

# Desenvolvimento de Sistemas Software 2015/16

- **Escolaridade**
  - 2T + 2PL
  - 5 ECTS ~ 140 horas (aulas representam 60h!)
- **Equipa Docente**
  - José Creissac Campos ([jose.campos@di.uminho.pt](mailto:jose.campos@di.uminho.pt)) – T / PL
    - horário de atendimento: 6as 14h00-16h00 (marcação prévia!)
  - António N. Ribeiro ([anr@di.uminho.pt](mailto:anr@di.uminho.pt)) – PL
- **Canais de comunicação**
  - Aulas teóricas (**canal principal**)
  - Blackboard
  - Aulas PL (turno)

# Desenvolvimento de Sistemas Software

## • Avaliação

- Teste/Exame ( $\geq 9.0$ ) - uma prova individual, escrita, sobre a matéria leccionada
- Trabalho Prático ( $\geq 10.0$ ) - um projecto em grupo de análise e desenvolvimento de um sistema software
- Classificação Final ( $\geq 10.0$ )  
.6 Exame + .4 Trabalho

**IMPORTANTE:**  
Mínimos são condição necessária, mas  
não suficiente, para garantir aprovação  
à UC.

# Desenvolvimento de Sistemas Software

- **Regras de Funcionamento**
  - Aulas T
    - Apresentação e discussão da matéria
    - Sem registo regular de presenças
  - Aulas PL
    - Realização de exercícios pelos alunos  
(aplicação da matéria leccionada nas aulas T)
    - Registo de presenças
    - Reprovação por faltas (regras do Regulamento Académico)
  - Congelamentos de nota prática
    - Só notas **superiores** a 10 de 2015/16
    - Nota “congelada” sujeita a um tecto de 15 valores



# **DESENVOLVIMENTO DE SISTEMAS SOFTWARE**

# O que é um bom Sistema Software?

- **útil e utilizável** – bom software facilita a vida dos utilizadores responder às necessidades reais dos utilizadores
- **confiável** – sem *bugs*!

**Erro Browser**

Está a usar um browser não suportado.  
Por favor, efectue o pedido de certificado MULTICERT utilizando Internet Explorer ou Firefox.

**corrigir**

- **acessível**
- **rápido**

**✗ bug**

**Formulário de candidatura**  
Application form

**Terminar sessão**  
Close session

**Sumário do projecto**  
Project summary

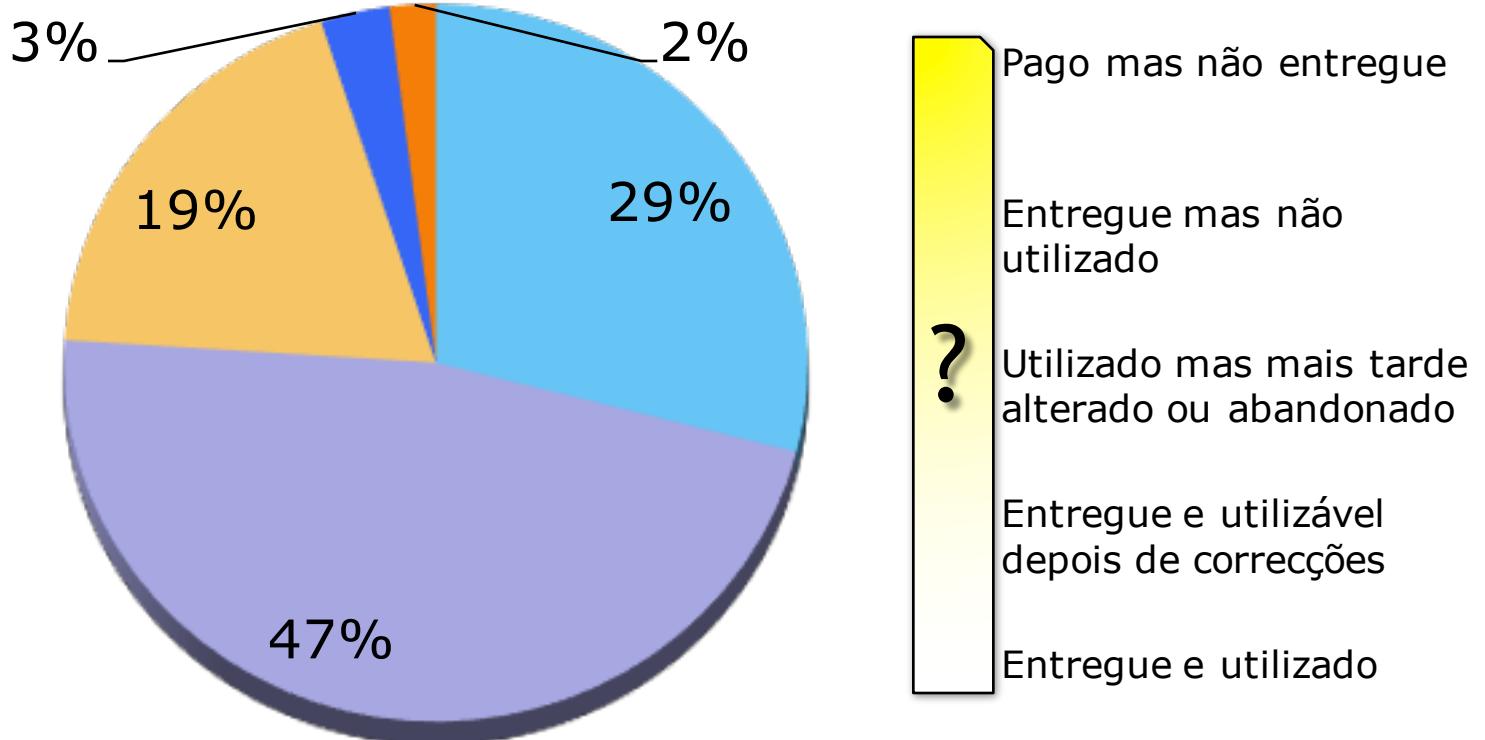
ADODB.Command error '800a0e2e'  
Application uses a value of the type 'Date/Time' in a field with a data type of 'Text'. The value must be enclosed in quotes.  
/projectos/cache/69/cache.area



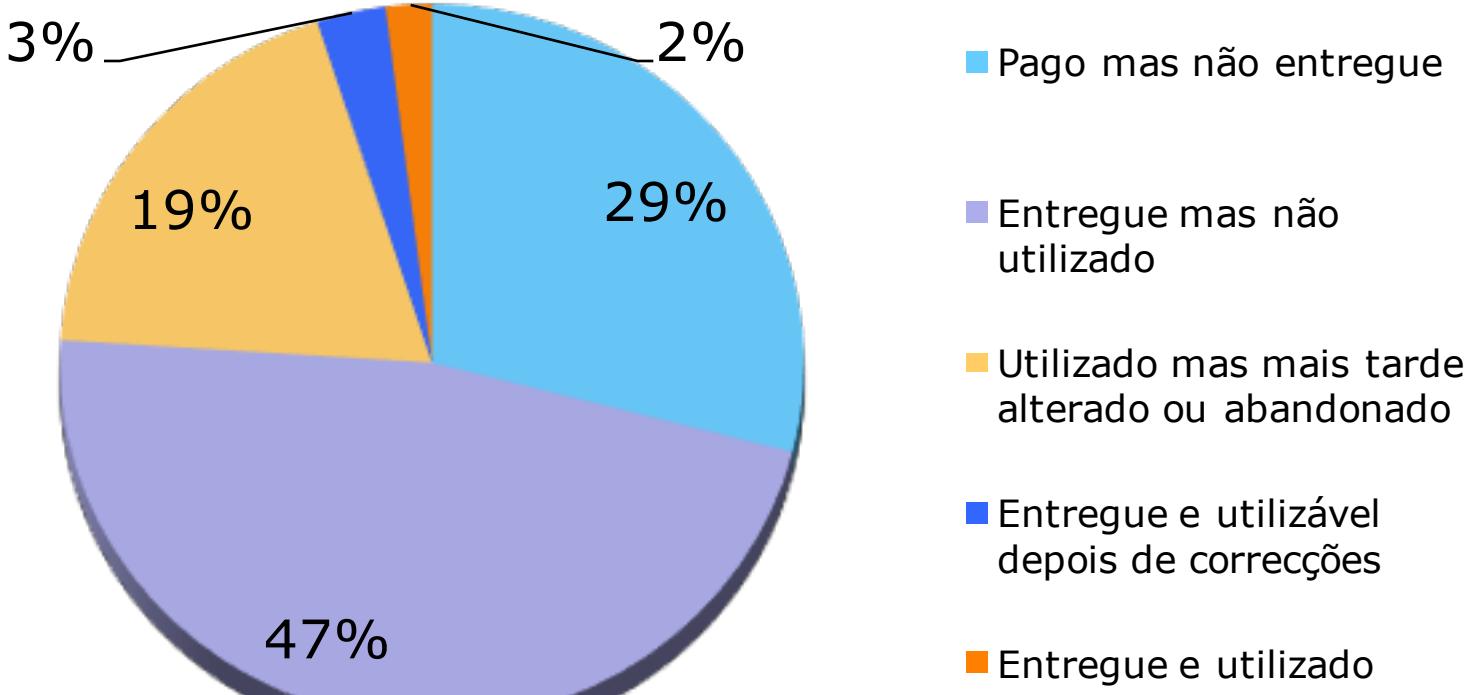
# O que é um bom Sistema Software?

- **Aquele que satisfaz as necessidades dos seus utilizadores:**
  - **útil e utilizável** – bom software facilita a vida dos utilizadores; deve responder às necessidades reais dos utilizadores; Análise de requisitos!
  - **confiável** – sem *bugs*!
  - **disponível** – se não está disponível nada mais interessa! está disponível a tempo e horas? está disponível na plataforma tecnológica pretendida?
  - **flexível** – as necessidades dos utilizadores mudam, os *bugs* têm que ser corrigidos
    - ✗ *bug* do ano 2k veio mostrar a falta de flexibilidade de muitos sistemas;
  - **acessível (financeiramente)** – quer na compra quer na manutenção; fácil e rápido de desenvolver;

# Como vai o desenvolvimento de Software?



# Como vai o desenvolvimento de Software?



- Mais de 75% do software pago não chegou a ser utilizado!
- Apenas 5% do software pago foi utilizado continuadamente (deste, 3% necessitou de correcções).

Fonte: US GAO, 1992

# Como vai o desenvolvimento de Software?

**Inquérito realizado em 1994 a 352 companhias (Standish Group):**

- 31% de todos os processos de desenvolvimento de software são cancelados antes de estarem terminados.
- 53% dos projectos custam 189% do estimado.
- 9% dos projectos de grandes companhias respeitam os prazos e o orçamento.
- 16% dos projectos de pequenas companhias respeitam os prazos e o orçamento.
- 56% de todos os *bugs* pode ser atribuídos a erros cometidos durante a fase de análise (i.e., não se esteve a construir o sistema certo!)

**Mais alguns dados sobre grandes projectos (>50,000 linhas de código):**

- produtividade média está abaixo das 10 linhas de código por dia;
- em média, encontram-se 60 erros por cada 10,000 linhas de código.

# Como vai o desenvolvimento de Software?

## Em conclusão:

- Problemas com o desenvolvimento de software:
  - Atrasos na entrega.
  - Incumprimento dos orçamentos.
  - Falha na identificação e satisfação das necessidades dos clientes.
  - Produtos entregues com falhas.

# Exemplos

Alguns exemplos de sistemas com problemas atribuíveis ao software:

- **Sonda Mariner I, Julho de 1962**

Deveria ter voado até Vénus. Apenas quatro minutos após o lançamento despenhou-se no mar. Descobriu-se depois que um operador de negação lógica tinha sido omitido por acidente no código do programa responsável por controlar os foguetes...

- **Therac-25, finais dos anos 80**

Máquina de Raios-X totalmente controlado por software. Diversos problemas provocaram a administração de radiação excessiva a vários doentes.

- **Aeroporto Internacional de Denver, início dos anos 90**

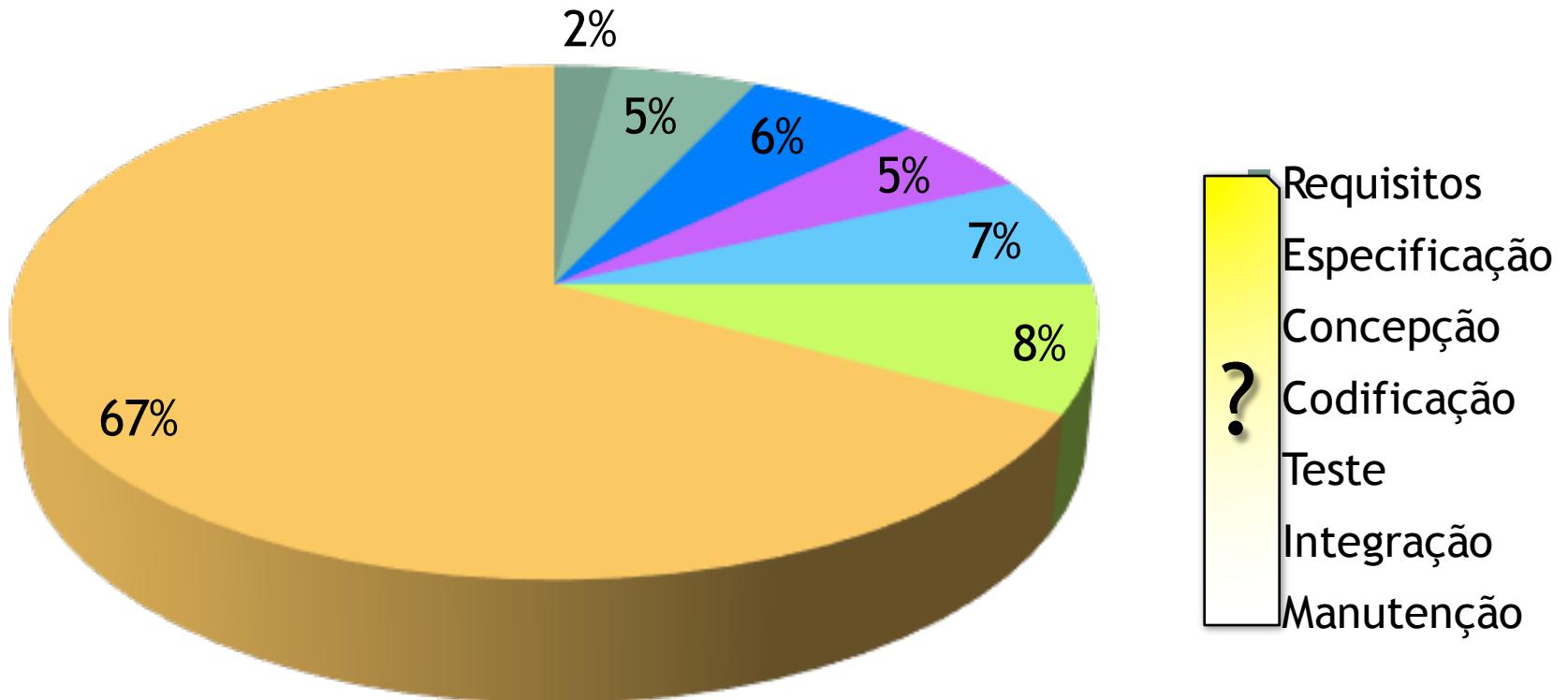
Sistema de tratamento de bagagem envolvendo mais de 300 computadores. O projecto excedeu os prazos de tal forma que obrigou ao adiamento da abertura do aeroporto (16 meses). Foi necessário mais 50% do orçamento inicial para o pôr a funcionar.

- **Colocação de professores, inícios sec. XXI**

Empresa inicialmente contratada não consegui desenvolver uma versão funcional do sistema e foi necessário contratar uma nova empresa.

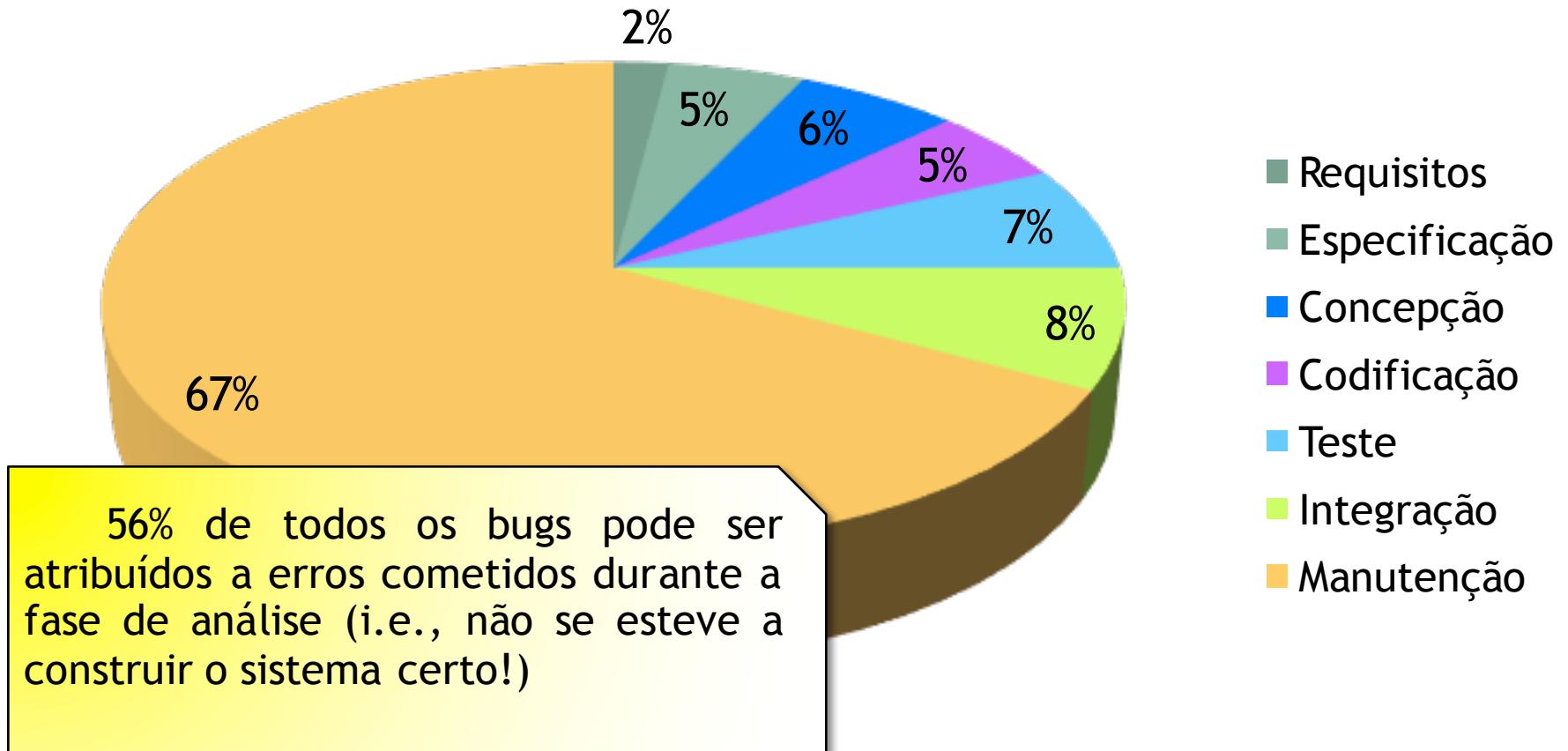
# Como vai o desenvolvimento de Software?

## Repartição de custos dos projectos



# Como vai o desenvolvimento de Software?

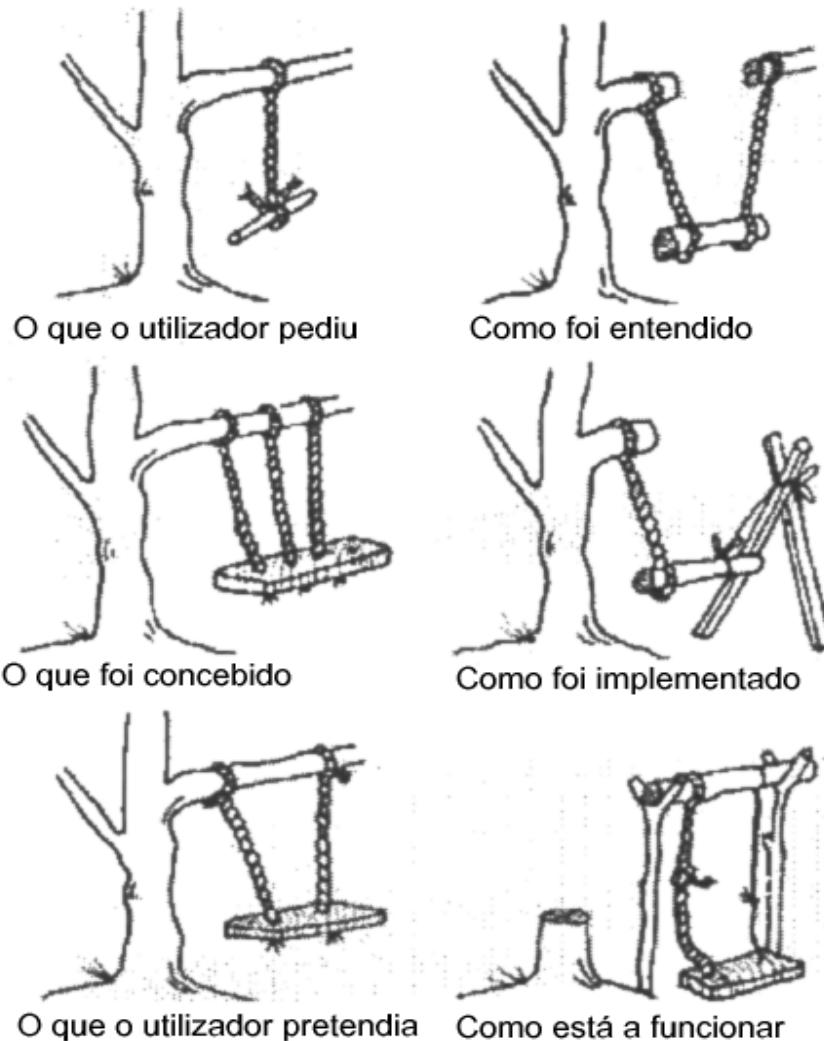
## Repartição de custos dos projectos



Fonte: Stephen R. Schach. *Object-Oriented and Classical Software Engineering*, 5<sup>th</sup> ed. McGraw-Hill. 2002

# Desenvolvimento de Software (Riscos)

Desenvolver um bom sistema não é tarefa trivial



- Riscos associados aos requisitos.
- Riscos tecnológicos.
- Riscos de competência.
- Riscos políticos.

# Desenvolvimento de Software (Riscos) (II)

## Riscos associados aos requisitos

É necessário comunicar com os *stakeholders* para:

- compreender que tarefas o sistema deve suportar;
- compreender como o sistema encaixa nas restantes actividades dos *stakeholders*.

Um dos maiores desafios é construir o sistema certo.

## Riscos Tecnológicos

- Qual a tecnologia mais apropriada?
- Como combinar diferentes tecnologias?

É necessário validar as soluções tecnológicas o mais cedo possível.

Muito relevante no actual contexto (cf. tecnologias Web - Facebook).

# Desenvolvimento de Software (Riscos) (II)

## Riscos de Competência

- É necessário saber-se o que se está a fazer (obviamente?).
- É necessário ter os profissionais adequados.

Exemplo de OO: fácil de aprender/mais difícil de aproveitar ao máximo.

## Riscos Políticos

- Por muito bom que seja o SI só terá sucesso se tiver o apoio das *pessoas certas*.

# Respostas I - Tecnologia

- Primeiras abordagens à Crise do Software preocupavam-se mais com a produtividade do que com a qualidade.
- As tecnologias de programação têm vindo a tornar-se cada vez mais sofisticadas (tanto aos níveis dos paradigmas como das ferramentas).

## **Paradigmas de Programação**

O modo como estruturamos o código tem vindo a evoluir como resposta ao aumento da complexidade do software:

- Programação estruturada (70's) - Estruturar o código para controlar complexidade.
- Programação modular - Estruturar o código, mas também os dados.
- Programação orientada aos objectos (80's) - Aumenta o poder expressivo na estruturação de dados/código.

## **Ferramentas**

Ambientes de Desenvolvimento Integrado (IDEs) cada vez mais sofisticados procuram facilitar a tarefa de programação (e também de análise).

## Respostas II - Métodos de Desenvolvimento

- A tecnologia só por si não resolve os problemas (e estes têm vindo a aumentar!)
- A tecnologia é apenas uma ferramenta, é necessário saber como utilizá-la
- Hoje em dia a *Crise do Software* tem muito a ver com a qualidade do produto final.
- São necessários métodos de desenvolvimento que garantam produtividade e qualidade.

### método

do Lat. *methodu* < Gr. *méthodos*, caminho para chegar a um fim

s. m., processo racional que se segue para chegar a um fim; modo ordenado de proceder; processo; ordem; conjunto de procedimentos técnicos e científicos;

(<http://www.priberam.pt/dlpo/>)

O desenvolvimento de software não pode ser encarado como *arte*,  
mas como **Engenharia**.

Necessitamos de métodos e ferramentas apropriados.  
Em DSS apresenta-se uma proposta, existem outras!

# Aulas Teóricas - Programa

- O Processo de Desenvolvimento de Software – diferentes abordagens.
- Modelação de Sistemas Software em UML:
  - Visão geral – os diferentes níveis de modelação
  - Modelação comportamental:
    - Diagramas de Use Case;
    - Diagramas de Interacção (Sequência/Colaboração);
    - Diagramas de Estado (Statecharts);
    - Diagramas de Actividade;
  - Modelação estrutural:
    - Diagramas de Classe (revisão de conceitos OO);
    - Diagramas de Package; Diagramas de Instalação (Deployment).
- Uma abordagem ao desenvolvimento baseada em UML
- Mapeamento de objectos no modelo relacional

# Práticas Laboratoriais - Programa

- Apresentação da Ferramenta de Modelação:
  - modelação em UML;
  - geração de código.
- Estudos de caso:
  - pequenos exemplos para apreensão dos conceitos;
  - realização do trabalho.

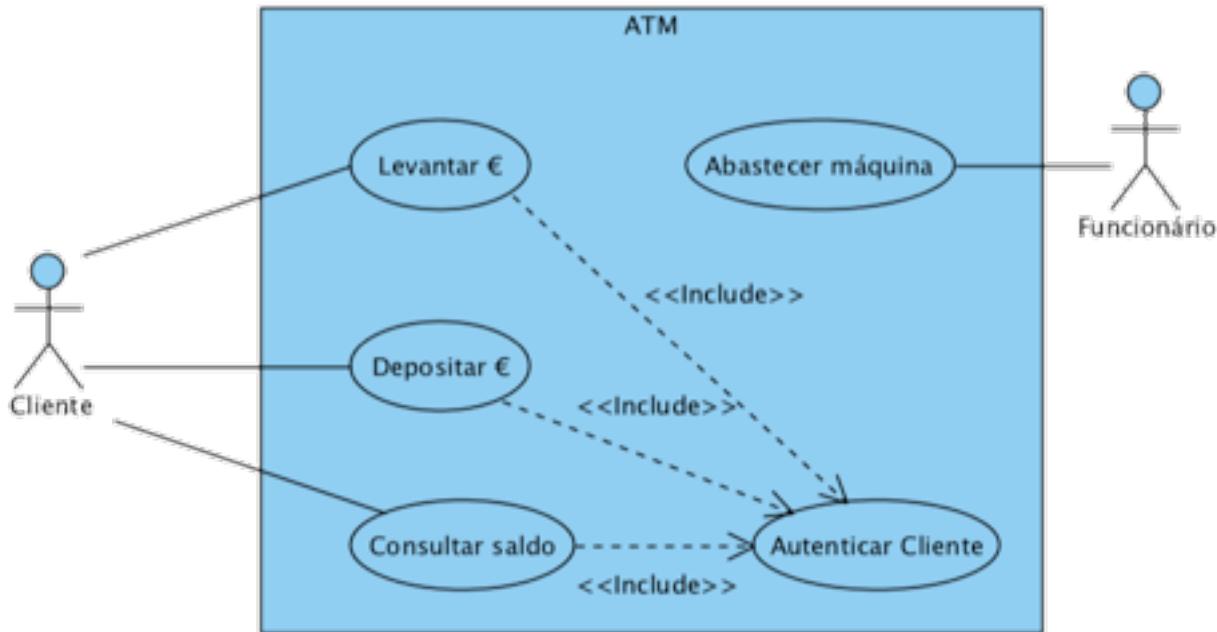
# UML - Unified Modelling Language

(Booch, Jacobson & Rumbaugh)

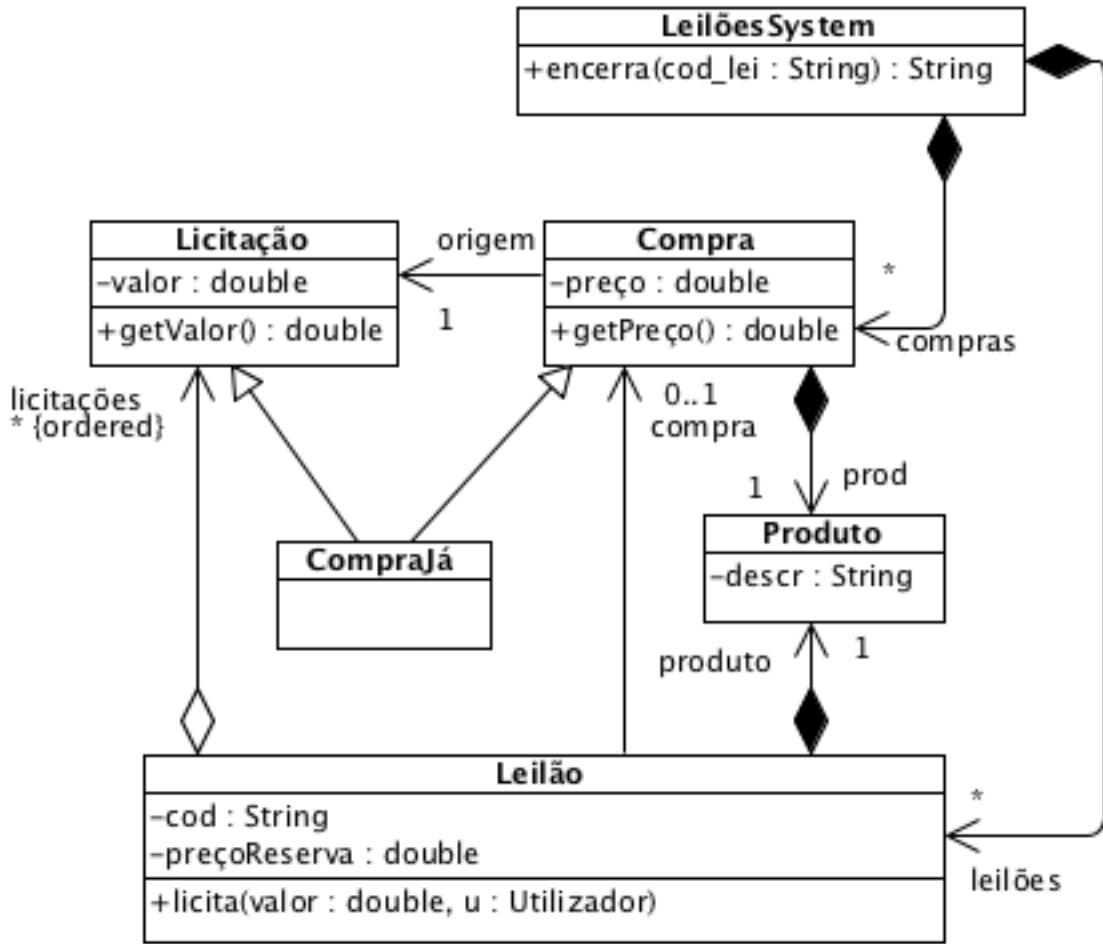
- A UML é uma **família de linguagens gráficas de modelação**
  - inclui (13) diagramas para as diferentes fases de desenvolvimento
- A UML foi pensada para o **desenvolvimento orientado aos objectos**, mas é independente das linguagens de programação a utilizar
  - permite explorar o paradigma OO
- A UML possibilita trabalhar a diferentes níveis de abstracção
  - facilita **comunicação e análise**
- A UML **NÃO É** um processo de desenvolvimento de software, mas pode ser utilizado com diferentes processos
- A UML é uma norma mantida pelo OMG (Object Management Group)
- pode também dizer-se que é a norma *de facto* da indústria de software
- A UML é **suportada por ferramentas** (50+)
  - *Rational Rose (IBM), Together (Borland), Visual Paradigm, Poseidon, etc.*

# Alguns Exemplos de Diagramas UML

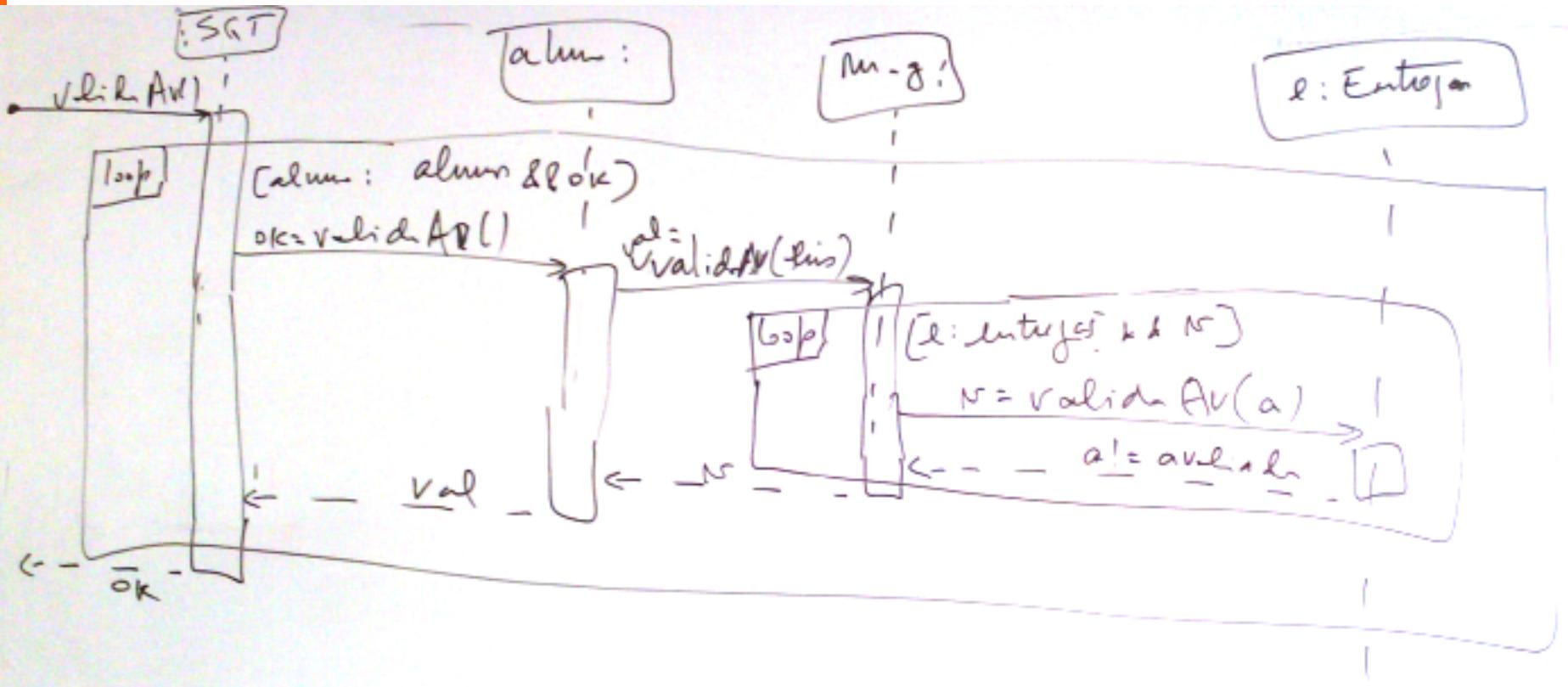
## Diagrama de Use Case



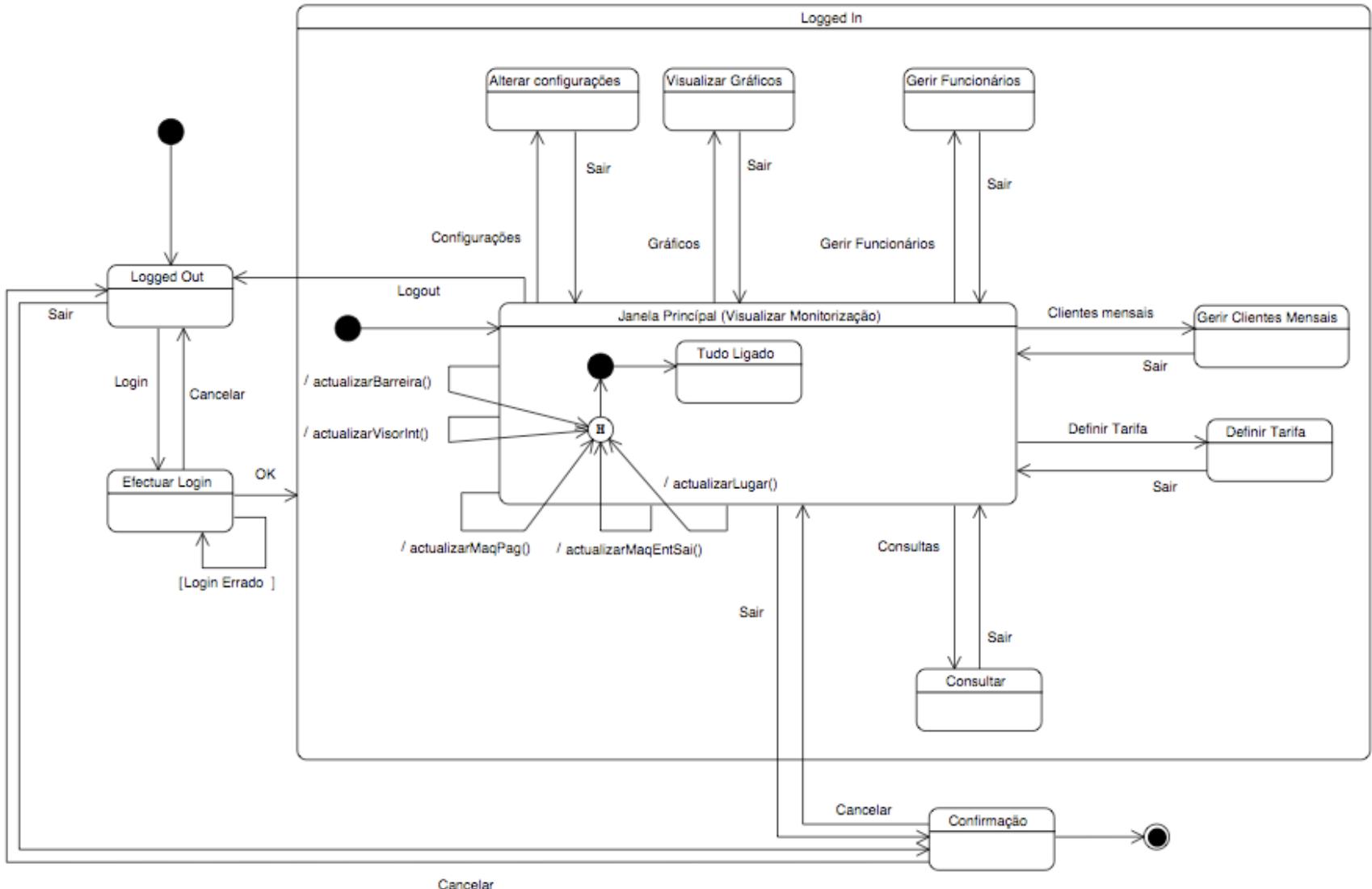
## Diagrama de Classe



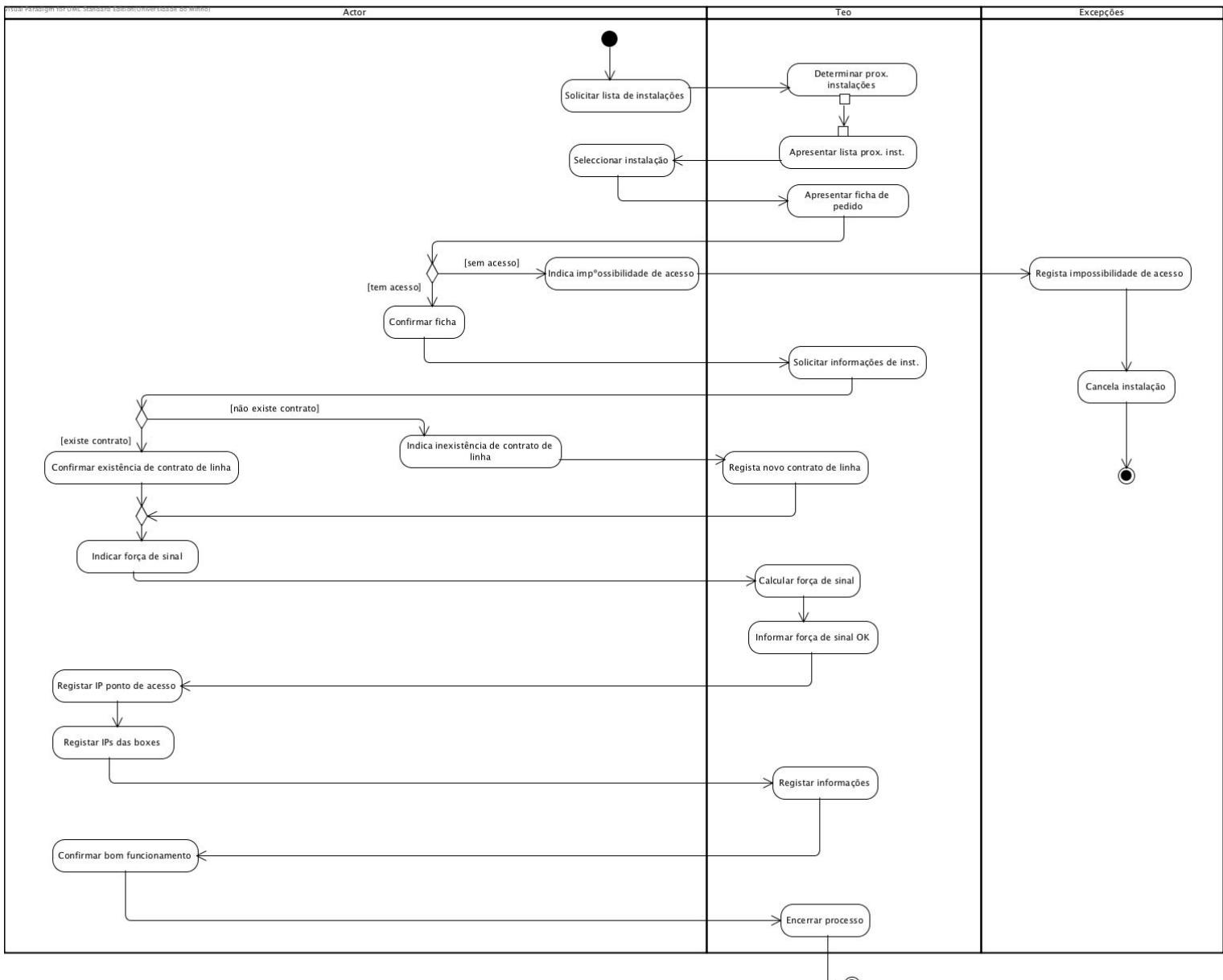
## UML - Diagrama de Sequência



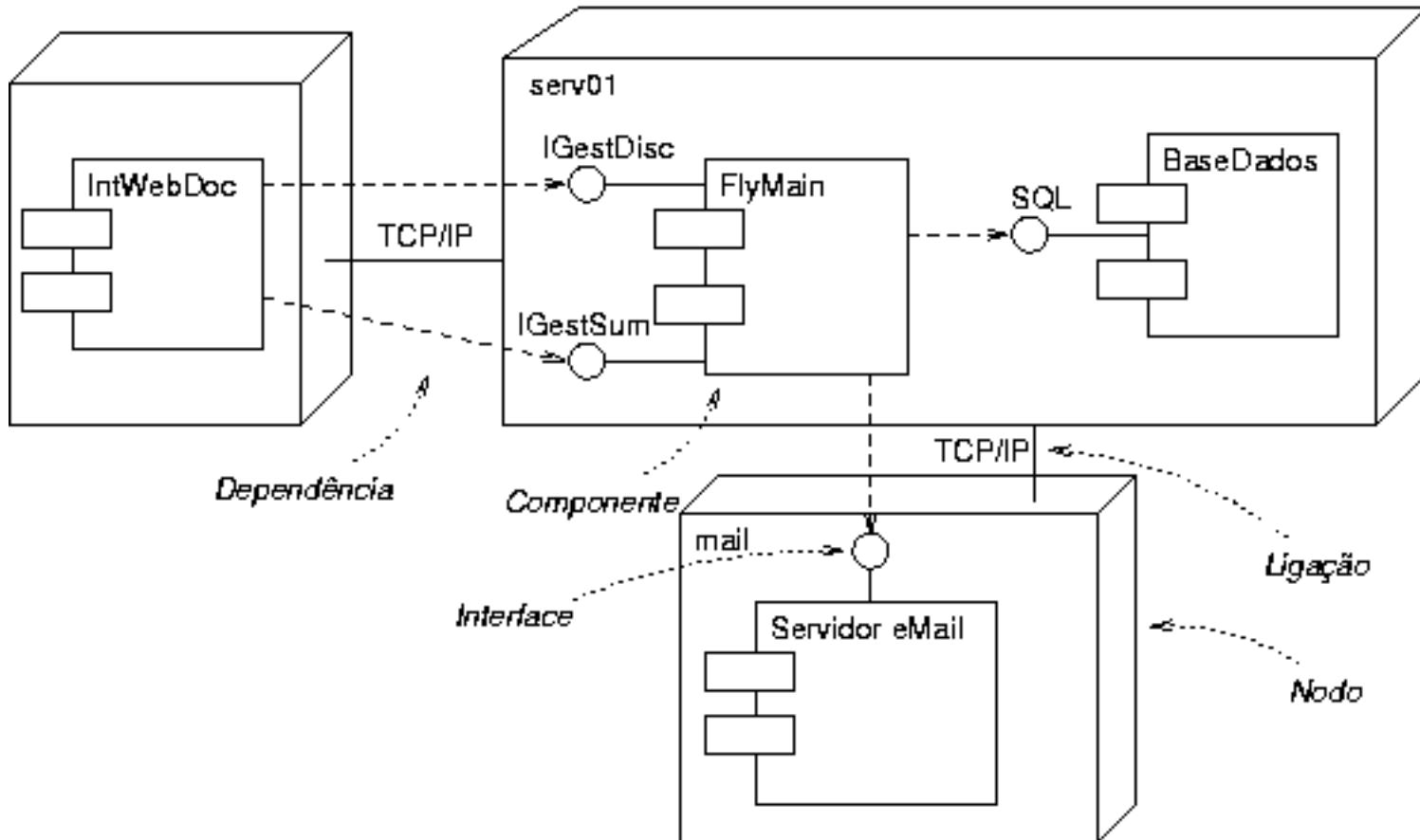
## UML - Diagrama de Estado (Statechart)



# UML - Diagrama de Actividade



## UML - Diagrama de Instalação



# Bibliografia

- J. Arlow, I. Neustadt. *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design (2nd edition)*. Addison-Wesley Professional, 2005.
- D. Pilone, N. Pitman. *UML 2.0 in a Nutshell (2nd edition)*. O'Reilly Media, 2005.
- Martin Fowler. *UML Distilled (third edition)*. Addison-Wesley, 2004.  
(bom livro!)
- Scott W. Wembler, *The Elements of UML 2.0 Style*, Cambridge University Press, 2005.
- R. Pressman. *Engenharia de Software*, 6th. Ed., McGraw Hill, 2005.

Em português:

- M. Nunes & H. O'Neill. *Fundamental do UML, 3<sup>a</sup> edição*. FCA. 2007.
- Apontamentos de suporte às aulas teóricas  
(irão sendo disponibilizados ao longo do semestre).

# Trabalho Prático

- Grupos de 3-5 elementos
  - Aumentar a capacidade de trabalho
  - Fomentar a discussão de soluções alternativas
  - Assumir vários papéis na equipa
- Enunciado será apresentado dentro de uma semana.
- A realizar **durante** o semestre - três momentos relevantes:
  1. Inscrição dos grupos com entrega intermédia:
    - relatório de análise de requisitos + API da camada de negócio
    - entregue até às 24h00 de 14 de Novembro (~8 semanas) - **eliminatório**
  2. Entrega final: relatório, **modelação** e software produzidos
    - entregue até às 24h00 de 30 de Dezembro (~7 semanas) - para classificação
    - Critérios mínimos serão definidos
  3. Apresentação e discussão:
    - em Janeiro (altura a definir)

## Teste / Exame

- Individual
- Com consulta de apontamentos
  - Estritamente proíbida a utilização de dispositivos com capacidades de comunicação
- No exame aplicam-se as mesma regras de cálculo da nota que para o teste
- Datas
  - Teste: 5 de Janeiro
  - Exame de recurso: 25 de Janeiro

## A fazer...

- Inscrição nos turnos PL (hoje, sexta, às 14h30 no Bb)
  - Turnos de 30 alunos
  - Organizarem-se em grupos de 3 a 5 elementos

# Sumário

- Apresentação da UC
  - Avaliação
  - Trabalho
- Motivação
  - Alguns dados sobre desenvolvimento de software
  - Necessidade de um método de desenvolvimento
  - Necessidade de uma linguagem de modelação
- UML (apresentação)
- Bibliografia