Universidade do Minho
Escola de Engenharia
Departamento de Informática

Desenvolvimento de Sistemas Software

LEI – 3º ano / 1º semestre

2014/2015

António Nestor Ribeiro
anr@di.uminho.pt
(com contributos de J.C. Campos e F.M. Martins)

http://www.di.uminho.pt

Desenvolvimento de Sistemas Software

- **Escolaridade**
  - 2T + 2PL
  - 5 ECTS – 140 horas (ficamos com 4h / semana)
- **Equipa Docente**
  - António Nestor Ribeiro (anr@di.uminho.pt) – T/PL
  - Rui Couto (rui.couto@di.uminho.pt) - PL (previsivelmente!)
- **Canais de comunicação**
  - Aulas teóricas (**canal principal**)
  - Blackboard
  - Aulas PL (turno)

Desenvolvimento de Sistemas Software

- **Avaliação**
  - Exame (≥ 9.0) – uma prova escrita sobre a matéria leccionada
  - Trabalho Prático (≥ 10.0) – um projecto de análise e desenvolvimento de um sistema software
  - Classificação Final (≥ 10.0)

    .6 Exame + .4 Trabalho

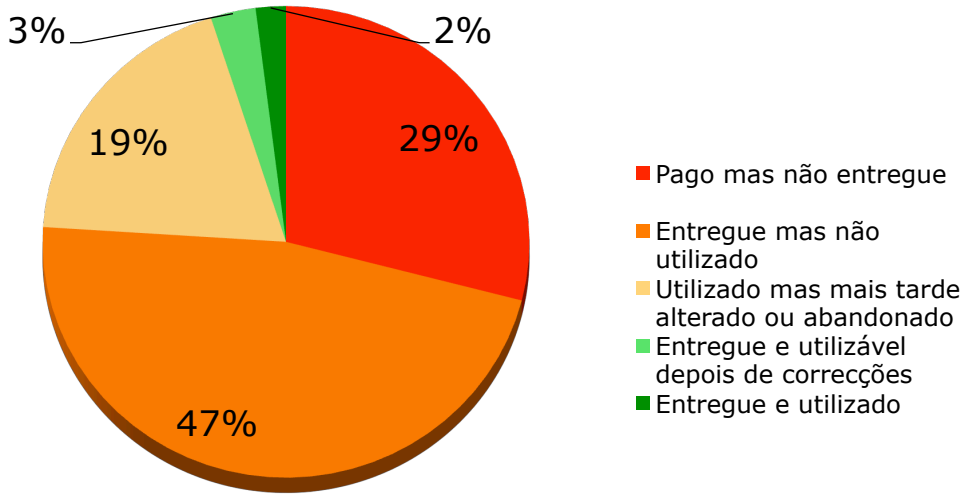
DESENVOLVIMENTO DE SISTEMAS SOFTWARE

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?



- 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: 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.

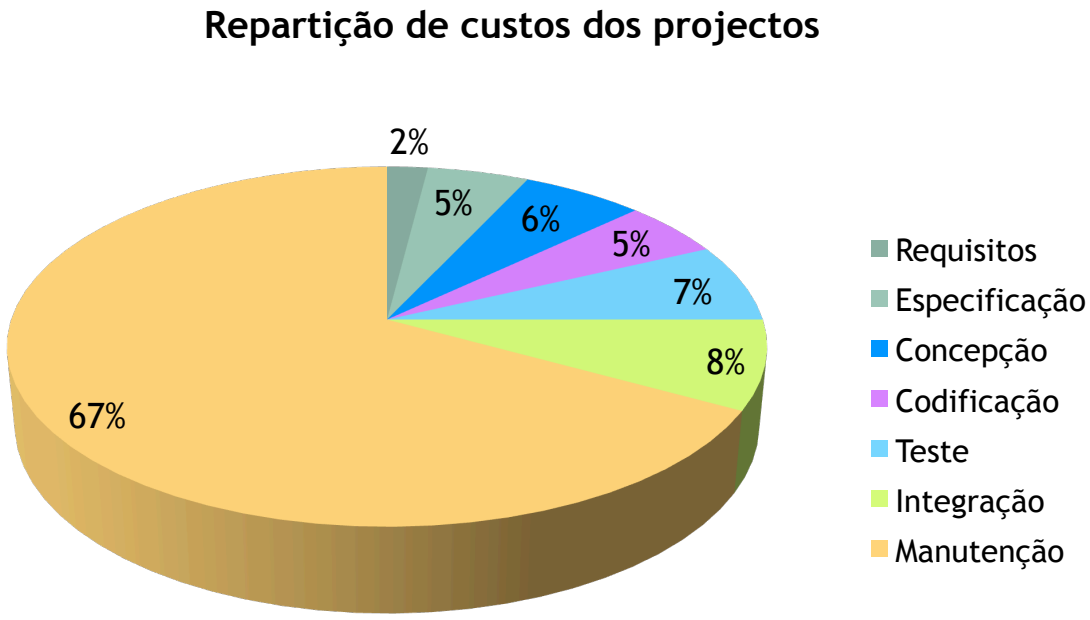
- Aeroporto Internacional de Denver, início dos anos 90

  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?



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

Como vai o desenvolvimento de Software?

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

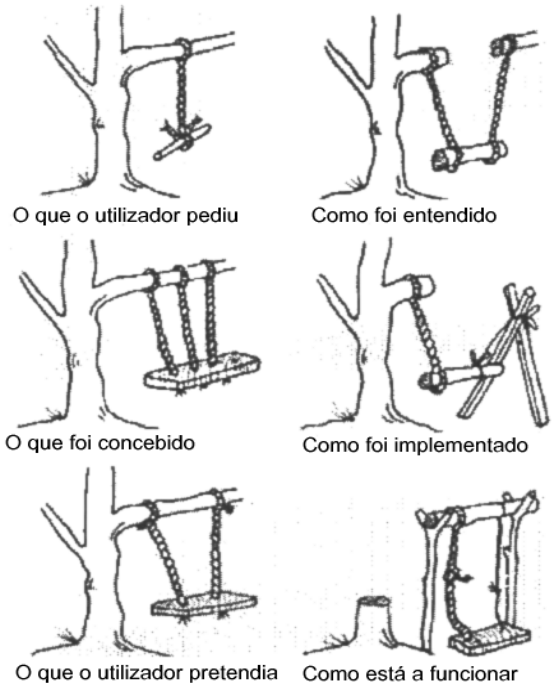
- 31% de todos os processos de desenvolvimento de software são cancelados antes de estarem terminados.
- 53% dos projectos custam 189% do estimado.
- 9% dos projectos de grandes companhias respeitam os prazos e o orçamento.
- 16% dos projectos de pequenas companhias respeitam os prazos e o orçamento.
- 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.

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 UML;
  - geração de código.
- Estudos de caso:
  - pequenos exemplos para apreensão dos conceitos;
  - realização do trabalho.

Trabalho Prático

- Grupos de 3-5 elementos
  - Aumentar a capacidade de trabalho
  - Fomentar a discussão de soluções alternativas
  - Assumir vários papeis na equipa
- A realizar em duas fases durante o semestre:
  - Fase 1: análise de requisitos
    - até 14 de Novembro (8 semanas) – 30% da nota
  - Fase 2: relatório final, **modelação** e o software produzidos
    - Até 10 de Janeiro 2015 (7 semanas) – 70% da nota
  - Apresentação e discussão:
    - … a definir, mais para o fim do semestre.

Desenvolvimento de Sistemas Software

- **Regras de Funcionamento**
  - Aulas PL
    - Registo de presenças
    - Reprovação por faltas (regras do Regulamento Académico)
  - Aulas T
    - Sem registos de presença
  - Congelamentos de nota prática
    - Só notas superiores a 10 de 2013/14
    - Nota “congelada” sujeita a um tecto de 13 valores

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

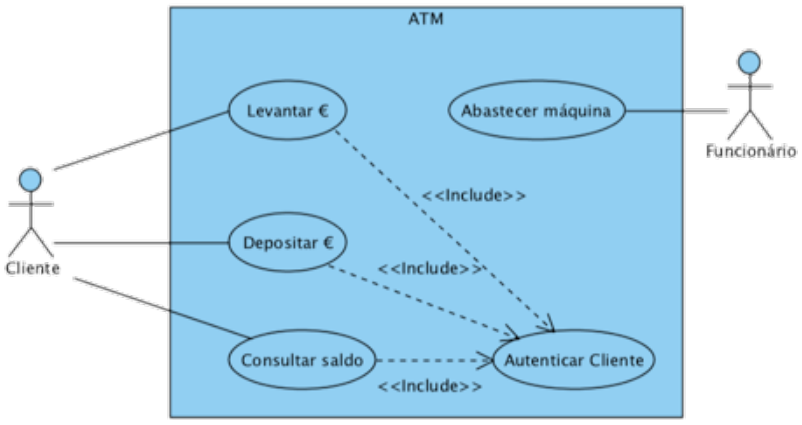
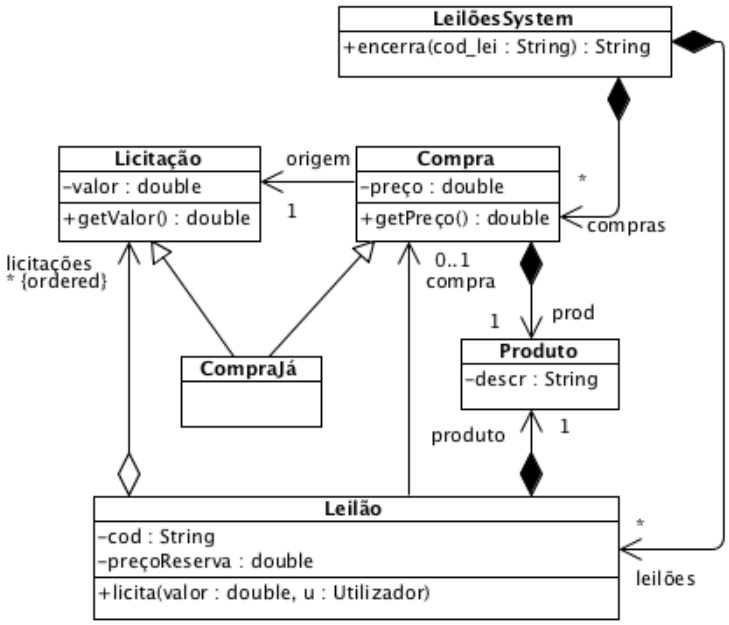
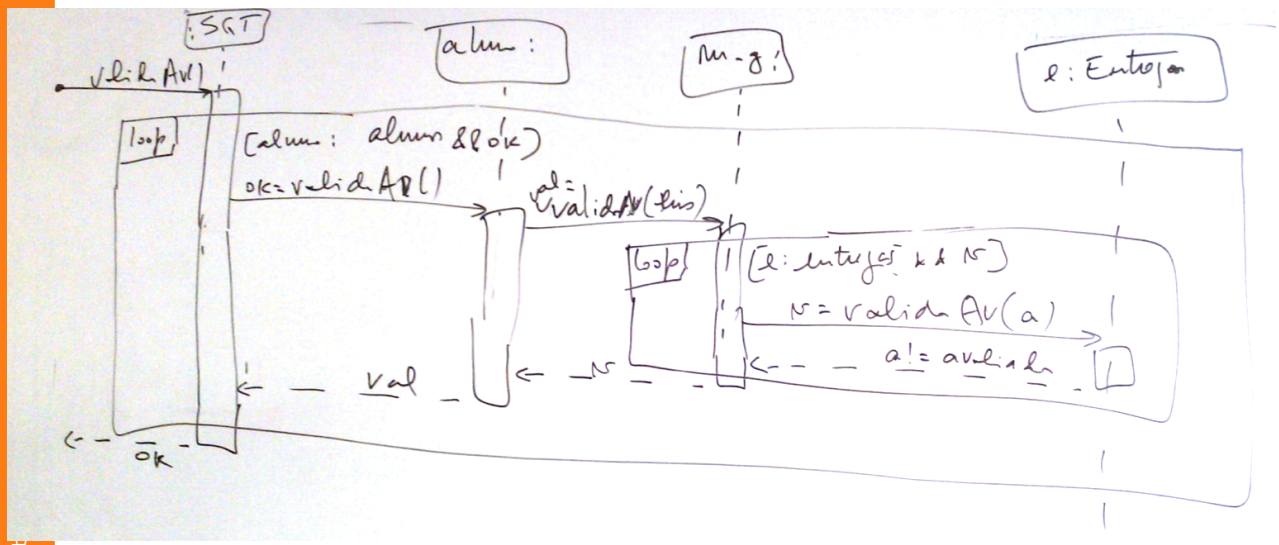


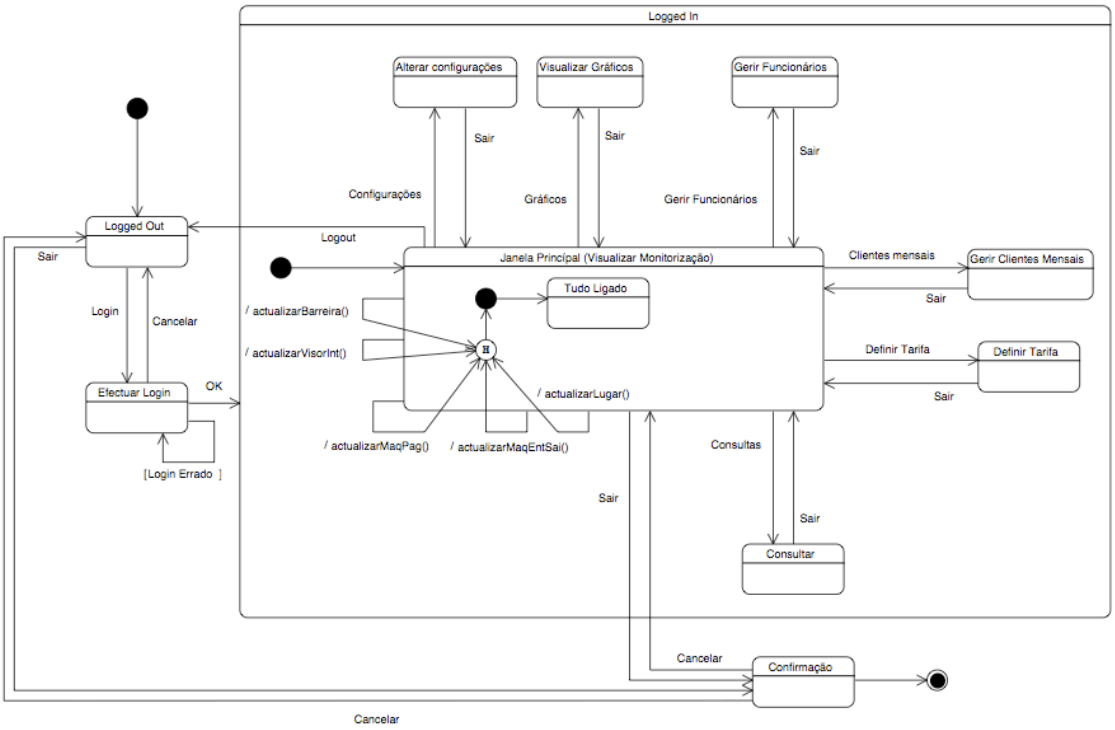
Diagrama de Classe



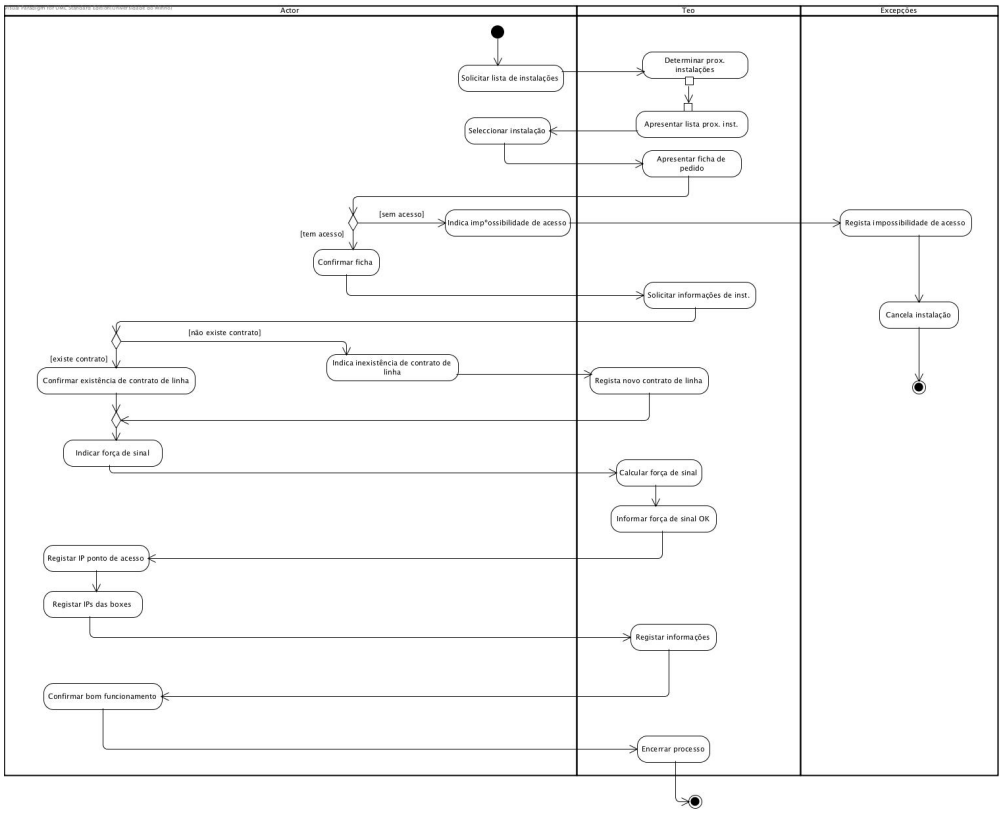
UML - Diagrama de Sequência

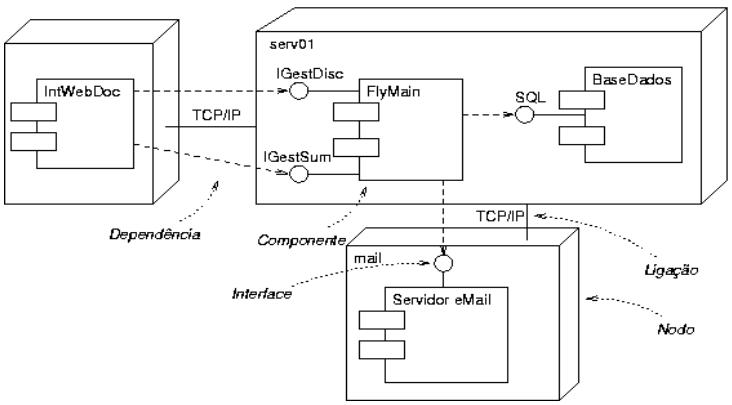


UML – Diagrama de Estado (*Statechart*)



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, 5ª edição*. FCA. 2007.
- Apontamentos de suporte às aulas teóricas
  (irão sendo disponibilizados ao longo do semestre).

A fazer…

- Inscrição nos turnos PL (próxima sexta)
  - Senha de pré-inscrição: dss201415
- Turnos disponíveis a partir das 12:45h de 17.09 (done!)
  - 3ª 09h-11h (**PL3**)
  - 3ª 11h-13h (**PL2**)
  - 3ª 14h-16h (**PL5**)
  - 6ª 09h-11h (**PL1**)
  - 6ª 14:30h-16:30h (**PL4**)
- Organizarem-se em grupos de 3 a 5 elementos
  - Inscrição será feita no BB dentro de duas semanas.

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
- Bibliografia