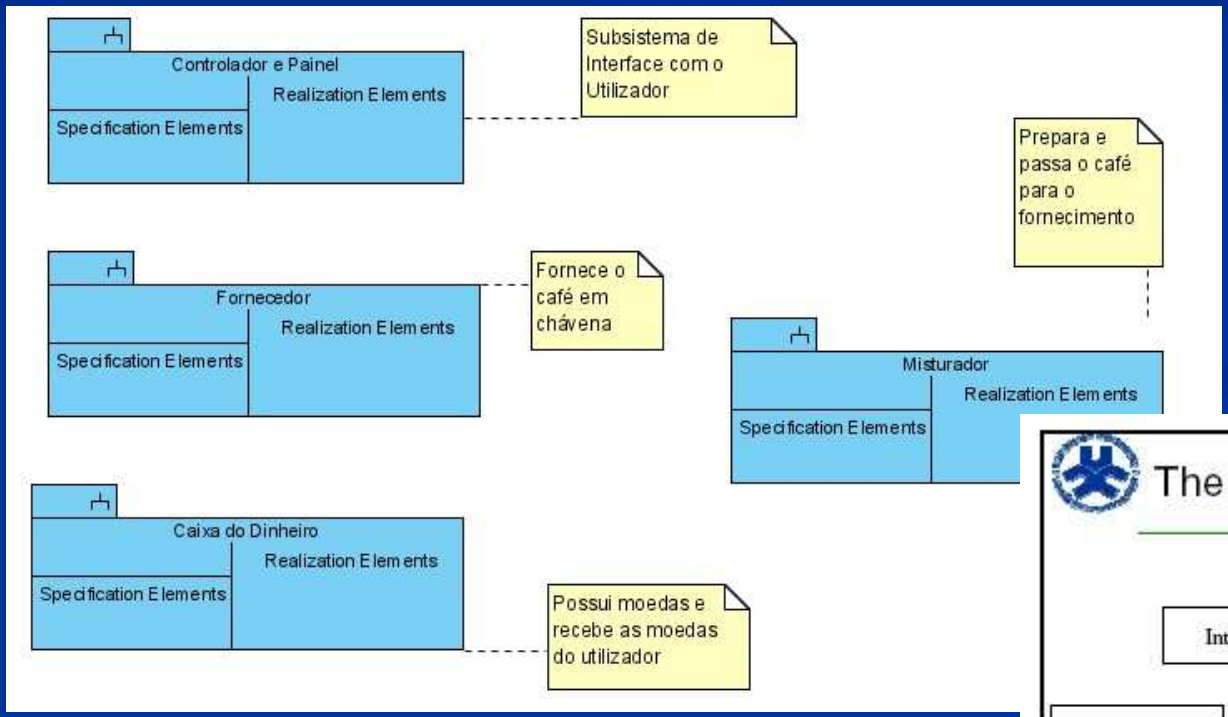
Alguns Use Cases de manutenção são, na realidade, sinais.

Use Case desdobrado em 2 sub-tarefas, uma das quais pode existir ou não, e é mais informativa do que realmente importante.

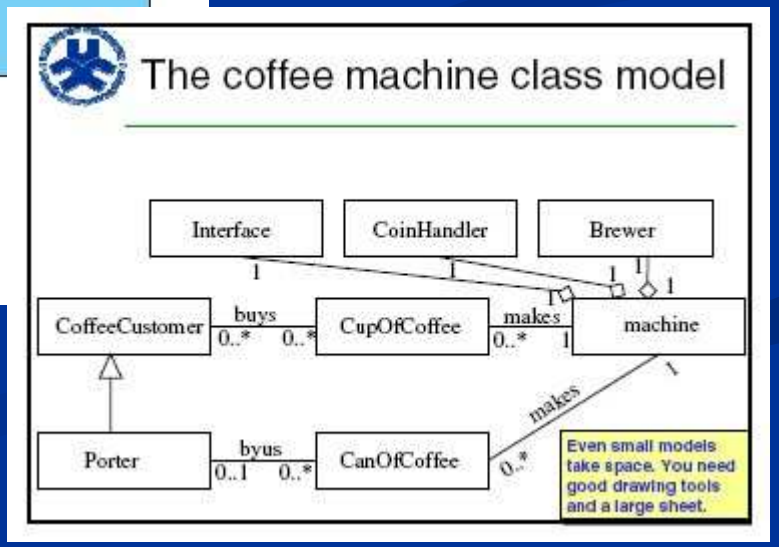


Máquina de Venda de X

▣ Assim, muito rapidamente se centra o interesse nos componentes funcionais, tal como fizemos com a máquina de café.



Usando, a alto nível, a notação de **subsistemas**,



ou mesmo usando Diagrama de Classes

MVx: Comportamento básico

Caixa (Moedas)	
Description: Dispositivo que guarda as moedas, contabiliza as moedas introduzidas pelo cliente e dispensa o troco se necessário	
Responsibilities:	
Name	Collaborator
Sabe o preço de um produto	Controlador
Sabe o total das moedas introduzidas	
Sabe se uma moeda é válida	
Rejeita moedas inválidas	
Devolve moedas introduzidas	Controlador
Devolve troco se for caso disso	
Sabe se pode dar troco	

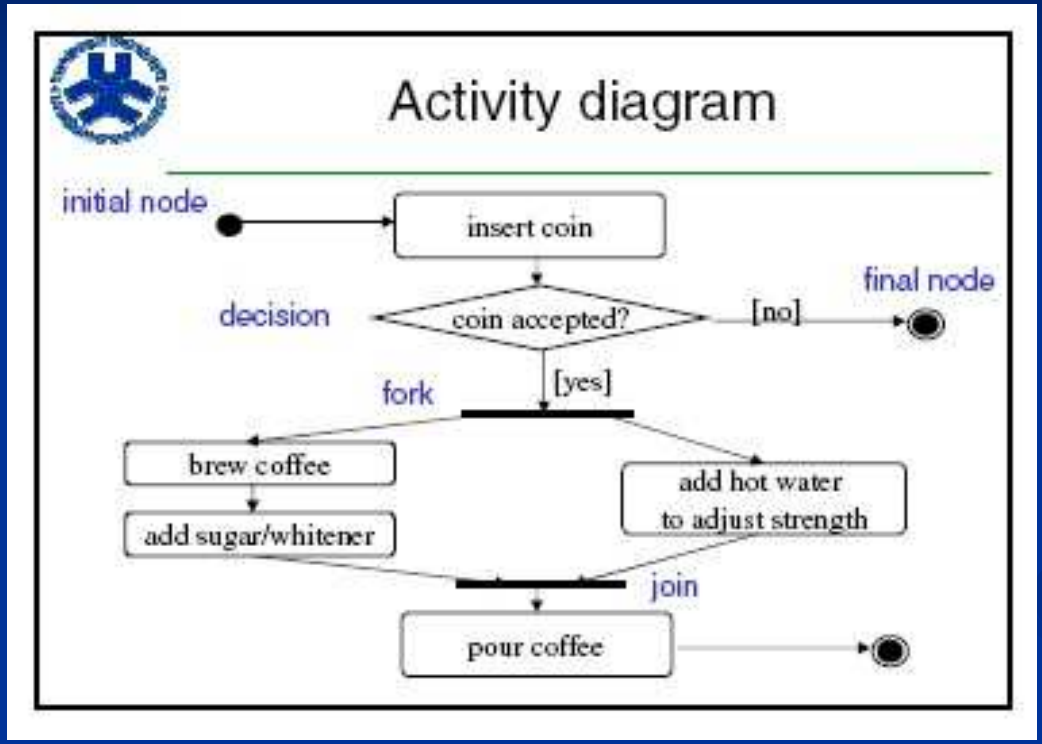
Definimos SRC-cards para cada subsistema, indicando para cada um:

a) O que deve saber;

b) O que deve fazer.

Os SRC-cards permitem depois analisar a coerência da sequência das actividades a realizar.

MVx: Comportamento básico



Diagramas de Actividade normais acabam por se revelar muito pobres na definição do real comportamento que garante a satisfação da tarefa fundamental da máquina !!

Um diagrama de actividades, para estes casos, pode descrever o essencial mas deve usar sinais e, eventualmente, temporizações.

MVx: Comportamento básico

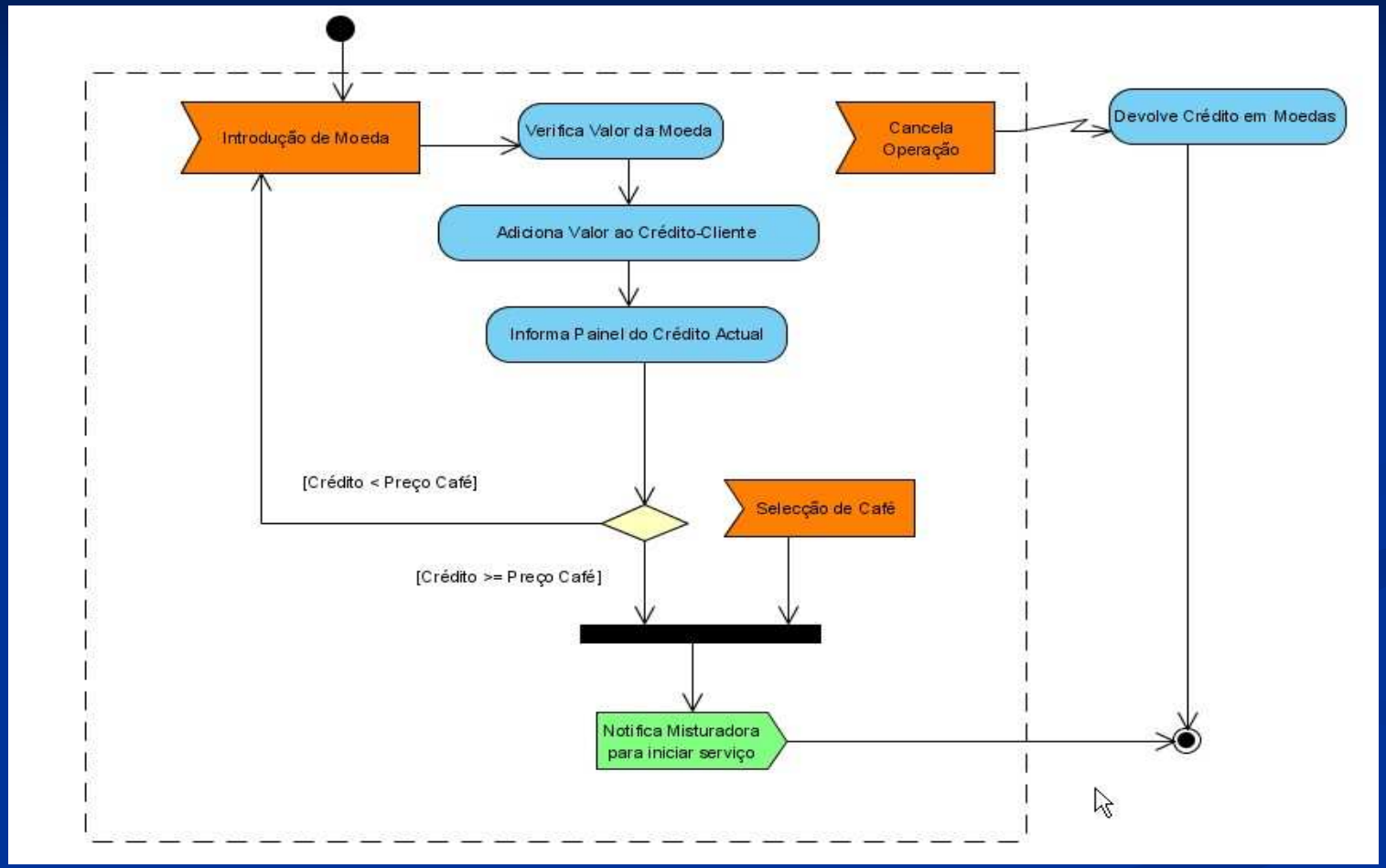
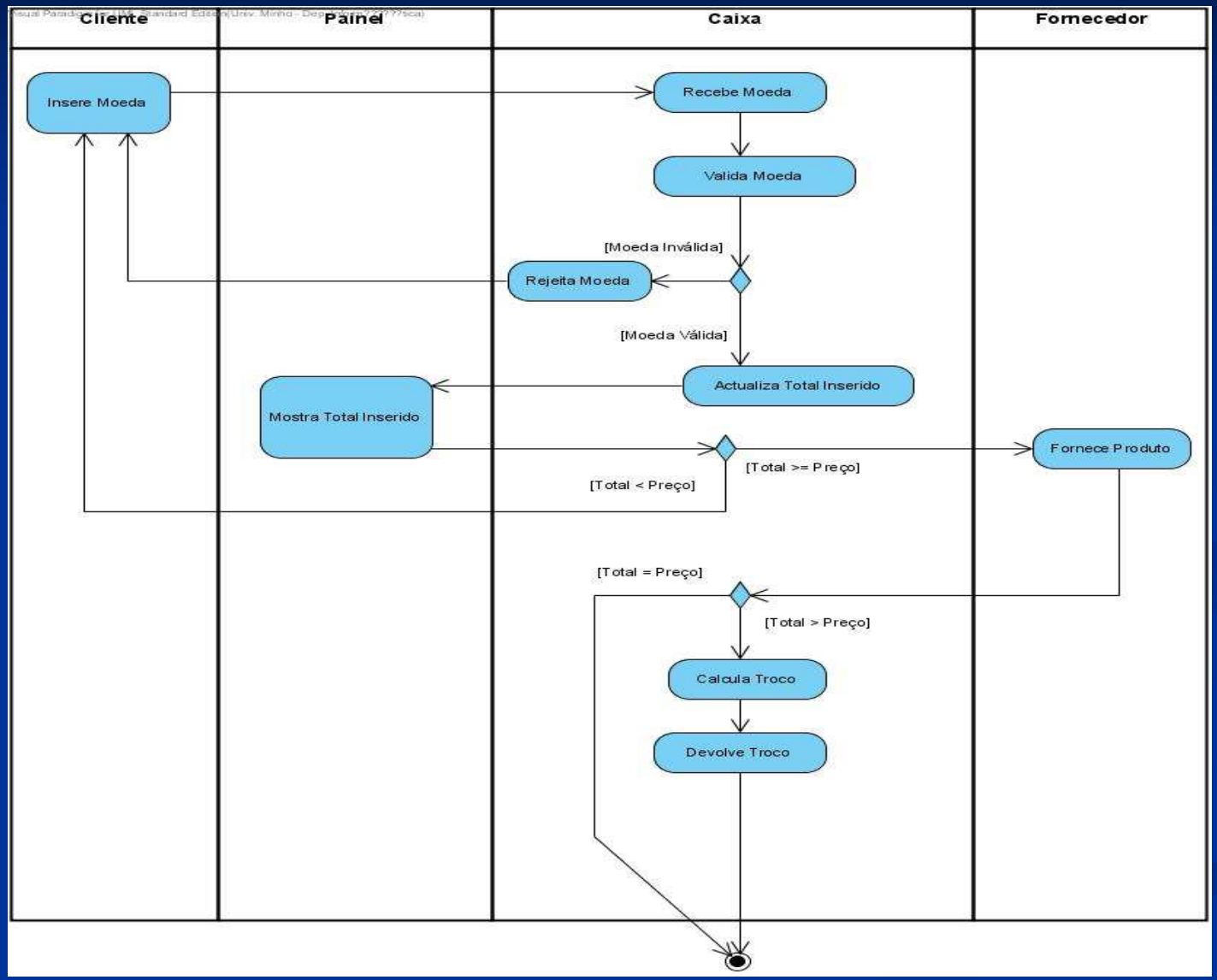


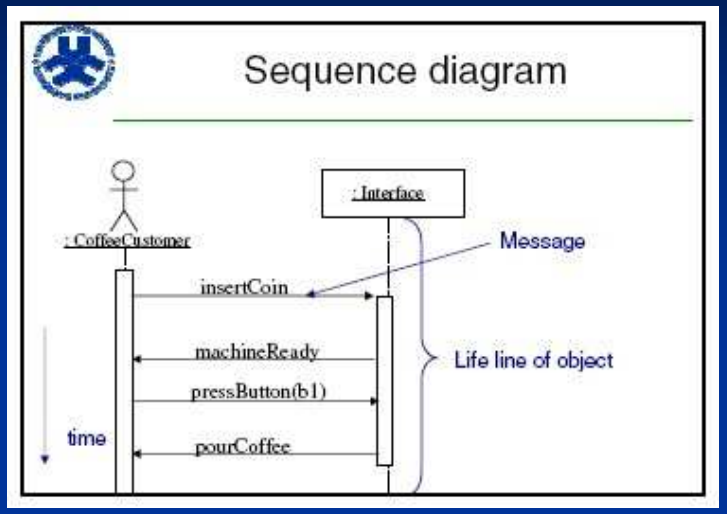
Diagrama de Actividades com sinais, eventos e excepções (mas, e **tempo** ?)

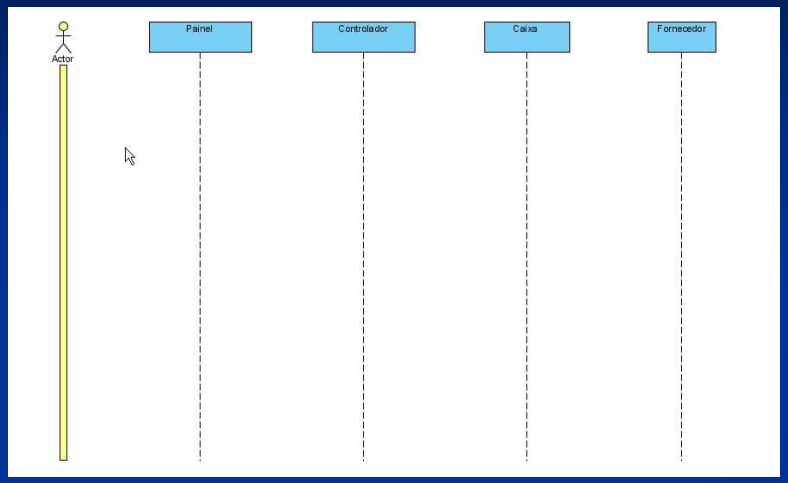
MVx: Comportamento básico

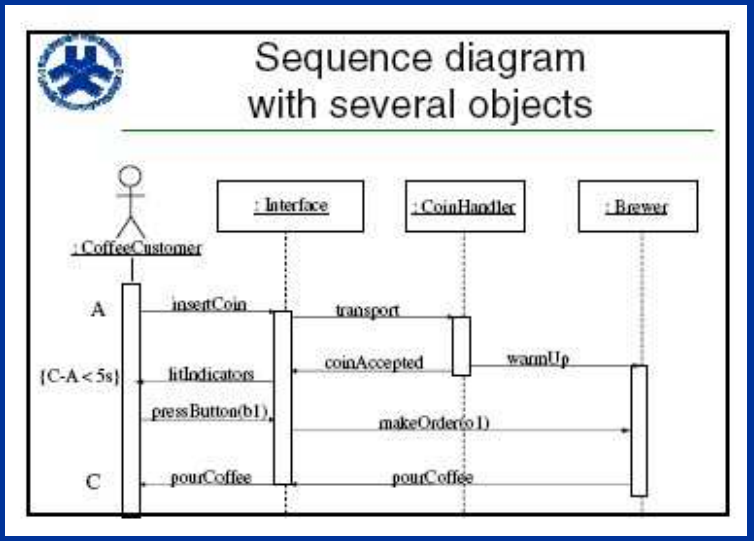


Como vimos, SwimLanes podem ajudar a compreender melhor a sequência de colaborações entre os vários subsistemas e o exterior

MVx: Comportamento básico



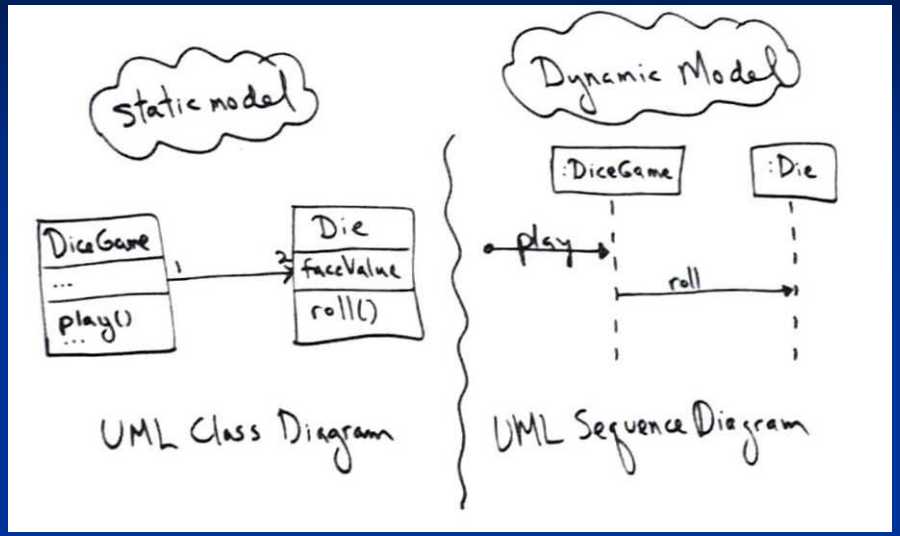




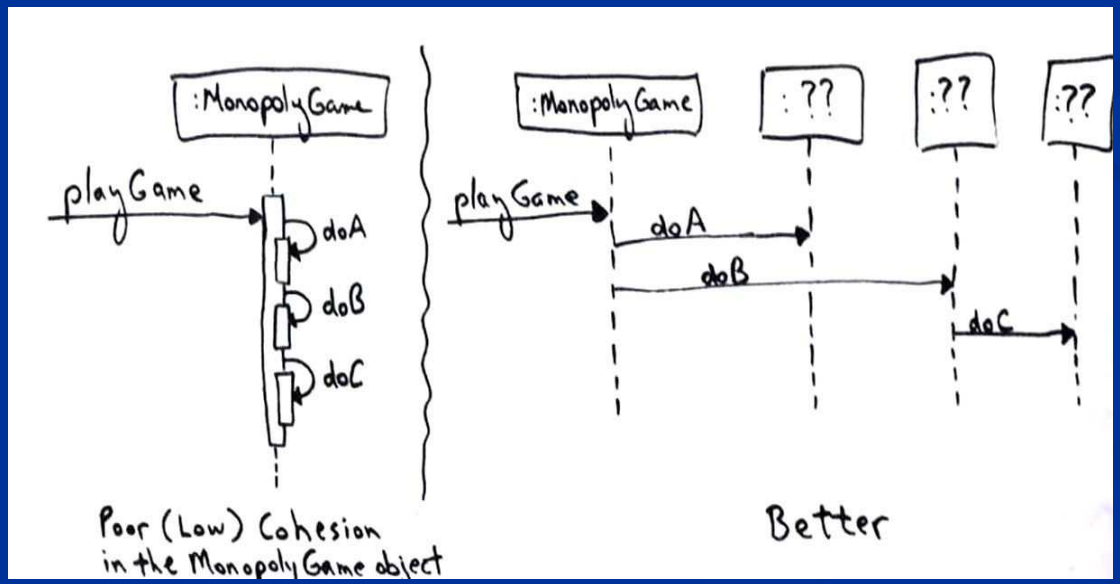
Diagramas de Sequência ajudam a compreender o funcionamento interno, mas revelam-se, ainda assim, pouco expressivos.

Mas permitem testar cenários especiais.

Mas ...



Foram feitos "esboços" em UML usando vários tipos de notações e modelos, nem sempre com a abstracção e iteração que os projectos justificam, talvez porque a equipa de projecto fosse difusa e as responsabilidades também.



"Esboços" (rascunhos, esquiços) significam só por si discussão de ideias, iteração, etc. Será que tal pode ser ensaiado em sala de aula ? Sim, mas com prévia atribuição de funções. Mas todos devem saber exercer a sua. Impossível ??

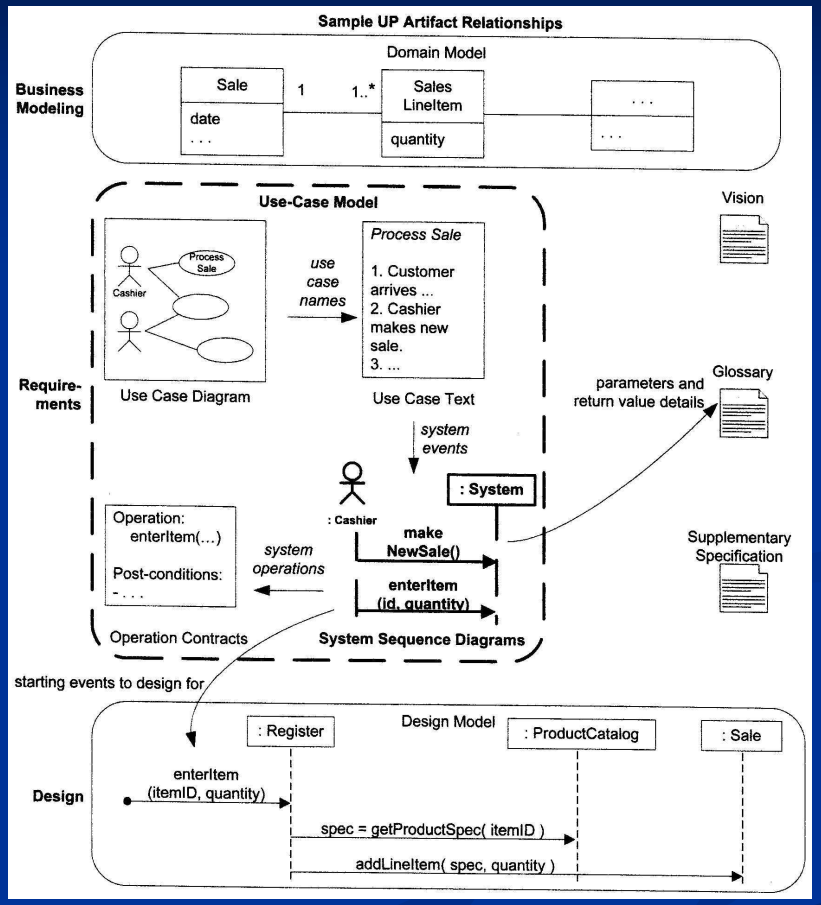
Modelação de SI

Devemos abordar "case studies" que reflictam o que, neste contexto, é o nosso principal objectivo:

Modelar Sistemas de Informação.

E com método, porque ....

Há muito que ligar entre si de forma coerente ... e porque ...



"For every complex problem, there is a simple answer, and it's wrong"

-H.L. Mencken

Case Study 2: SGCV

PROJECTO 2:

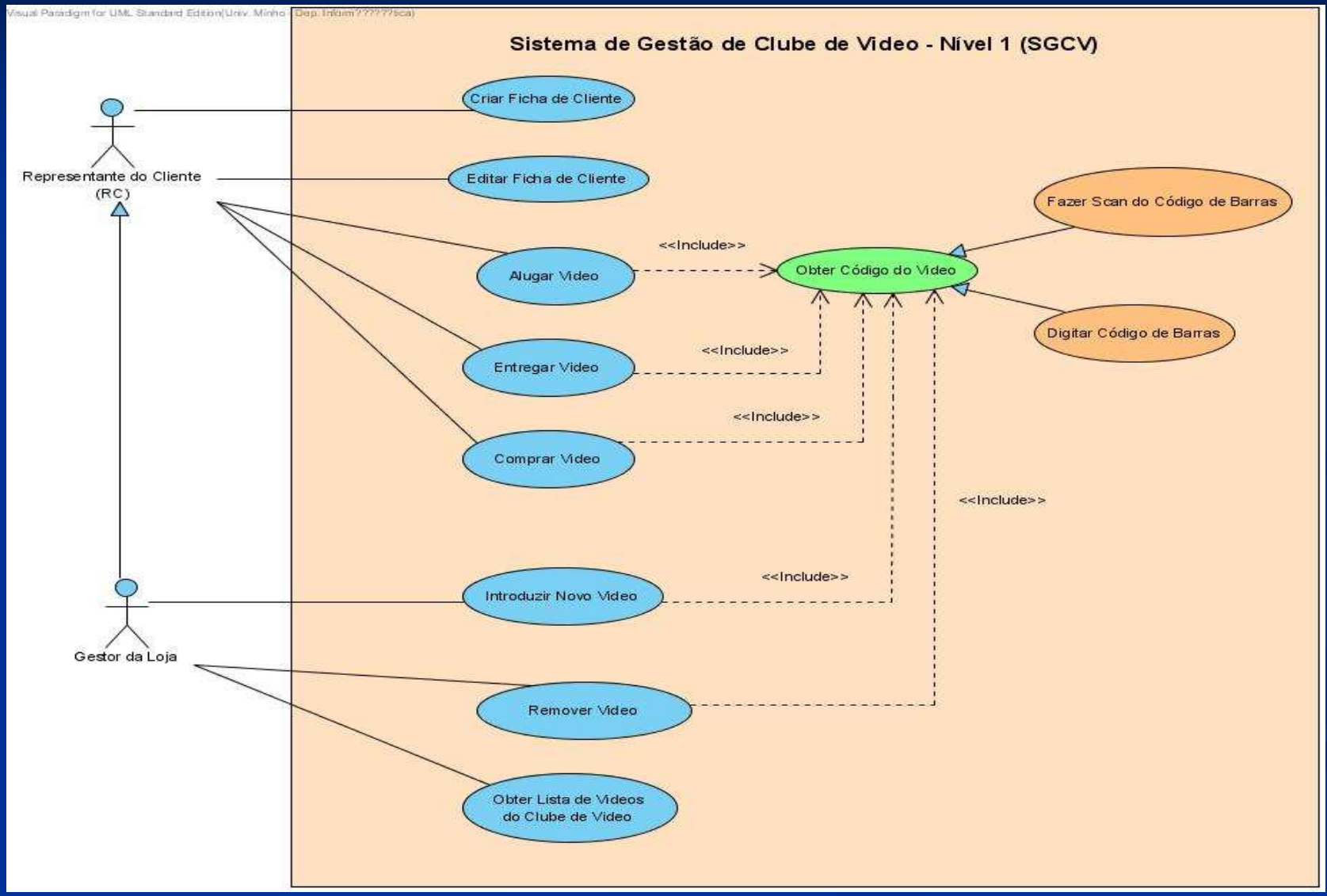
Concepção de um Sistema de Gestão de um Clube de Video (SGCV), no qual um funcionário deve poder realizar, usando o sistema, as usuais operações de: Vender, Alugar, Receber, etc., videos. Um gestor da loja pode adicionar novos videos e abater videos antigos. Cada video tem um código único, cada cliente tem um código único, existe um preço de aluguer por dia, etc.

Questão A) Partir de um primeiro diagrama de Use Cases;

Questão B) Especificar os Use Cases textuais;

Questão C) Fazer diagramas de actividade e sequência e procurar inferir regras de sistematização da modelação (sucesso vs. excepção), temporizações, sub-diagramas, etc.

SGCV: Use Cases



Ideias Importantes 1

The power of the use-case technique lies in getting users and analysts alike to explore how to handle alternate situations. Terminator flows don't explore anything at all. When a developer programs the Basic Flow, the terminator flows are there by default, as manifested by UNIX Core Dump, Windows General Protection Fault (GPF), Java Stack Trace, System Reboot, NT blue screen, and so on.

If you are a movie buff like me, you will remember the line, "I'll be back!"[1] Likewise, we should think about how our use case can resume and continue the flow. A useful system always seeks to either resolve or steer around problems, back toward a successful transaction. To do this, we as analysts must understand our users' and other stakeholders' mentalities, as well as the diversity in the business domain, and the technical challenges to be resolved.

Ideias Importantes 2

Rather than describe use cases head-on, I find it easier to sneak up on them from behind and start by describing scenarios. A **scenario** is a sequence of steps describing an interaction between a user and a system. So if we have a Web-based on-line store, we might have a Buy a Product scenario that would say this:

> *The customer browses the catalog and adds desired items to the shopping basket. When the customer wishes to pay, the customer describes the shipping and credit card information and confirms the sale. The system checks the authorization on the credit card and confirms the sale both immediately and with a follow-up e-mail.*

This scenario is one thing that can happen. However, the credit card authorization might fail, and this would be a separate scenario. In another case, you may have a regular customer for whom you don't need to capture the shipping and credit card information, and this is a third scenario.

All these scenarios are different yet similar. The essence of their similarity is that in all these three scenarios, the user has the same goal: to buy a product. The user doesn't always succeed, but the goal remains. This user goal is the key to use cases: A **use case** is a set of scenarios tied together by a common user goal.

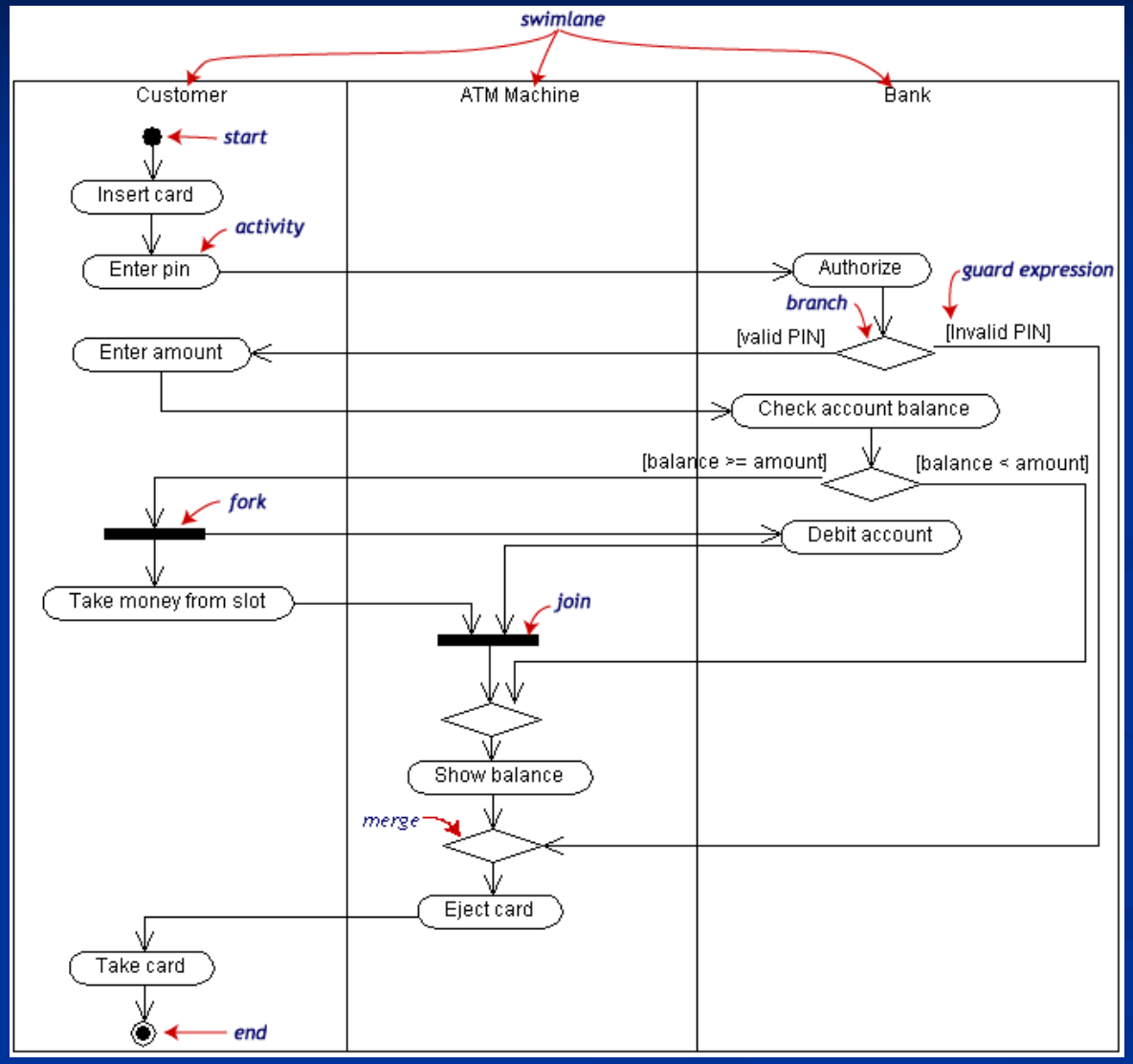
Ideias Importantes 3

- Não disse que as ideias anteriores eram minhas porque não eram. Mas, ao apresentá-las, quero dizer que é com base nelas que estou a tentar desenvolver uma metodologia que visa uma melhor estruturação e domínio da modelação em UML, sempre tendo por base uma estratégia "use-case driven".

- O primeiro passo importante é reconhecer que Use Cases ditos "normais" são "parciais", ou seja, nem sempre resultam em sucesso !!

- O segundo é reconhecer que Use Cases "normais" têm sempre cenários alternativos e cenários de erro. Assim, não vale a pena grande ênfase em Use Cases "totais". Talvez sim "partir" destes para depois pensar no que pode "correr mal".

Ideias Importantes 4



Misturar as diferentes situações é apenas fugir à realidade e confundir os objectivos.

Como vimos antes, as coisas na realidade são como se apresenta a seguir …

Ideias Importantes 5

Use case: Withdraw Money
Main Success Scenario:

TRANSACÇÕES

1. - user identifies himself by a card
2. system reads the bank ID and account number from card and validates them
3. - user authenticates by PIN
4. system validates that PIN is correct
5. - user requests withdrawal of an amount of money
6. system checks that the account balance is high enough
7. system subtracts the requested amount of money from account balance
8. system returns card and dispenses cash

Sucesso

Extensions:

2-8a. Wrong type of card or destroyed card:
1. System gives back the card

4a. Wrong pin less than 3 times:
1. System updates number of tries
2. start from action step 3

4-8a. Wrong pin 3 times:
1. System keeps the card

6-8a. Not enough money on account:
1. account does not contain enough money
2. system does not change the account
3. system returns card

substituição

Alternativas, Insucessos e tentativas de recuperar o sucesso

SGCV: Use Case Comum

Introduzir Novo Video – Use Case

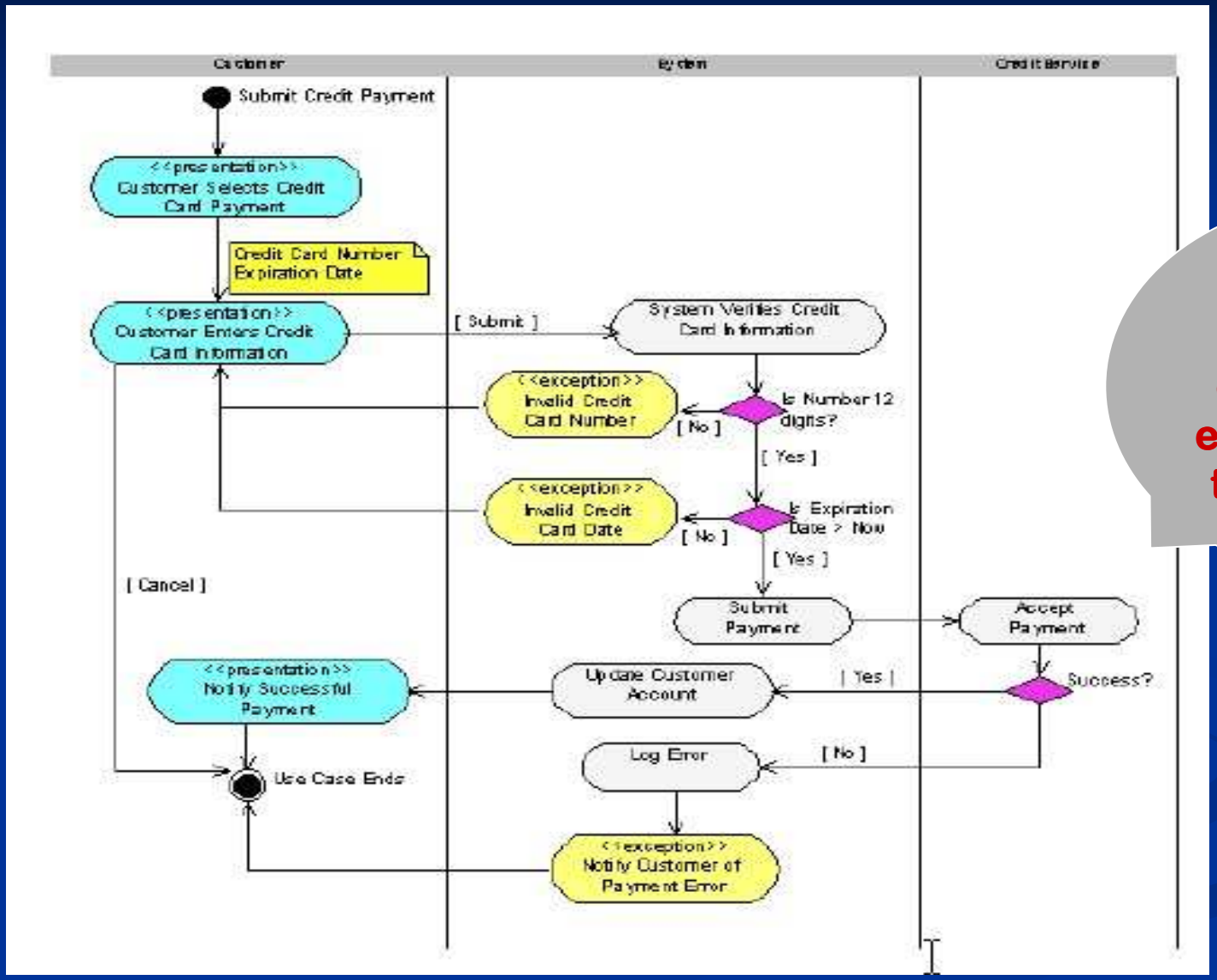
Use Case Description

Full			
Use Case ID	*Introduzir novo Video*		
Primary Actor	Gestor da Loja		
Brief Description	Adicionar um novo video à loja		
Preconditions			
Flow of Events		Actor Input	System Response
	1	*Obter Código do Video*	
	2		Validar ID Video
	3	Introduzir dados do Video	
	4		Registar os dados do Video
	5		Dar informação de registo OK
Post-conditions	Um novo Video foi registado no sistema para aluguer/compra		
Exception1	2a ID Video já existe ..Dar informação de erro no código do video		
Exception2	4a Erro no registo ..Dar informação de erro no registo		
Author	FMM		

Cenários de Sucesso

Insucesso
Como tratar ?

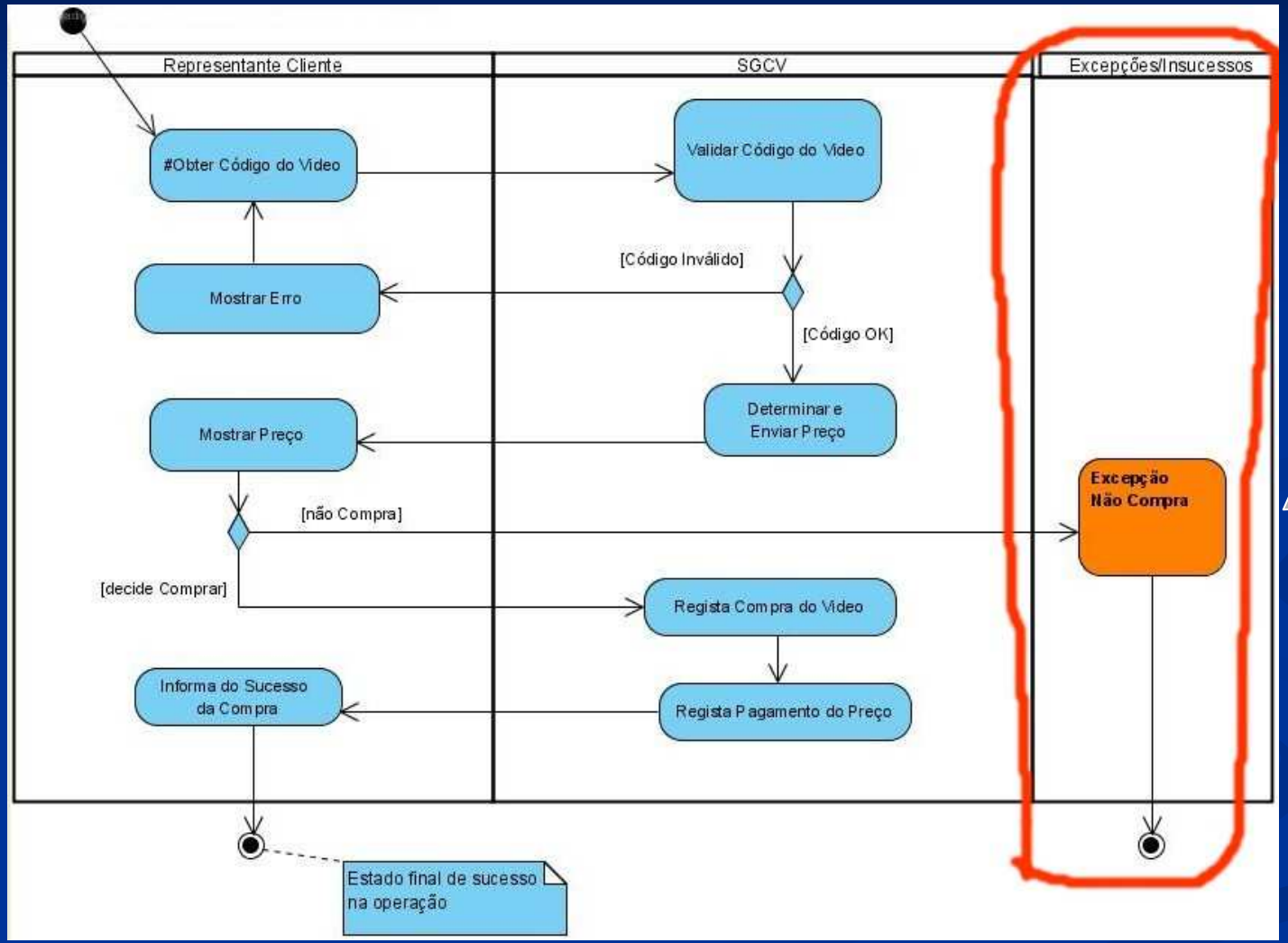
Extensão aos DA



Usar cores e estereótipos especiais para tais acções ?

É uma solução possível …

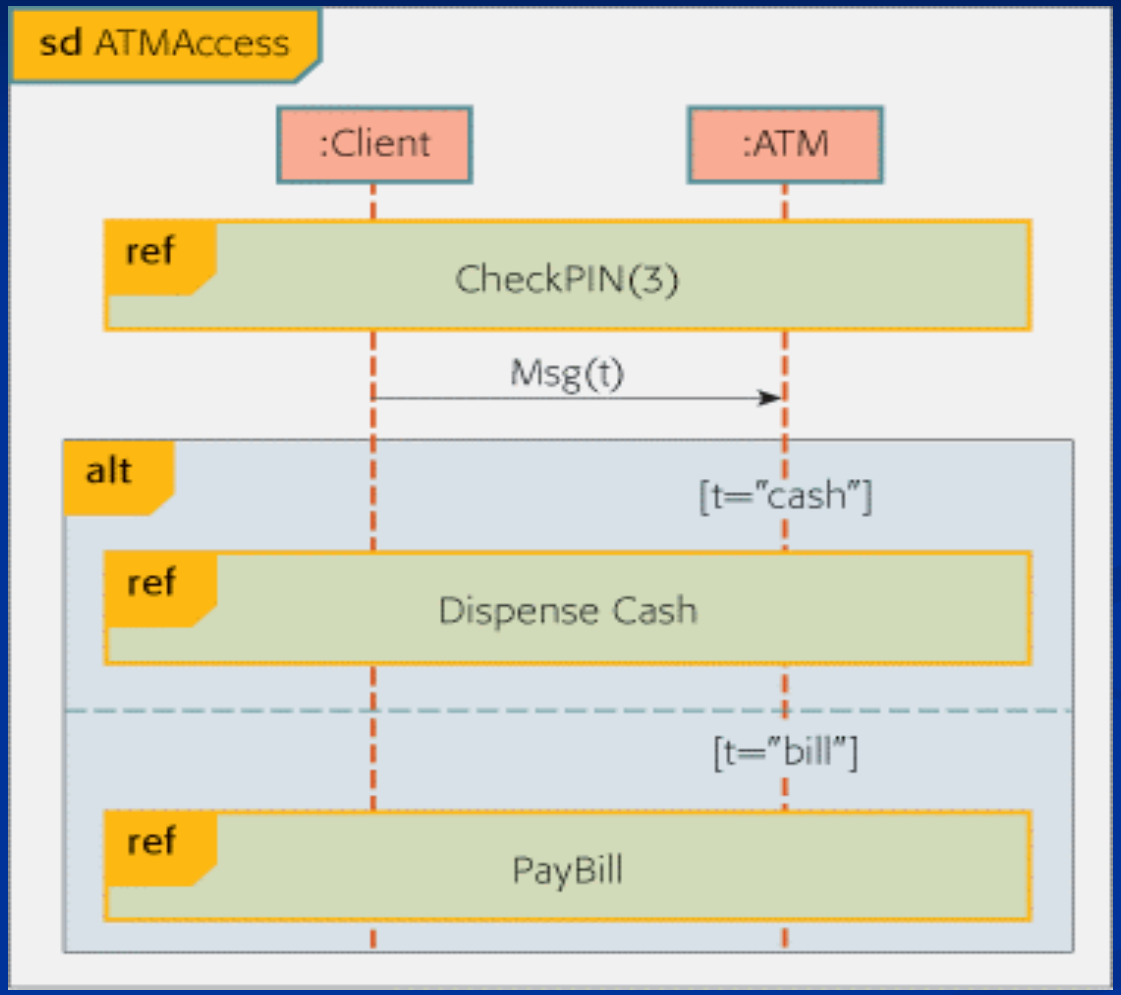
Proposta: Extensão aos DA



Criar SwimLane de Insucesso

1 geral ou 1 para cada entidade

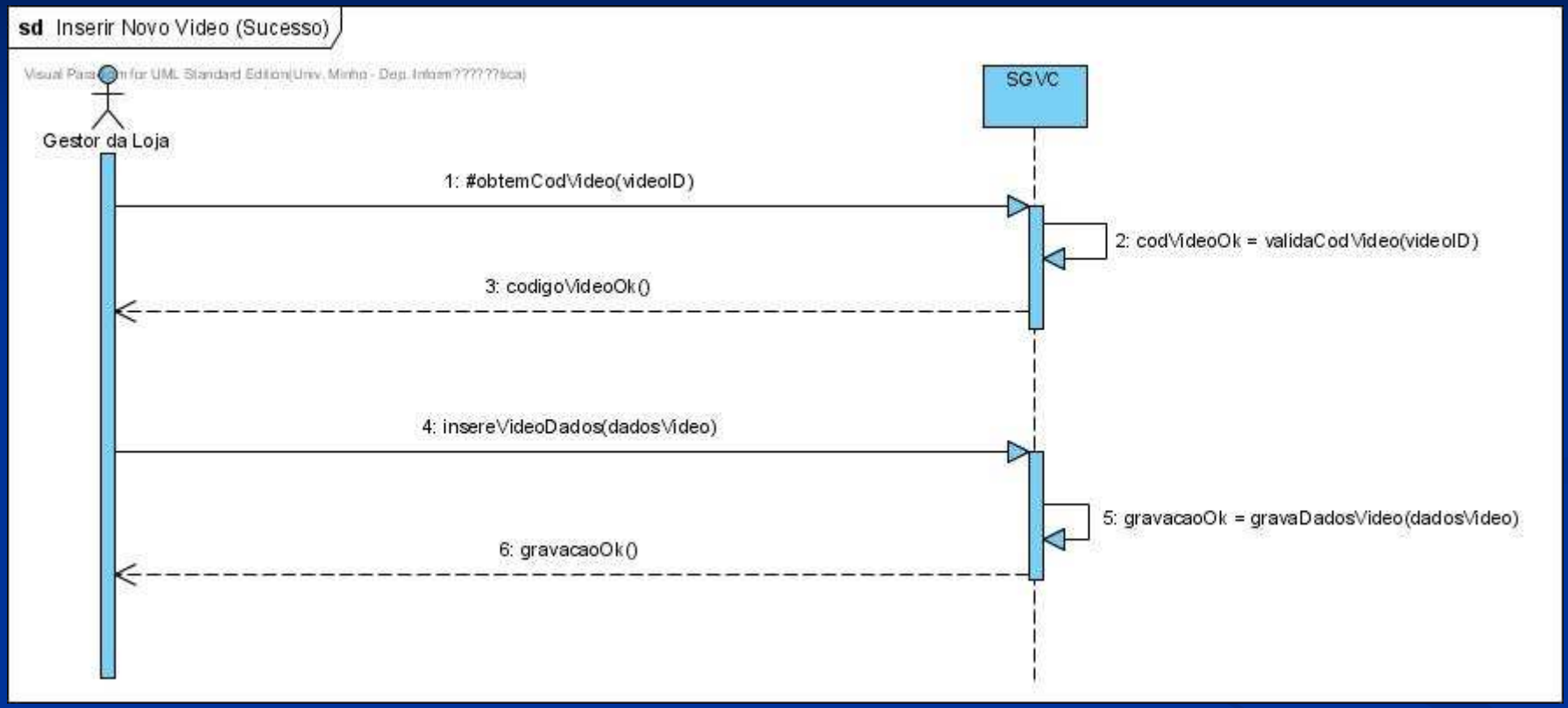
Modularidade em Diag. Sequência



A possibilidade de termos diagramas de sequência com modularidade permite-nos agora gerir tal modularidade, ou seja, ter uma regra (ou mais) que nos diga(m) "por onde" devemos "partir" estas sequências, isto é, com que critério ou método.

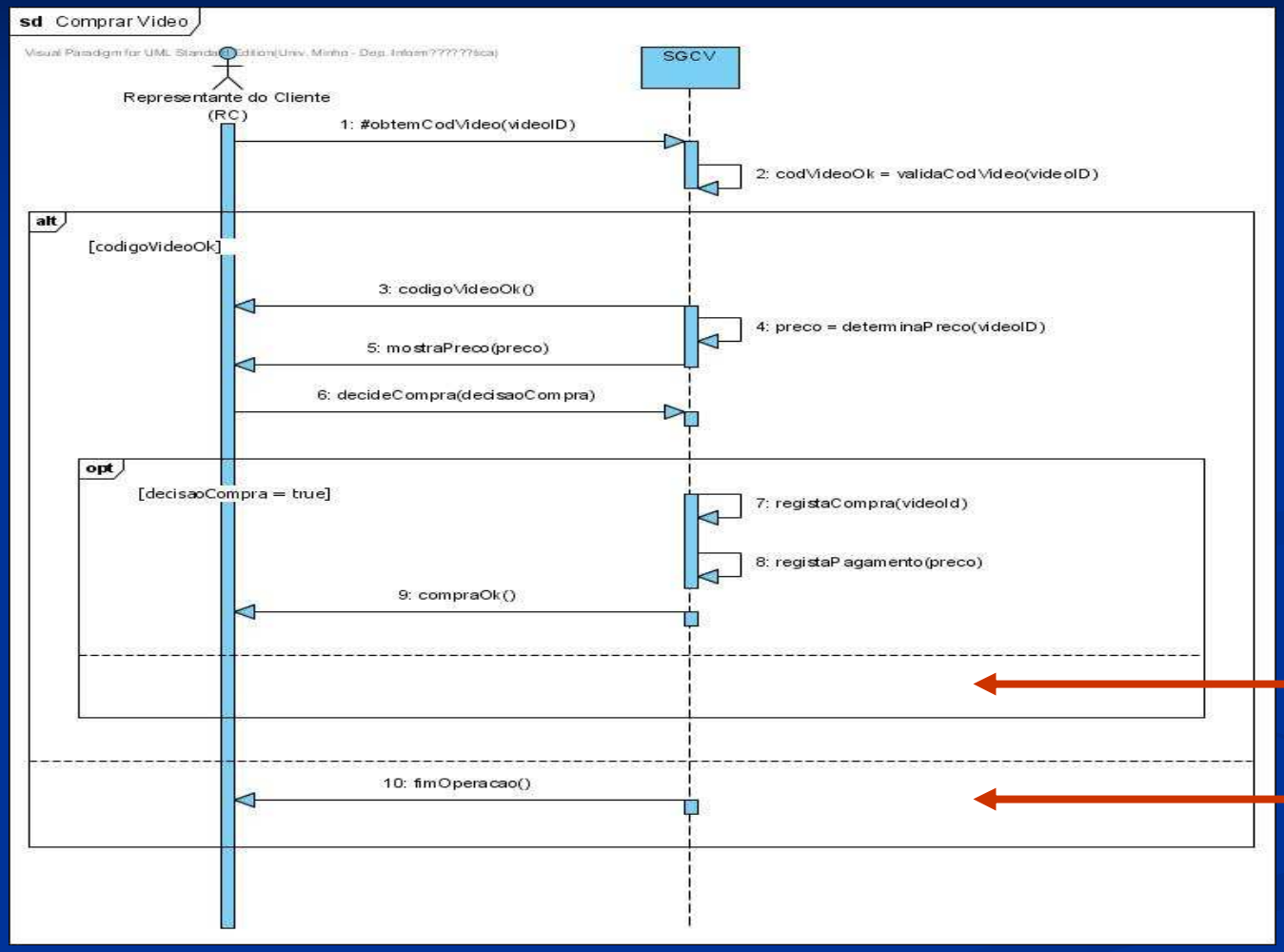
Naturalmente que isto apenas tem interesse para além das partições óbvias, como no exemplo.

Cenário de Sucesso: partida



O cenário de sucesso (ou os cenários de sucesso normais) são apenas pontos de referência para a definição de todas as situações de erro (tratáveis – recuperáveis ou não).

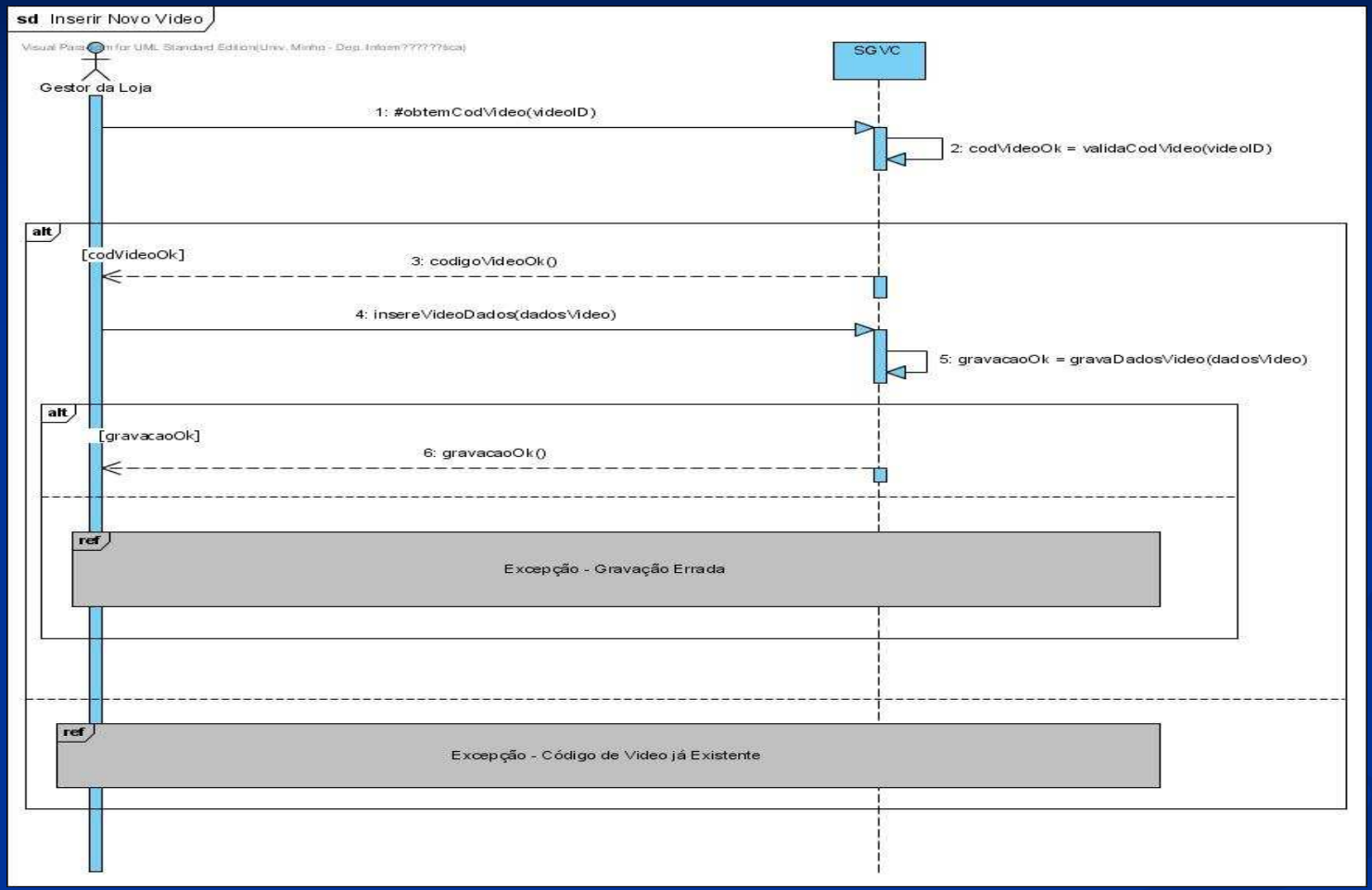
Modularidade em Comprar Video



É melhor que nada, mas não estrutura a semântica da especificação.

Os 2 casos possíveis de insucesso

Proposta Metodológica



"Partir" pelas situações de insucesso, usando subdiagramas

Apresentam-se todas as trocas de mensagens do comportamento de sucesso e "escondem-se" os comportamento de insucesso em subs.

Proposta Metodológica: Síntese

- Esta é a proposta metodológica a apresentar neste momento em que apenas nos estamos a concentrar em aspectos relacionados com a fase de ANÁLISE.

- Os Use Cases textuais, os Diagramas de Actividade e os de Sequência devem todos reflectir a ideia, ou seja, estarem "síncronos", coerentes.

- Os Use Cases textuais devem ser escritos segundo uma perspectiva transaccional, o que facilitará a sua interpretação operacional futura (cf. mensagens e métodos).

Continuação

▣ Não tendo ainda completado completamente as questões de Arquitectura (Modelação em UML) vamos começar a pensar também em questões relacionadas com a Engenharia Civil (OO Design).

Mas … agora vamos festejar o Natal, e … até nesta questão quase universal, "a quem tal se aplique " !!