

# Processo Unificado

---

Viviane Torres da Silva  
viviane.silva@ic.uff.br

<http://www.ic.uff.br/~viviane.silva/es1>

# Agenda

---

- Processo de Software
- Desenvolvimento Iterativo
- Desenvolvimento Evolutivo
- Desenvolvimento Ágil
- Processo Unificado
- Fronteira entre análise e projeto

# Processo de Software

---

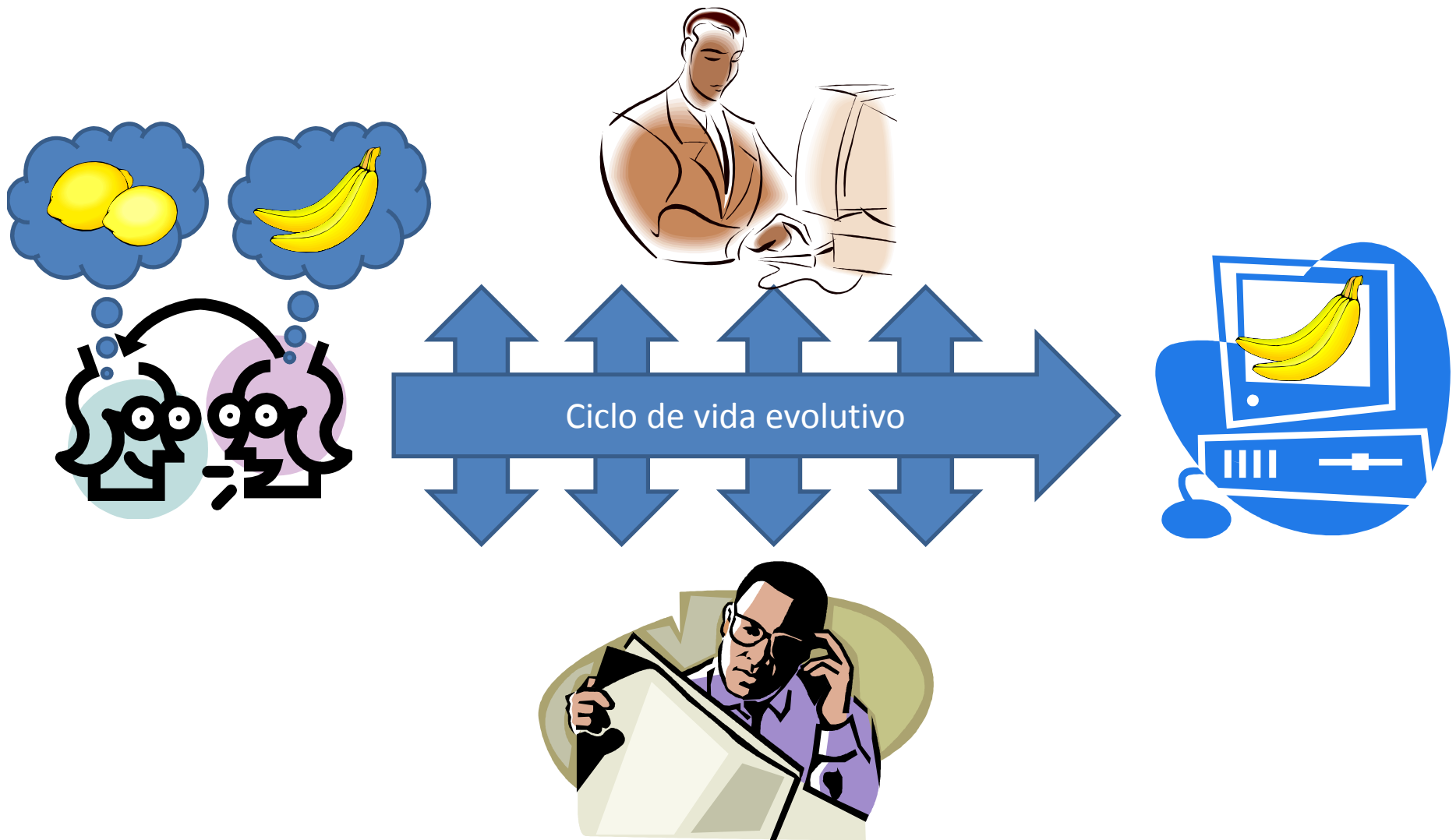
---



# Processo de Software

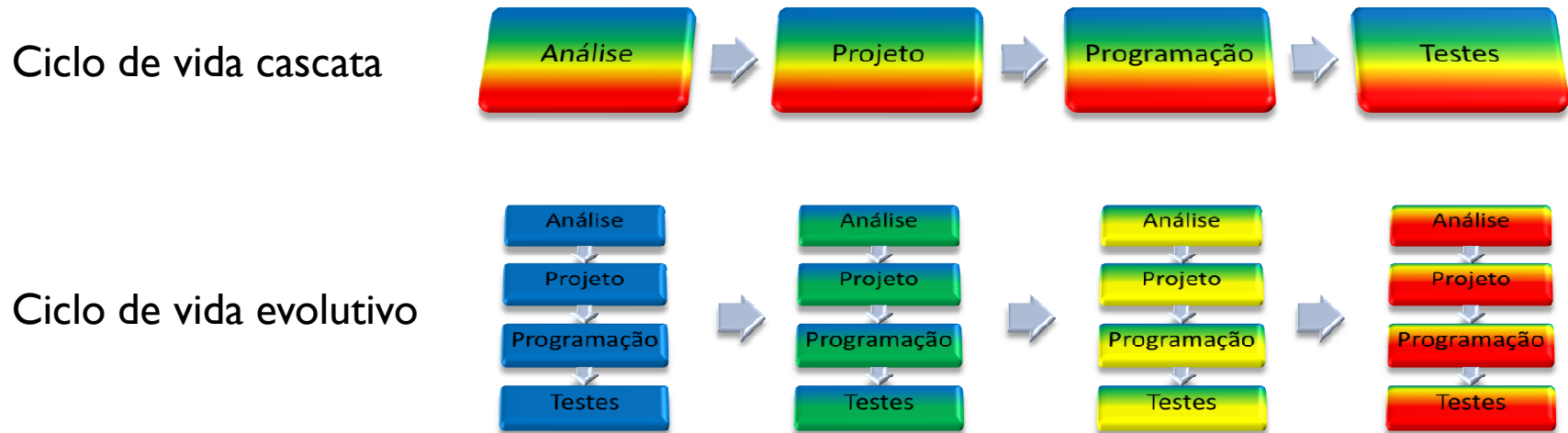
---

---



# Processo de Software

---



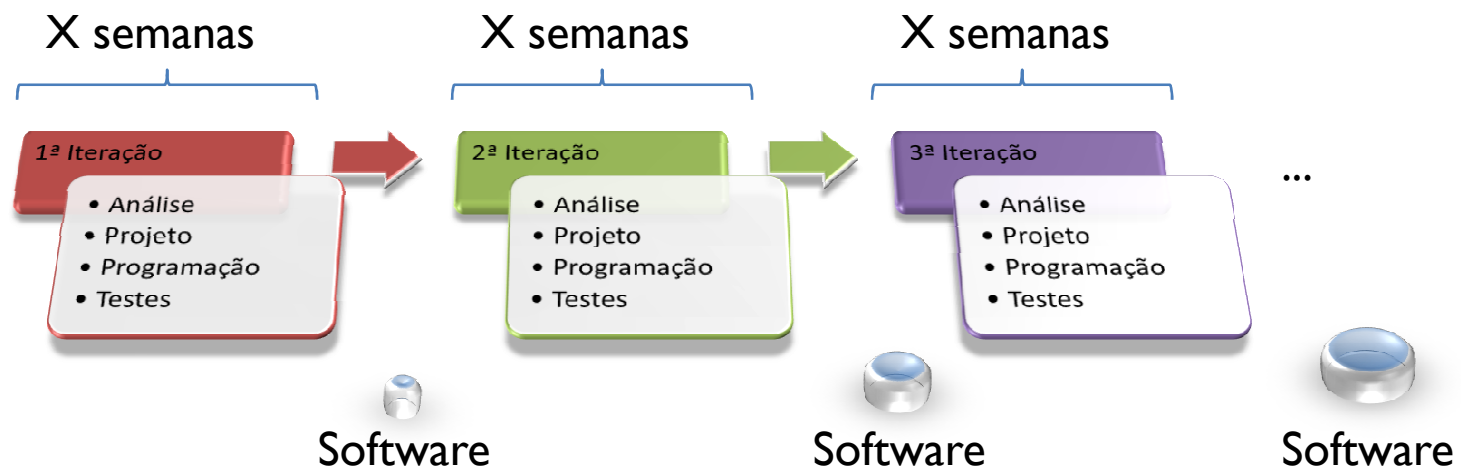
➤ Objetivo: Processo Unificado com aspectos de...

- Desenvolvimento iterativo
- Desenvolvimento evolutivo
- Desenvolvimento ágil

# Desenvolvimento Iterativo

---

- O desenvolvimento é organizado em “mini-projetos”
  - Cada “mini-projeto” é uma iteração
  - Cada iteração tem duração curta e fixa (de 2 a 6 semanas)
  - Cada iteração tem atividades de análise, projeto, programação e testes
  - O produto de uma iteração é um software parcial



# Desenvolvimento Iterativo

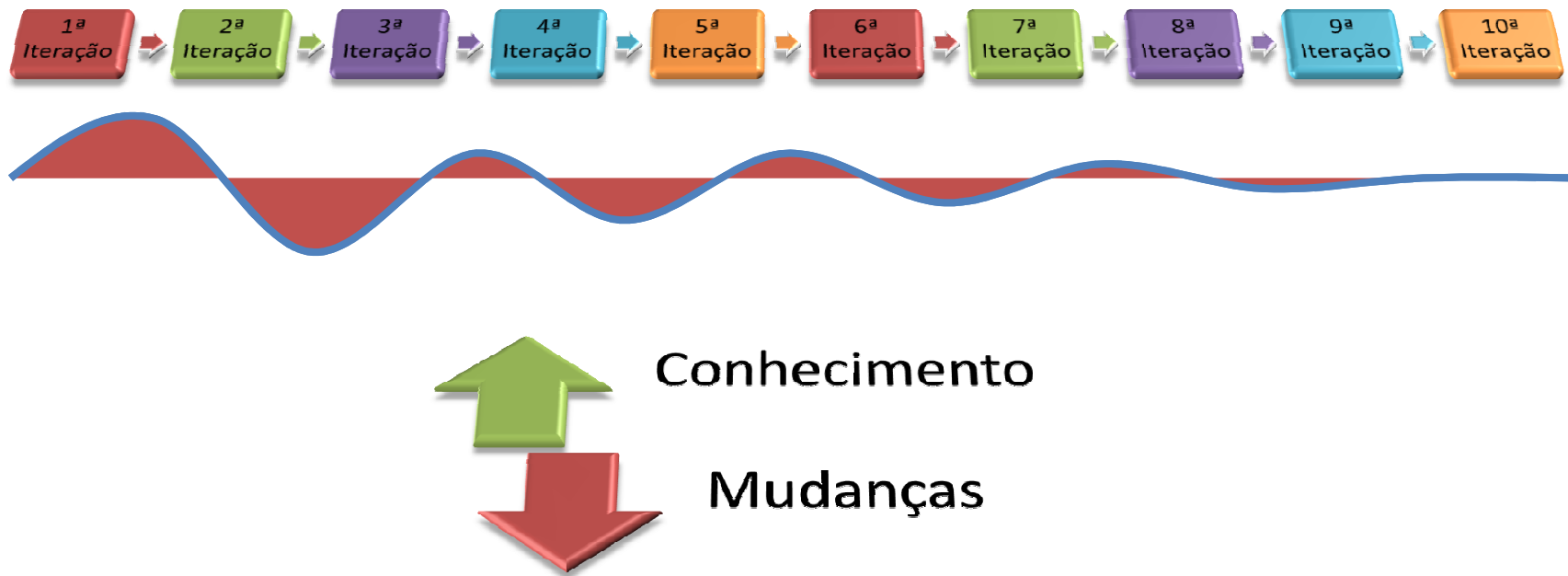
---

- A iteração deve ser fixa
  - A iteração nunca deve passar da duração previamente estipulada
  - Porém tarefas podem ser removidas ou incluídas
- O resultado de cada iteração é um software...
  - Incompleto
  - Em desenvolvimento (não pode ser colocado em produção)
  - Mas não é um protótipo!!!
- Esse software pode ser verificado e validado parcialmente
  - Testes
  - Usuários
- Podem ser necessárias diversas iterações (e.g. 10 a 15) para ter uma versão do sistema pronta para entrar em produção

# Desenvolvimento Iterativo

---

- Iterações curtas privilegiam a propagação de conhecimento
  - Aumento do conhecimento sobre o software
  - Diminuição das incertezas, que levam às mudanças

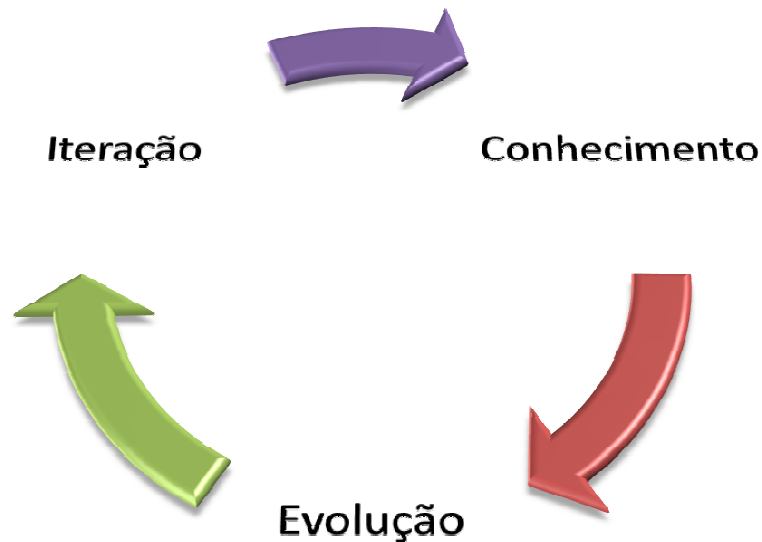




# Desenvolvimento Evolutivo

---

- As especificações evoluem (transformam) a cada iteração
  - A cada iteração, uma parte do software fica pronta
  - O conhecimento sobre o software aumenta
  - As especificações são evoluídas para retratar esse aumento de conhecimento sobre o que é o software



# Desenvolvimento Evolutivo

---

- Mudanças sempre acontecem em projetos de software
  - Requisitos mudam
  - O ambiente em que o software está inserido muda
  - As pessoas que operam o software mudam
  
- Estratégias para lidar com mudanças
  - Evitar as mudanças (corretivas) fazendo uso de boas técnicas de engenharia de software
  - Acolher mudanças por meio de um processo evolutivo

# Desenvolvimento Ágil

---

- São dadas respostas rápidas e flexíveis a mudanças
  - O projeto é replanejado continuamente
  - São feitas entregas incrementais e constantes do software, refletindo as mudanças solicitadas



# Desenvolvimento Ágil

---

## ➤ Princípios ágeis

- Satisfazer o cliente
- Acolher modificações nos requisitos
- Entregar o software com frequência
- Trabalhar junto ao cliente
- Manter os indivíduos motivados
- Promover conversas face a face
- Medir o progresso com software funcionando
- Manter um ritmo constante de trabalho
- Cuidar da qualidade
- Buscar por simplicidade
- Trabalhar com equipes auto-organizadas
- Ajustar o comportamento da equipe buscando mais efetividade

# Modelagem Ágil

---

- Tem intuito de apoiar o entendimento e a comunicação
  - Do problema (análise)
  - Da solução (projeto)
- Tem caráter exploratório
  - Não visa ser completa
  - Foca nos aspectos mais complexos e incomuns
- Foco no código
  - Enxerga o código como o “verdadeiro” projeto
  - Todos os modelos anteriores podem estar desatualizados ou serem imprecisos
- Não pretende ser o meio de comunicação principal
  - Deve ser feito pelos próprios desenvolvedores e não por equipes separadas

# Modelagem Ágil

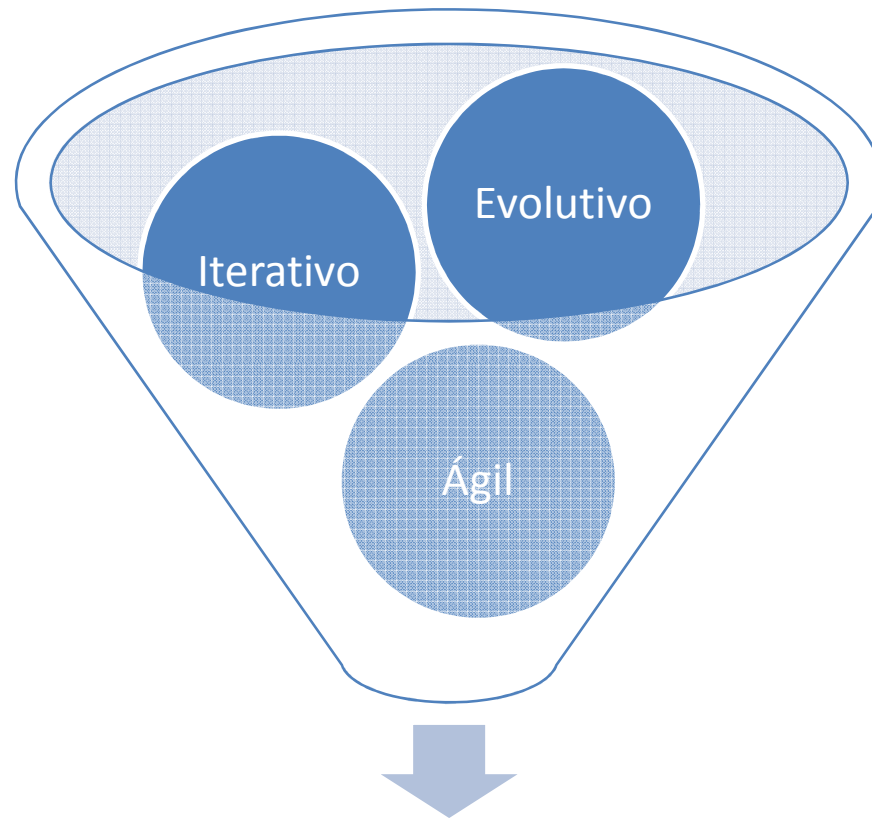
---

- Incentiva a modelagem em pares
  - Ou pequenos grupos
- Apóia a criação de visões do modelo em paralelo
  - Estática (e.g., diagrama de classes)
  - Dinâmica (e.g., diagrama de seqüência)
- Usa ferramentas simples
  - Quadro-branco + câmera digital
  - Ferramentas CASE
  - Editor de texto
- Utiliza simplificações da notação sempre que possível
  - Não usa todos os recursos da UML
  - Utiliza recursos adicionais se necessário

# Processo Unificado

---

---



Processo Unificado

# Processo Unificado

---

## Benefícios esperados

- Suavizar os riscos precoces
- Aumento da visibilidade do progresso
- Foco no envolvimento e comprometimento do usuário
- Controle sobre a complexidade
- Aprendizado incremental
- Menos defeitos
- Mais produtividade

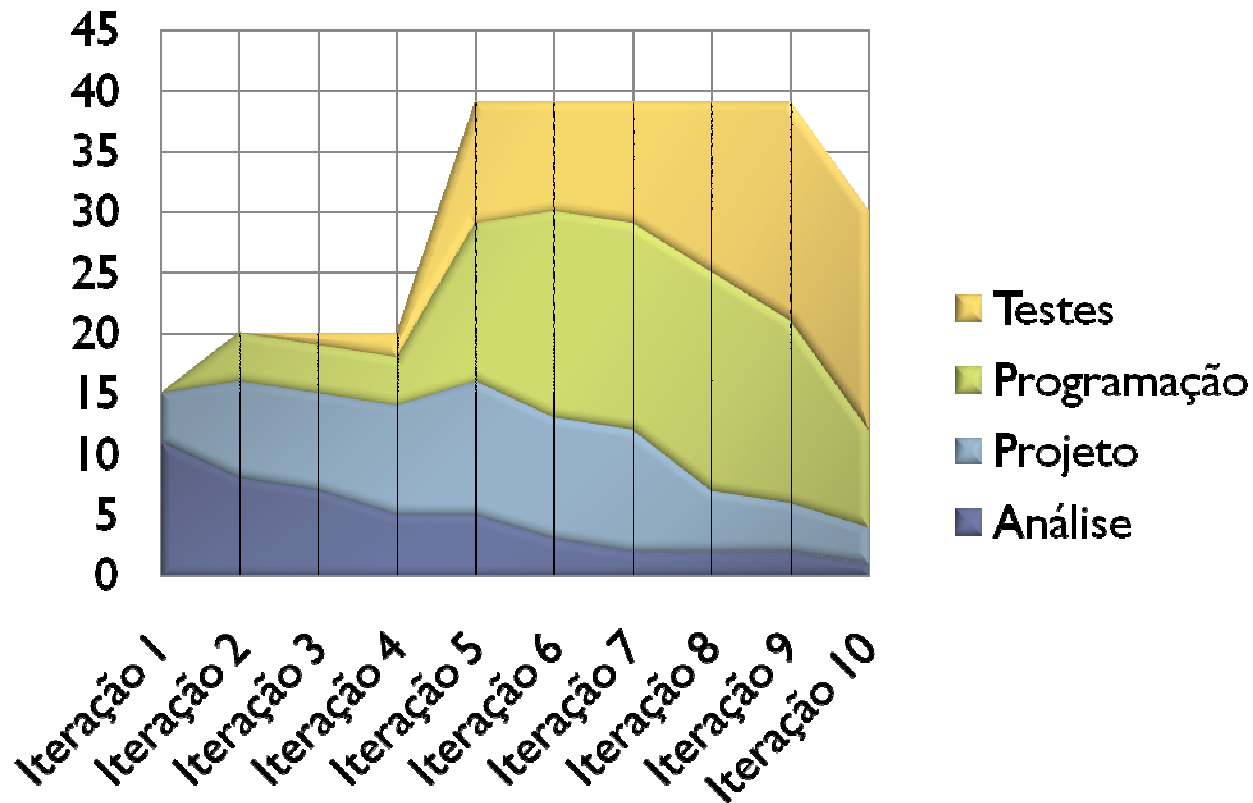


# Processo Unificado: Exemplo

---

- Analisar os requisitos no início do projeto
  - Casos de uso
  - Lista de requisitos não funcionais
- Priorizar os casos de uso
  - Significativos para a arquitetura como um todo
  - Alto valor de negócio
  - Alto risco
- Em cada iteração
  - Selecionar alguns casos de uso por ordem de prioridade para serem analisados em detalhes
  - Atribuir tarefas para a iteração a partir da análise detalhada desses casos de uso
  - Fazer projeto e programação de parte do software
  - Testar a parte do software recém projetada e programada e criar a *baseline* da iteração
  - Apresentar a *baseline* da iteração ao usuário

# Processo Unificado: Exemplo

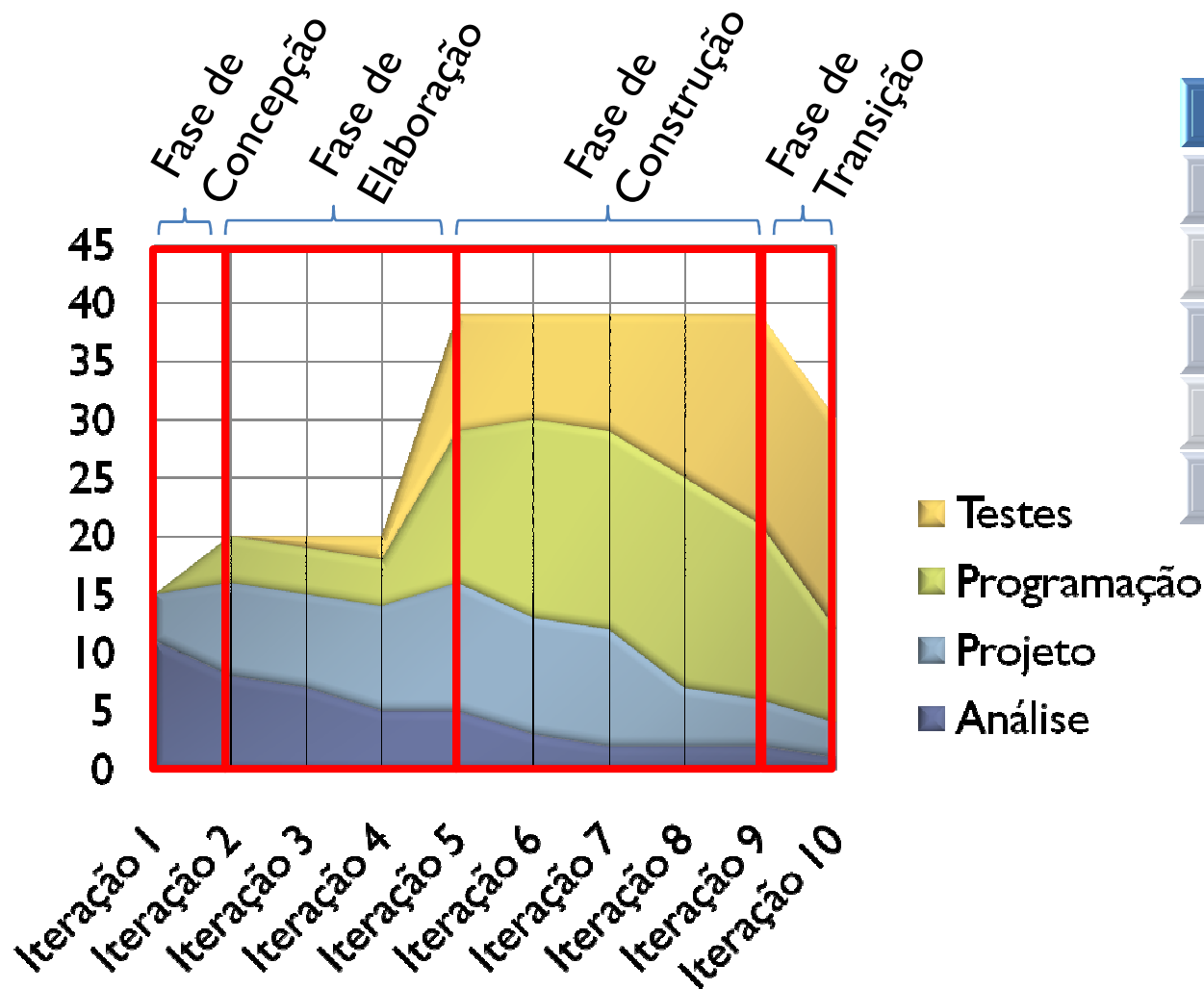


# Processo Unificado: Fases

---

- O desenvolvimento pode ser decomposto em fase, com o intuito de retratar a ênfase principal das iterações
  - Concepção: entendendo o sistema
  - Elaboração: elaborando a solução
  - Construção: implementando a solução
  - Transição: testando a solução
  
- Plano da fase
  - Abrangente e superficial
  
- Plano da iteração
  - Específico e detalhado

# Processo Unificado: Exemplo



Atividade	Esforço
Análise	10%
Projeto	15%
Programação	30%
Testes	15%
Gerência	30%

# Processo Unificado: Concepção

---

## ➤ Consiste de

- Identificação de riscos
- Listagem inicial dos requisitos
- Esboço dos casos de uso
- Identificação de arquiteturas candidatas
- Estimativas iniciais de cronograma e custo

## ➤ Principais características

- Menor fase do projeto
- Escopo ainda vago
- Estimativas ainda vagas

## ➤ Esforço e duração aproximados

- 5% do esforço do projeto
- 10% da duração do projeto

# Processo Unificado: Elaboração

---

## ➤ Consiste de

- Mitigação dos riscos
- Detalhamento da maioria dos requisitos e casos de uso
- Estabelecimento e validação da arquitetura do software
- Detalhamento das estimativas de cronograma e custo

## ➤ Principais características

- Grande parte das atividades de análise e projeto já concluída
- Diminuição significativa das incertezas
- *Baseline* da arquitetura é estabelecida (definição da arquitetura)

## ➤ Esforço e duração aproximados

- 20% do esforço do projeto
- 30% da duração do projeto

# Processo Unificado: Construção

---

## ➤ Consiste de

- Implementação dos demais componentes da arquitetura
- Preparação para a implantação

## ➤ Principais características

- Maior fase do projeto
- *Baseline* de testes do produto é estabelecida (definição dos testes)

## ➤ Esforço e duração aproximados

- 65% do esforço do projeto
- 50% da duração do projeto

# Processo Unificado: Transição

---

## ➤ Consiste de

- Execução de testes finais
- Implantação do produto
- Treinamento dos usuários

## ➤ Principais características

- *Baseline* de liberação do produto é estabelecida

## ➤ Esforço e duração aproximados

- 10% do esforço do projeto
- 10% da duração do projeto



# Processo Unificado: Características

---

- Os requisitos **não** são completamente definidos antes do projeto
- O projeto **não** é completamente definido antes da programação
- A modelagem **não** é feita de forma completa e precisa
- A programação **não** é uma tradução mecânica do modelo para código
- As iterações **não** duram meses, mas sim semanas
- O planejamento **não** é especulativo, mas sim refinado durante o projeto

# Exercício

---

- Pensando no trabalho da disciplina e conhecendo o Processo Unificado, como vocês aplicarão as idéias do processo unificado?
  - Concepção: entendendo o sistema
  - Elaboração: elaborando a solução
  - Construção: implementando a solução
  - Transição: testando a solução
  
- Pensar nas apresentações

# Análise x Projeto

---

---

# Fronteira entre análise e projeto



---

- A orientação a objetos diminui a distância entre as fases do processo de desenvolvimento;

**Mundo Real**



**Análise & Projeto**

Carro
 velocidade : Integer
 acelera()

**Código**

```
public class Carro
{
    private int velocidade;

    public void acelera()
    {
        velocidade++;
    }
}
```

# Fronteira entre análise e projeto

---

- A orientação a objetos torna nebulosa a fronteira entre análise e projeto:

**Análise**

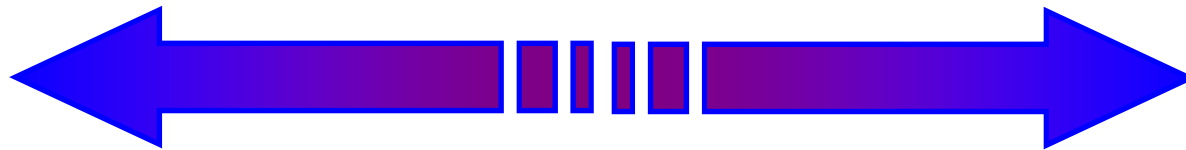


- O que?
- Requisitos
- Investigação

**Projeto**



- Como?
- Solução lógica



Onde termina a análise e começa o projeto?

# Fronteira entre análise e projeto

---

- Principais funções de AOO:
  - Identificar as funcionalidades e entidades do sistema
  - Encontrar abstrações adequadas para representar o problema
- Principais funções de POO:
  - Atribuir responsabilidades às entidades do sistema
  - Encontrar abstrações adequadas para representar a solução
- Ambos são representados através de modelos

**Análise**



Investigação



**Projeto**



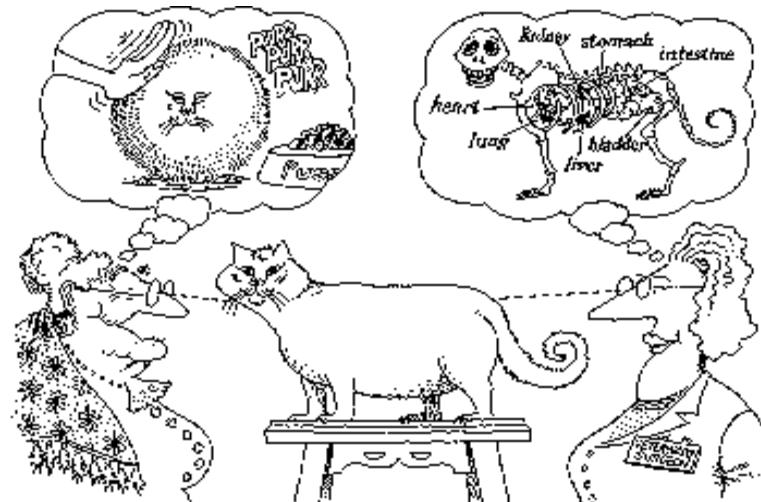
Solução

# Modelos

---

## ➤ O que são modelos?

- Abstrações da realidade
- Focam somente no que realmente interessa para um determinado observador em um dado momento



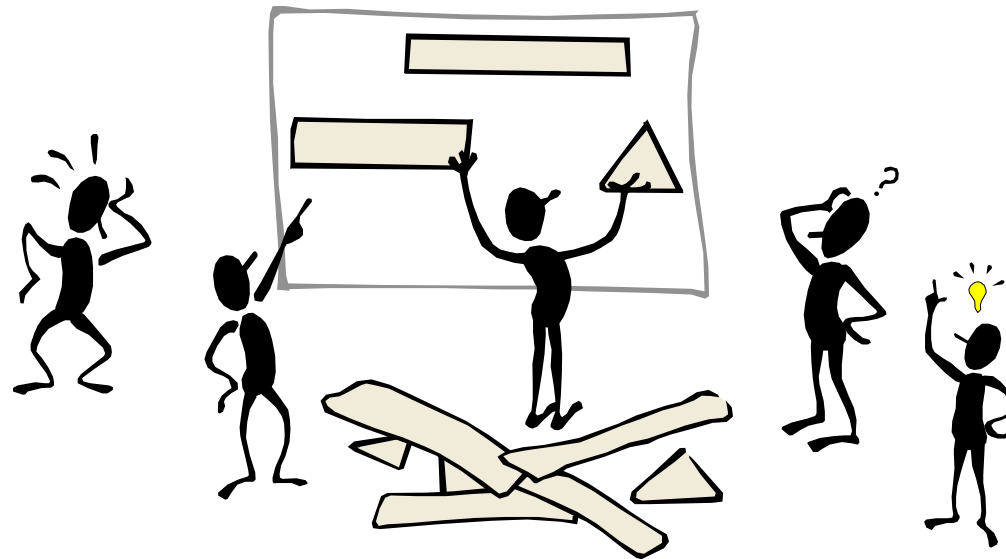
Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.

[Fonte: BOOCH, G., 1993]

# Modelos

---

- Para que modelos são úteis?
  - Possibilitar a comunicação entre pessoas
  - Permitir lidar com problemas complexos
  - Testar hipóteses antes de realizá-las



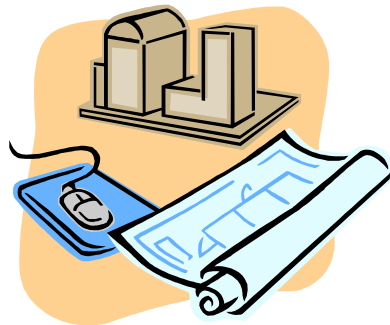
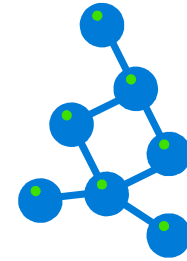
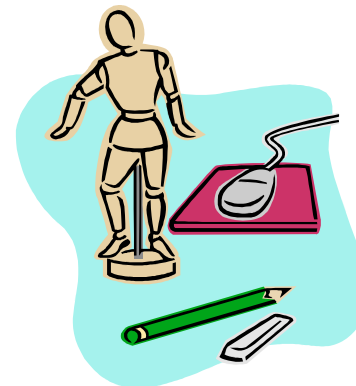
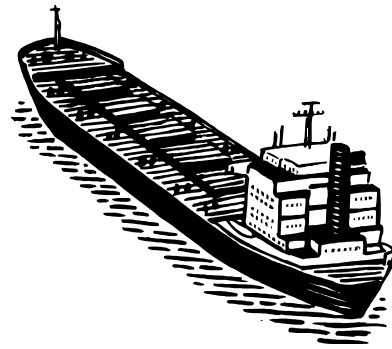


# Modelos

---

## ➤ Quais são as formas de modelos?

- Croquis
- Maquetes
- Manequins
- Plantas
- Diagramas
- Etc.



# Modelos

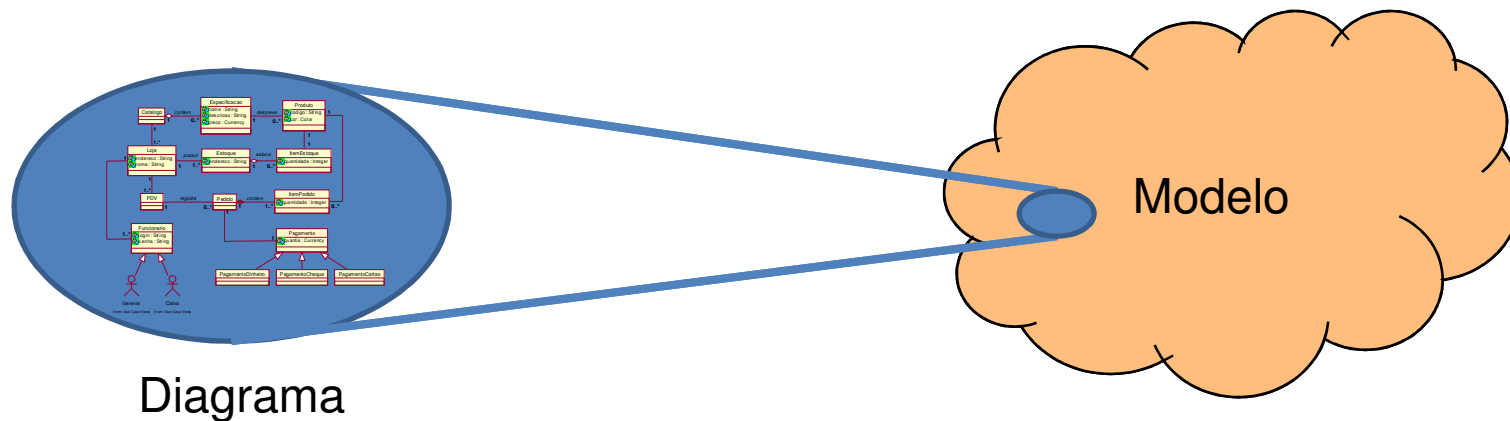
---

- Quais domínios do conhecimento utilizam modelos?
  - Todos!
- Modelos no domínio de desenvolvimento de software
  - Modelo também é software!
- Modelos servem para apoiar:
  - Derivação dos outros modelos
  - Codificação do sistema
- É preciso questionar a necessidade real de modelos que não servem para a derivação de outros modelos ou para a codificação do sistema!!!

# Modelos x Diagramas

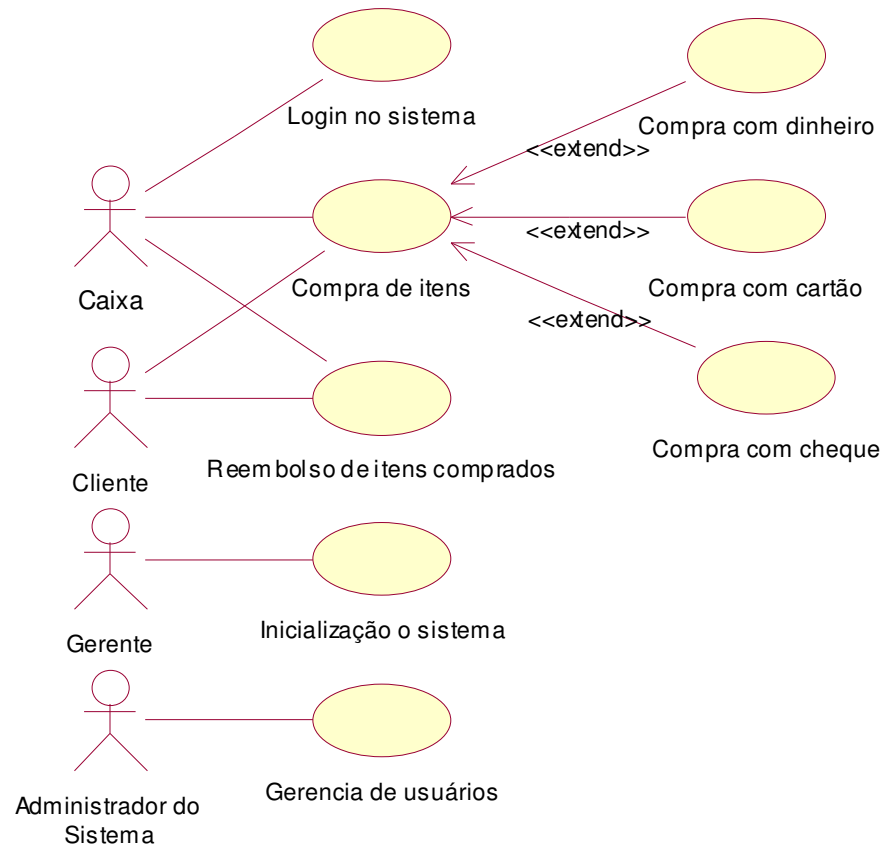
---

- O modelo contém toda a informação que representa o problema ou a solução
- O diagrama é uma visualização de parte de um modelo sob uma perspectiva
- Ou seja:
  - Se está no diagrama, está no modelo
  - Se não está no diagrama, não podemos concluir nada



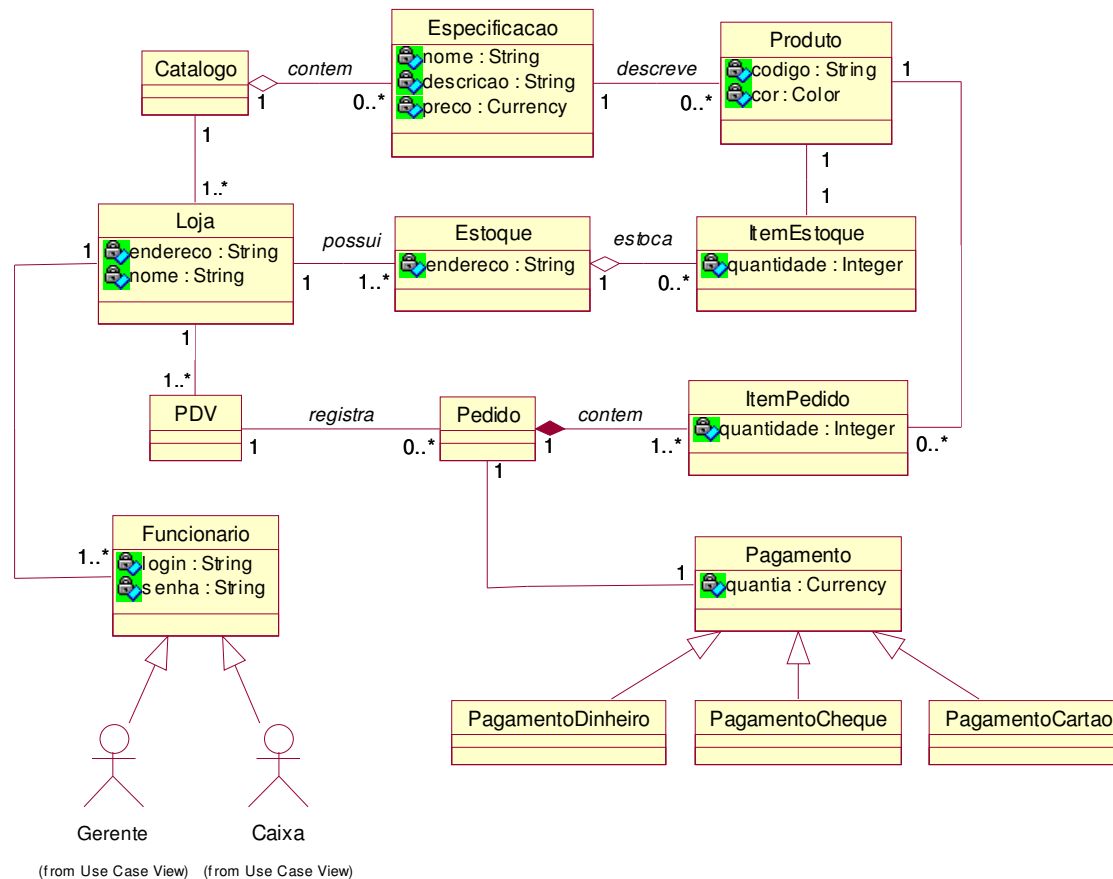
# Exemplo de modelos

## ➤ Diagrama de casos de uso



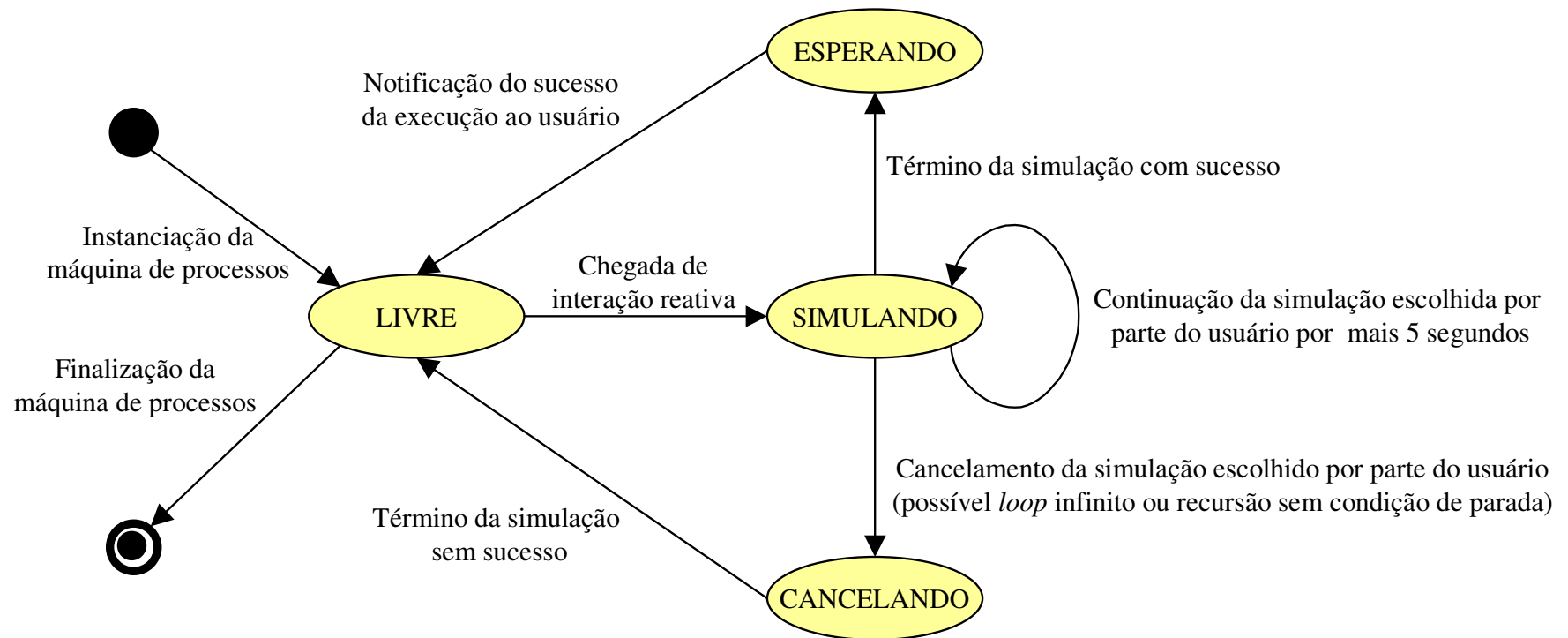
# Exemplo de modelos

## ➤ Diagrama de classes



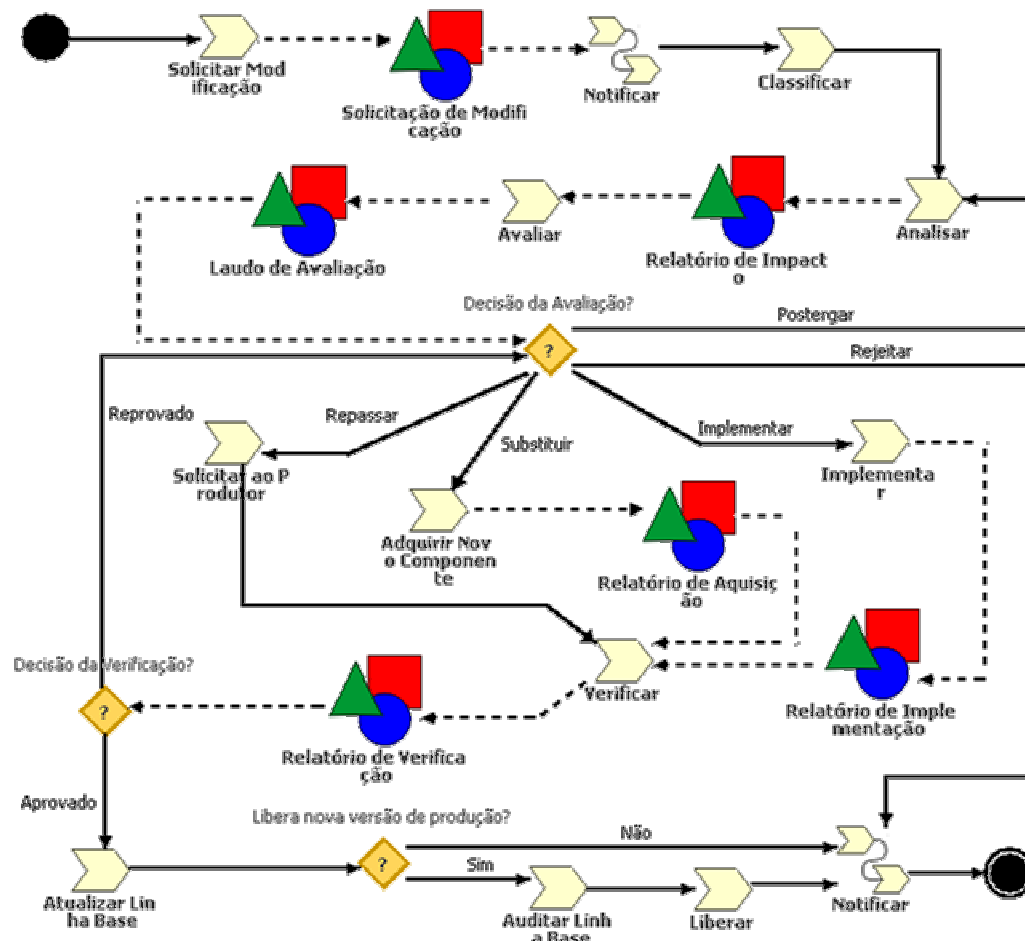
# Exemplos de modelos

## ➤ Diagrama de transição de estados



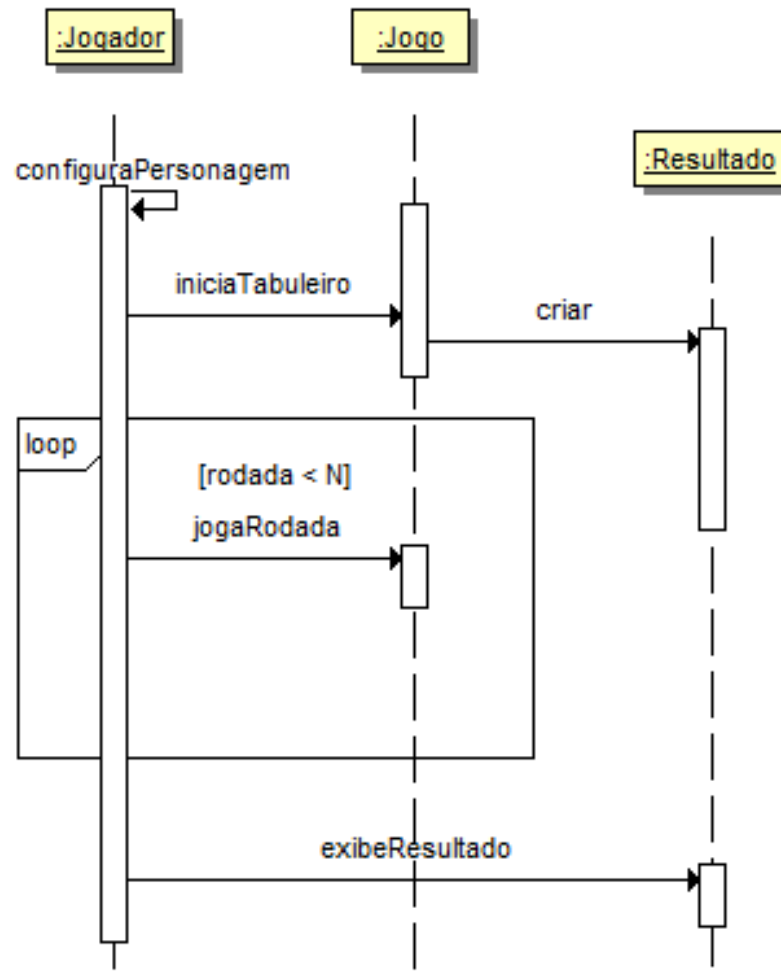
# Exemplo de modelos

- Diagrama de atividades – SPEM (Software and System Process Engineering Meta-Model)



# Exemplo de modelos

## ➤ Diagrama de seqüência





# Bibliografia

---

- Craig Larman, 2007, “Utilizando UML e Padrões”, 3ª ed.
- Várias transparências foram produzidas por Leonardo Murta
  - <http://www.ic.uff.br/~leomurta>