Introdução aos Sistemas de Informação

Sumário:

- Definição de Sistema de Informação.
- Metodologias de Desenvolvimento.
- Processos de Desenvolvimento.
- Introdução ao UML.

O que é um Sistema de Informação?

Sistema. *s. m.* Reunião de partes ligadas entre si, formando uma estrutura complexa. Conjunto de meios, processos destinados a produzir um resultado = Método.
Informação. *s. f.* Conjunto de conhecimentos reunidos sobre um determinado assunto; documentação.
Dicionário de Língua Portuguesa Contemporânea. Academia de Ciências de Lisboa.

Sistema de Informação:

Fornece um meio para reunir informação de uma dada organização, fornecendo procedimentos para registo e disponibilização dessa informação, com o objectivo de auxiliar a organização nas suas actividades.

- Não é necessariamente um sistema software;
- Existem vários tipos de sistemas de informação...

Uma taxonomia para Sistemas de Informação

Diferentes tipos de Sistemas de Informação (SI) segundo Alter:

- Sistemas de Automação de Escritórios
- Sistemas de Comunicação
- Sistemas de Processamento de Transacções
- Sistemas de Informação para a Gestão
- Sistemas de Informação para Executivos
- Sistemas de Suporte à Decisão
- Sistemas de Execução
- Sistemas *Groupware*

Atenção: Taxonomias são sempre redutoras!

Esta taxonomia representa uma visão funcional dos SI.

- Visão funcional — meio tecnológico para registar, guardar e disseminar informação.
- Visão estrutural — colleacção de pessoas, processos, dados, modelos tecnologia e linguagens.

Uma taxonomia para Sistemas de Informação (II)

Sistemas de Automação de Escritórios

- Fornecem meios para o processamento de dados ao nível do indivíduo;
- Incluem o processamento de texto, folhas de cálculo, etc.

Sistemas de Comunicação

- Fornecem meios para a partilha de informação;
- Incluem teleconferência, correio electrónico, correio de voz, fax, etc.

Sistemas de Processamento de Transacções

- Recolhem e armazenam informação acerca de transacções (eventos de interesse);
- Controlam alguns aspectos relativos às transacções.

Sistemas de Informação para a Gestão

- Analizam a informação produzida pelos Sistemas de Informação Transaccionais;
- Convertem informaçao sobre transacções em informação de controlo de desempenho e gestão da organização.

Uma taxonomia para Sistemas de Informação (III)

Sistemas de Informação para Executivos

- Evolução dos Sistemas de Informação para a Gestão vocacionada para executivos;
- Permitem análise da informação de forma simples e interactiva e a diferentes níveis de detalhe.

Sistemas de Suporte à Decisão

- Auxiliam a tomada de decisões fornecendo informação, modelos ou ferramentas de análise.

Sistemas de Execução

- Suportam o processo productivo da organização.

Sistemas *Groupware*

- Suportam o trabalho em equipa;
- Fornecem meios de partilha de informação, controlo de trbalho conjunto, etc..

Uma taxonomia para Sistemas de Informação (IV)

Sejam de que tipo forem, os SI:

- existem para auxiliar a organização
  - ➧ mais um meio a juntar a tantos outros;
- devem ser concebidos em função das necessidades da organização
  - ➧ vão ser utilizados pela organização, não por quem os concebeu.

É necessário perceber a organização para conceber um bom SI.

- Quais as actividades da organização a suportar.
- Qual a informação relevante que flui na organização.
- Quais as tarefas das pessoas da organização.

Entender o problema antes de desenvolver a solução!

O que é um bom Sistema (de Informação)

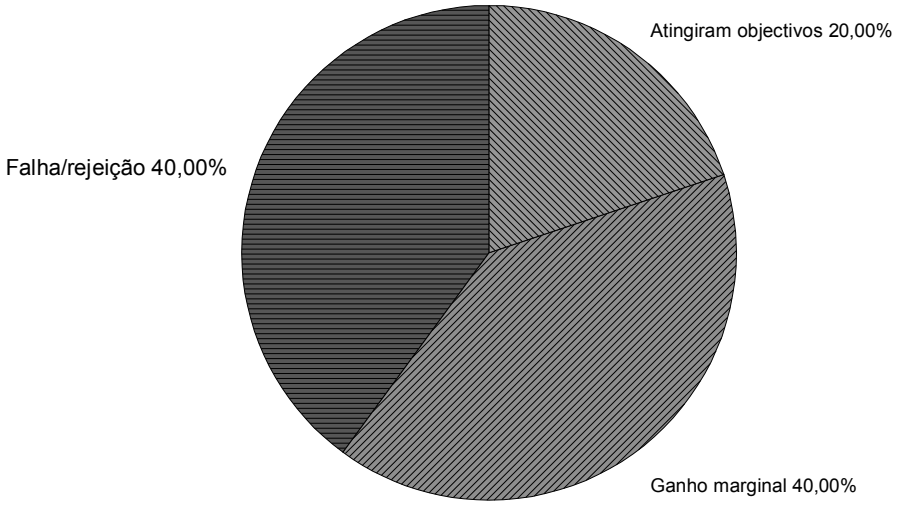
Aquele que satisfaz as necessidades dos seus utilizadores:

- **útil e utilizável** — bom software facilita a vida dos utilizadores ⇨ deve responder às necessidades reais dos utilizadores;
- **confiável** — sem *bugs*!
- **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** (financieramente) — quer na compra quer na manutenção ⇨ fácil e rápido de desenvolver;
- **disponível** — se não está disponível nada mais interessa! ⇨está disponível a tempo e horas? está disponível na plataforma tecnológica pretendida?

Como vai o desenvolvimento de software?

Estatísticas sobre desenvolvimento de Software

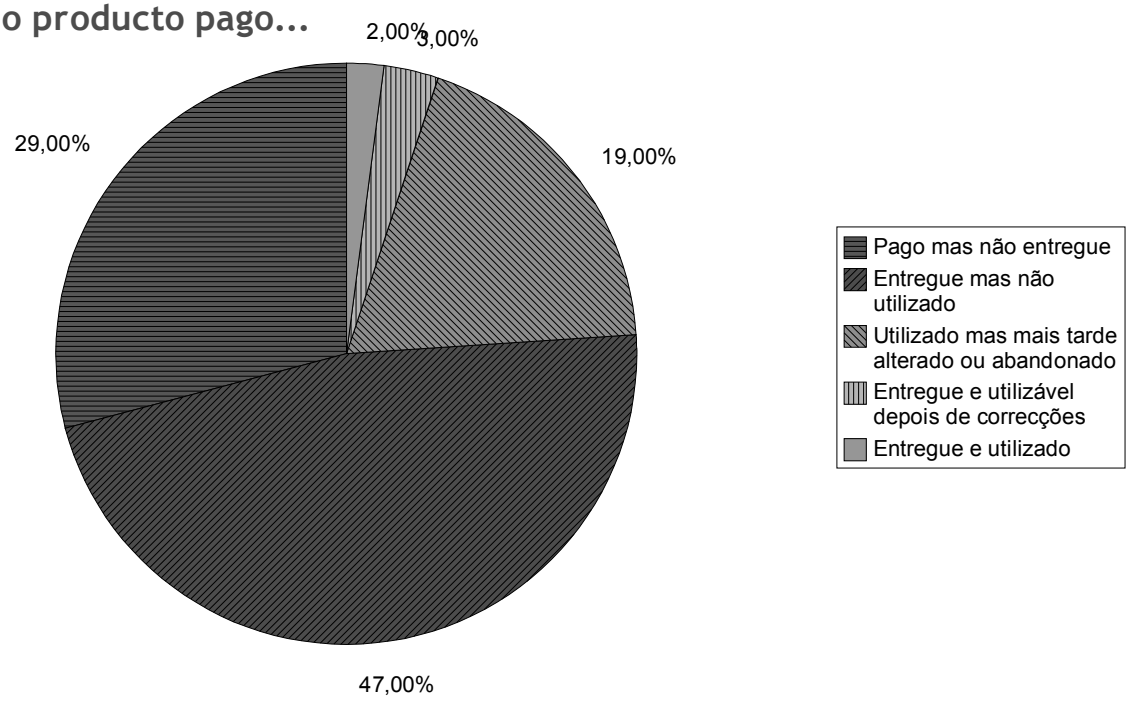
Sobre o producto entregue...



Fonte: Eason, 1988

Estatísticas sobre desenvolvimento de Software (II)

Sobre o producto pago...



- 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

Estatísticas sobre desenvolvimento de Software (III)

Sobre o processo...

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

- 56% de todos os *bugs* pode ser atribuido a erros cometidos durante a fase de análise (i.e., não se esteve a contruir o sistema certo!)
- 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.

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

- productividade média está abaixo das 10 linhas de código por dia;
- em média, encontram-se 60 erros por cada 10,000 linhas de código;
- custo de manter o software ultrapassa o dobro do custo de desenvolvimento.

Estatísticas sobre desenvolvimento de Software (IV)

Tipos de erro:

- Erro devido a causas físicas — o sistema físico de suporte falha;
- Erro de software — os nossos conhecidos *bugs*;
- Erro humano — o operador do sistema utiliza-o de forma errada.

Resultados de alguns estudos:

- 60%-90% de todas as falhas são atribuíveis a erro humano (Hollnagel, 1993) .
- 92% das fatalidades consideradas num estudo entre 1979 e 1992 podiam ser atribuidas a problemas na interacção humano-computador, apenas 4% a causas físicas e 3% a erros de software (MacKenzie, 1994).

Mas...

Erro de qual humano?!

Do humano que utiliza o sistema, ou do humano que o desenhou/implementou?

Estatísticas sobre desenvolvimento de Software (V)

Exemplos

Alguns exemplos de sistemas com problemas atribuíveis ao software:

- Sonda Mariner I, Julho de 1962
  Deveria ter voado até Venus. 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.
- Ariane 5, Junho de 1996
  Explodiu no voo inaugural devido a uma série de falhas no software de nevegação. Em circustâncias específicas era efectuada uma divisão por zero. O sistema de segurança consistia em ter redundância: em caso de erro os dados eram processados por outro programa (uma cópia do primeiro! — abordagem adequada para hardware, mas não para software)..

Estatísticas sobre desenvolvimento de Software (VI)

Em conclusão:

Problemas com o desenvolvimento de software:

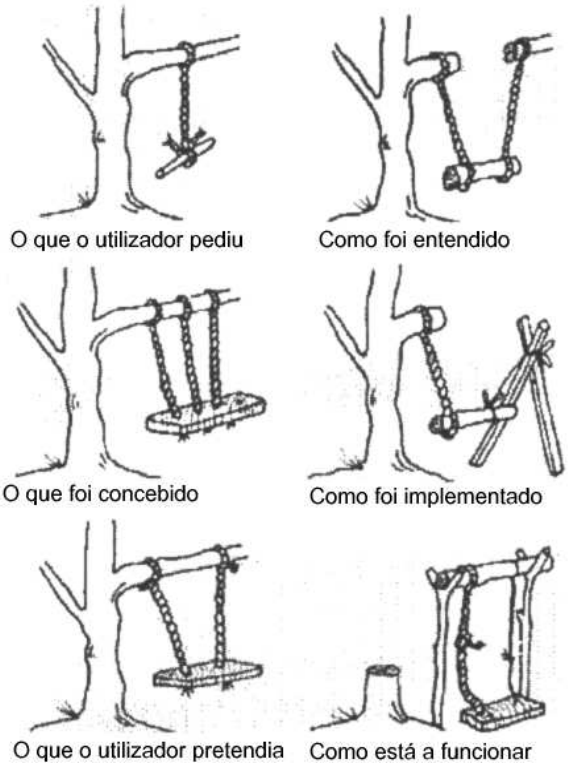
- Atrasos na entrega.
- Incuprimento dos orçamentos.
- Falha na identificação e satisfação das necessidades dos clientes.
- Produtos entregues com falhas.

Algumas causas para o problema:

- Complexidade dos sistemas tem vindo a aumentar
  Quanto mais o hardware evolui, mais somos *tentados a atacar* problemas cada vez mais complexos.
- Condições de utilização do software são cada vez menos controladas
  Cada vez se torna mais difícil prever como/onde o software será utilizado.

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 peritos da organização para:

- compreender que tarefas o sistema deve suportar;
- compreender como o sistema *encaixa* nas actividades da organização.

Um dos maiores desafios é construir o sistema *certo*.

Riscos Tecnológicos

- qual a tecnologia mais apropriada?
- Como controlar a complexidade?

É necessário validar as soluções tecnológicas o mais cedo possível.

Riscos Tecnológicos

- É necessário saber-se o que se está a fazer (obviamente?).
- Exemplo de OO: fácil de aprender/difícil de dominar.

Riscos Tecnológicos

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

Respostas I - Tecnologia

- Primeiras abordagens `*Crise do Software* preocupavam-se mais com a productividade 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.
- Programação orientada aos aspectos (90's)
  Estruturação passa a ser feita, não em termos de objectos, mas em termos de *aspectos de interesse*.

Respostas I – Tecnologia (II)

Orientação aos Objectos

- *Mundo* visto como sendo composto por objectos com identidade própria e capacidade de interacção uns com os outros
- Vantagens reclamadas pela orientação aos objectos:
  - uma formal *natural* e compreensível de pensar/programar;
  - uma forma robusta e estável de programar;
  - facilita a manutenção dos sistemas e aumenta a productividade.
- Quatro aspectos fundamentais da orientação aos objectos:
  - Identidade (dos objectos);
  - Classificação (objecto como instância da sua classe);
  - Polimorfismo (operação+classe=método);
  - Herança. (super/sub-classes)

Respostas I – Tecnologia (III)

Identidade

- cada objecto tem identidade própria;
- podem existir objectos iguais; continuam, no entanto, a serem distintos (cf. gémeos).

Classificação

- os objectos agrupam-se em classes;
- todos os objectos de uma mesma classe possuem o mesmo conjunto de propriedades (atributos), o mesmo comportamento (operações) e as mesmas relações com outros objectos.

Polimorfismo

- a mesma operação pode ter comportamentos diferentes em objectos de classes diferentes;
- é uma forma de abstracção.

Herança (Especialização/Generalização)

- classes são organizadas de acordo com as suas similitudes e diferenças;
- facilita a reutilização.

Respostas I – Tecnologia (IV)

Abstracção

- prestar atenção aos aspectos essenciais, ignorando os detalhes irrelevantes;
- nível de abstracção? demasiada abstracção, crie-se uma sub-classe (especializa-se); demasiado detalhe, crie-se uma super-classe (generaliza-se);
- O desenvolvimento passa a "*middle-out*".

Encapsulamento

- dados e operações correspondentes são agrupados numa única entidade.

Information hiding (ocultação de informação)

- torna-se possível esconder o estado (dados) dos objectos;
- acesso aos dados passa a ter que ser efectuado (de forma mais controlada) através das operações.

Representation hiding (ocultação da representação)

- ao ocultarmos os dados, passa também a ficar oculta a forma como estes estão representados.

Respostas I – Tecnologia (V)

Ferramentas

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

<u>Ferramentas para o UML</u>

- Rational Rose — inicialmente da Rational, agora da IBM;
- Together — inicialmente da TogetherSoft, agora da Borland;
- Poseidon — www.gentleware.com (versão *community edition*);
- Visual Paradigm — www.visual-paradigm.com (versão *community edition*);
- ArgoUML — argouml.tigris.org (open source /BSD)
- plugins para NetBeans, Eclipse, Jbuilder, etc.
- etc., etc., etc.

Respostas II – Métodos de Desenvolvimento

Mas...

- 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 productividade 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/)

Respostas II – Métodos de Desenvolvimento (II)

Processo de desenvolvimento

Um processo define *quem* está a fazer *o quê*, *quando* e como tendo em vista atingir um dado objectivo (Jacobson et al. 99)

Um processo:

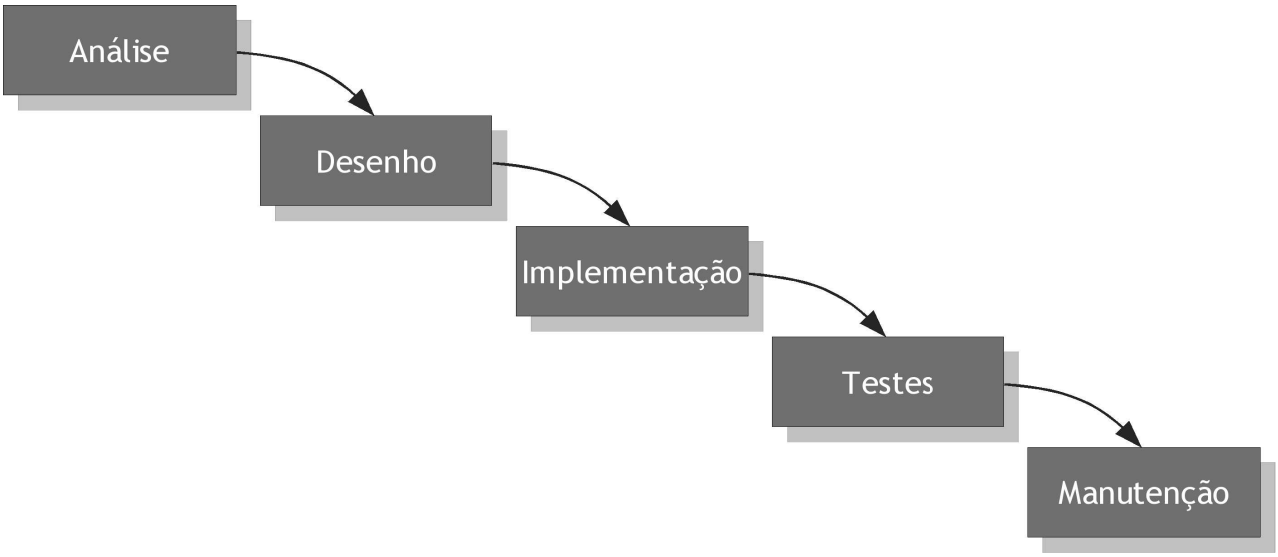
- Identifica um conjunto de regras que definem como o desenvolvimento de um sistema deve ser efectuado.
- Inclui, normalmente, uma descrição dos documentos que devem ser produzidos e em que ordem.
- Pode incluir indicação da linguagem (de modelação) a ser utilizada para a produção dos documentos.

Método de desenvolvimento:

Processo de desenvolvimento + Linguagem de modelação

Respostas II – Métodos de Desenvolvimento (III)

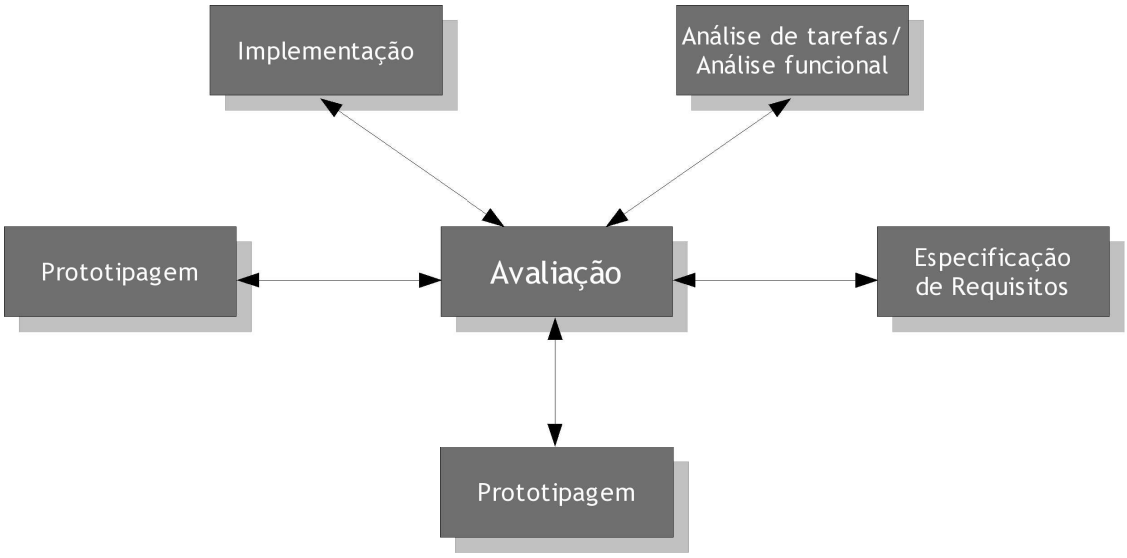
Waterfall



- Este tipo de processo define uma série de etapas executadas sequencialmente.
- Assume que é sempre possível tomar as decisões mais correctas — irrealista!

Respostas II – Métodos de Desenvolvimento (IV)

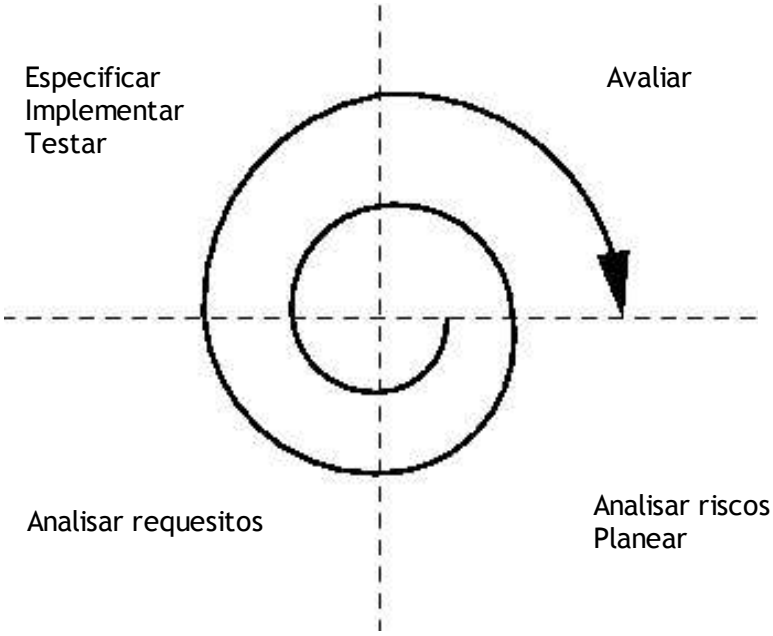
Estrela



- Neste tipo de processo coloca-se em destaque a necessidade de avaliar o sistema em todas as fases de desenvolvimento.

Respostas II – Métodos de Desenvolvimento (V)

Espiral



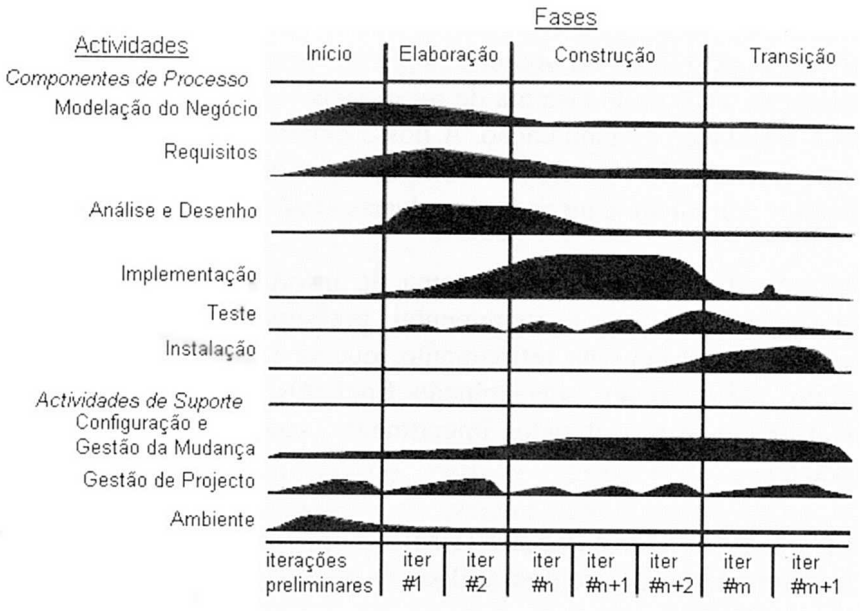
- Neste tipo de processo reconhece-se a necessidade de iterar para controlar riscos.

Respostas II – Métodos de Desenvolvimento (VI)

(Rational) Unified (software development) Process

Um processo:

- guiado por casos de uso (use cases);
- centrado na arquitectura do sistema a desenvolver;
- iterativo e incremental:
  - Início;
  - Elaboração;
  - Construção;
  - Transição.



Respostas II – Métodos de Desenvolvimento (VII)

Início

- Identificar o problema.
- Definir âmbito e natureza do projecto.
- Fazer estudo de viabilidade.

Resultado da fase: decisão de avançar com o projecto.

Elaboração (Análise / Concepção Lógica)

- Identificar o que vai ser construído (quais os requisitos?).
- Identificar como vai ser construído (qual a arquitectura?).
- Definir tecnologia a utilizar.

Resultado da fase: uma arquitectura geral (conceptual) do sistema.

Respostas II – Métodos de Desenvolvimento (VIII)

Construção (Concepção Física/Implementação)

- Processo iterativo e incremental.
- Em cada iteração tratar um (conjunto de) *Use Case*:
  análise / especificação / codificação / teste / integração

Resultado da fase: um sistema de informação!

Transição

- Realização dos acertos finais na instalação do sistema.
- Optimização, formação.

Resultado da fase: um sistema instalado e 100% funcional (espera-se!).

Respostas II – Métodos de Desenvolvimento (IX)

User-Centered Design

- Processo (e filosofia) de desenvolvimento que procura desenvolver produtos úteis e fáceis de utilizar envolvendo os utilizadores no processo de desenvolvimento.
- A base é a constituição de uma equipa de desenvolvimento multi-disciplinar, incluindo representantes dos utilizadores do sistema a desenvolver.
- Cinco princípios base:
  - Definir objectivos de negócio;
  - Compreender os utilizadores;
  - Avaliar a concorrência;
  - Adoptar uma perspectiva holística da utilização do produto;
  - Avaliar diferentes alternativas de desenho;
  - Ouvir/observar sempre os utilizadores.
- Etapas do processo: Definição do mercado / Análise de tarefas / Desenho e *Walkthroughs* / Avaliação e Validação / Comparação com a concorrência.
- Para saber mais: www.ibm.com/ibm/easy/

Respostas II – Métodos de Desenvolvimento (X)

Agile Development:

- Uma reacção ao "*peso burocrático*" que outros processos começam a adequirir.
- Ênfase passa a ser em boas práticas de desenvolvimento e não em processos rígidos.

XP — eXtreme Programming

- Desenvolvimento iterativo e incremental — desenvolvimento numa sequência de pequenos passos;
- Teste contínuos e repetidos (*unit testing* automatizado, *regression testing* — Junit) — o processo de testes substitui a especificação do sistema (os testes devem estar definidos antes da escrita do código);
- Programação "*aos pares*" — equipas de dois programadores: enquanto um programa, o outro analisa o código produzido;
- Integração do cliente na equipa de desenvolvimento — auxilia na definição do sistema, escrita de testes e no esclarecimento de dúvidas;
- *Refactoring* — análise constante do código para remoção de complexidade;
- Partilha da posse do código — todos podem alterar o código a qualquer momento.

Controlar a qualidade do processo

CMM — Capability Maturity Model for Software (SEI/CMU)

- Avaliar a *maturidade* das organizações em termos dos princípios e das práticas adoptados no desenvolvimento.
- Cinco níveis de maturidada:

1) Inicial — processo ad hoc e por veszes mesmo caótico (sucesso depende do esforço individual);
2) Repetível — processos básicos de gestão de projecto, procura-se replicar práticas de projectos bem sucedidos (controlo do custo, tempo, funcionaliade);
3) Definido — existe um processo de gestão e engenharia documentado (todos os projectos utilizam versões adaptadas desse processo);
4) Gerido — são recolhidas medidas de desempenho dos projectos (o processo de desenvolvimenot é compreendido e controlado);
5) Optimização — procura o melhoramento contínuo do processo.

Previsibilidade, eficácia e controlo aumentam ao subir de nível.

Para saber mais: www.sei.cmu.edu/cmm

Método de desenvolvimento:

Processo de desenvolvimento + Linguagem de modelação

Linguagens de Modelação

- Permitem-nos escrever modelos da solução a desenvolver.
- Modelos:
  - "*Thinking made public*";
  - Simplificações da realidade — representações abstractas de um sistema, efectuadas de um ponto de vista específico.
- Abstracção:
  - Mecanismo poderoso para lidar com a complexidade;
- Uma linguagem de modelação tem:
  - léxico — regras que definem quais os elementos válidos da linguagem;
  - sintaxe — regras que definem quais as combinações válidas dos elementos;
  - semântica — regras que definem o significado dos modelos legais.
- A linguagem UML é diagramática — modelos são expressos com diagramas.

Linguagens de Modelação (II)

Tipos de Modelos:

- Preditivos
  Utilizados para prever o comportamento de um sistema.
- Normativos
  Utilizados para definir comportamentos adequados do sistema.
- Descritivos
  Utilizados para descrever a estrutura e comportamento do sistema.

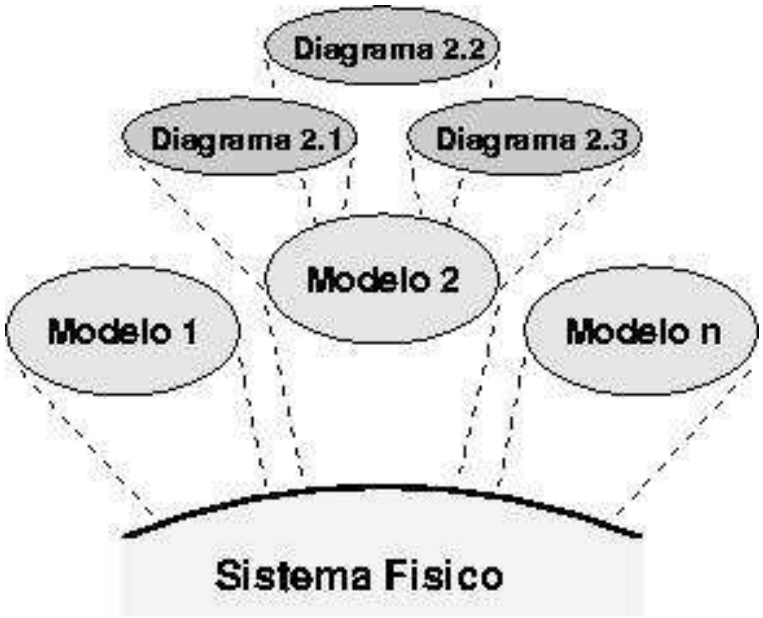
No UML:

- Utilizam-se vários modelos para representar uma mesma realidade
  (os modelos não são a realidade)
- Utilizam-se vários diagramas para representar um modelo
  (os diagramas não são os modelos)

De uma forma geral os modelos do UML são **descritivos**.

Linguagens de Modelação (III)

Os diagramas não são os modelos! / Os modelos não são o sistema!



Linguagens de Modelação (IV)

Vantagens da utilização de modelos

- Auxiliam a **compreender** a realidade.
  Sendo abstracções da realidade, os modelos permitem descrever o que é considerado essencial num dado contexto, escondendo detalhes desnecessários/ irrelevantes nesse contexto.
- Ajudam a **comunicar** ideias de forma simplificada.
  Sendo simplificações da realidade, permitem comunicar apenas os aspectos pretendidos.
- Ajudam a documentar as decisões tomadas durante o desenvolvimento.
  Os modelos desenvolvidos constituem uma base documental para a descrição do processo de desenvolvimento.

Modelos Conceptuais vs. Modelos de Especificação

Linguagens de Modelação (V)

Problemas com a utilização de modelos

- Mais uma "linguagem" a aprender.
  Isso acarreta **custos**, quer monetários (para as organizações), quer cognitivos (para os indivíduos).
- Modelos apresentam uma **visão idealizada da realidade**.
  Existe o risco de durante o processo de modelação nos esquecermos que os modelos são representações da realidade e não a realidade.
  É necessário considerar se estamos a utilizar abstracções adequadas e a modelar todos os aspectos relevantes.
- A fase de modelação **atrasa a produção de código** (pseudo-problema!)
  Espera-se, no entanto, que o código produzido seja de melhor qualidade (assim como o próprio sistema desenvolvido).
  Uma abordagem iterativa e incremental soluciona este *problema*. Por outro lado, é já possível passar, de forma semi-automática, dos modelos para o código.
  Regra dos 5/6 – 1/6 (análise e concepção vs. codificação)
  Atenção à "*analysis paralysis*"!

Linguagens de Modelação (VI)

A nossa linguagem de modelação vai ser o UML

- o UML foi pensado para o desenvolvimento de sistemas orientados aos objectos, mas é independente das linguagens de programação a utilizar
  ⇨ permite explorar o paradigma OO (cf. riscos tecnológicos)
- o UML possibilita o trabalho a diferentes níveis de abstracção
  ⇨ facilita comunicação e análise (cf. riscos de requisitos)
- o UML não é uma linguagem, mas uma família de linguagens gráficas para modelar e construir sistemas software
  ⇨ inclui modelos para as diferentes fases do desenvolvimento
- o UML não é um processo de desenvolvimento de software, mas pode ser utilizado com diferentes processos
- O UML é um *standard* mantido pelo OMG (Object Management Group)
- O UML é suportado por ferramentas
  ⇨ *Rational Rose (IBM), Together (Borland), Visual Paradigm, Poseidon, etc., etc.*

Em resumo

- Entender o problema, pensar (modelar) a solução
- Iterar, iterar, iterar...