INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAHIA

Introdução à Engenharia de *Software*

Profª Jocelma Rios

Jun/2013

O que pretendemos

- Apresentar os **conceitos básicos** de engenharia de *software* e as disciplinas que a compõem
- Apresentar as **questões mais relevantes** que envolvem as práticas relacionadas ao desenvolvimento de *software*
- Refletir sobre a **importância da sistematização e controle dos processos de desenvolvimento de *software*** para a garantia de sua qualidade

A natureza do *Software*

- É ao mesmo tempo um produto e um veículo para distribuir um produto (informação)
- É um **transformador de informações** → produz, gerencia, recupera, modifica, exibe dados e informações que pode ser desde um bit a bilhões de terabytes
- Tem se tornado mais **sofisticado e complexo** a cada dia, em razões das **constantes inovações** em hardware e nos softwares com os quais interage
- Construí-lo deixou de ser uma atividade solitária e passou a exigir equipes multidisciplinares, que precisam seguir **sofisticados processos de trabalho**

A natureza do *Software*

Questões que ainda permanecem...

- Por que concluir um *software* leva tanto **tempo**?
- Por que os **custos** de desenvolvimento são tão altos?
- Por que não conseguimos encontrar todos os **erros** antes de entregarmos o *software* aos clientes?
- Por que gastamos tanto tempo e esforço mantendo os *softwares* existentes?
- Por que continuamos a ter dificuldade em medir o progresso enquanto o *software* está sendo desenvolvido e mantido?

Definição de *Software*

São os programas de computador, a documentação associada e os dados de configuração necessários para que esses programas operem corretamente.

- Podemos classificar os *softwares* em duas categorias:
  - **Produtos genéricos:** produtos desenvolvidos para o mercado; pacotes de software. A especificação do *software* é controlada pela organização que o desenvolve.
  - **Produtos sob encomenda (personalizados):** produtos desenvolvidos para um cliente específico. Sua especificação é controlada pelo cliente → os desenvolvedores devem

Atributos de um bom *Software*

- **Facilidade de manutenção** → o software deve ser escrito de modo que possa evoluir para atender as necessidades mutáveis dos clientes
- **Nível de Confiança** → inclui confiabilidade, segurança e proteção
- **Eficiência** → inclui rapidez de resposta, tempo de processamento, utilização da memória, entre outros
- **Facilidade de Uso (usabilidade)** → o software deve dispor de uma interface apropriada com o usuário e de documentação adequada; deve ser utilizável sem esforços indevidos

Campos de aplicação de SW

- *Software* de sistema
- *Software* de aplicação
- *Software* científico/de engenharia
- *Software* embutido/embarcado
- *Software* para linha de produtos
- Aplicações para a web
- ✔ ***Software* com inteligência artificial**
- ✔ **Aplicações para dispositivos móveis (ubíquas e pervasivas)**
- ✔ ***Software* livre**

DESAFIOS!

Software legado

Softwares já existentes, ainda em operação, que têm sido continuamente modificados para se adequar a mudanças dos requisitos de negócio e plataformas computacionais

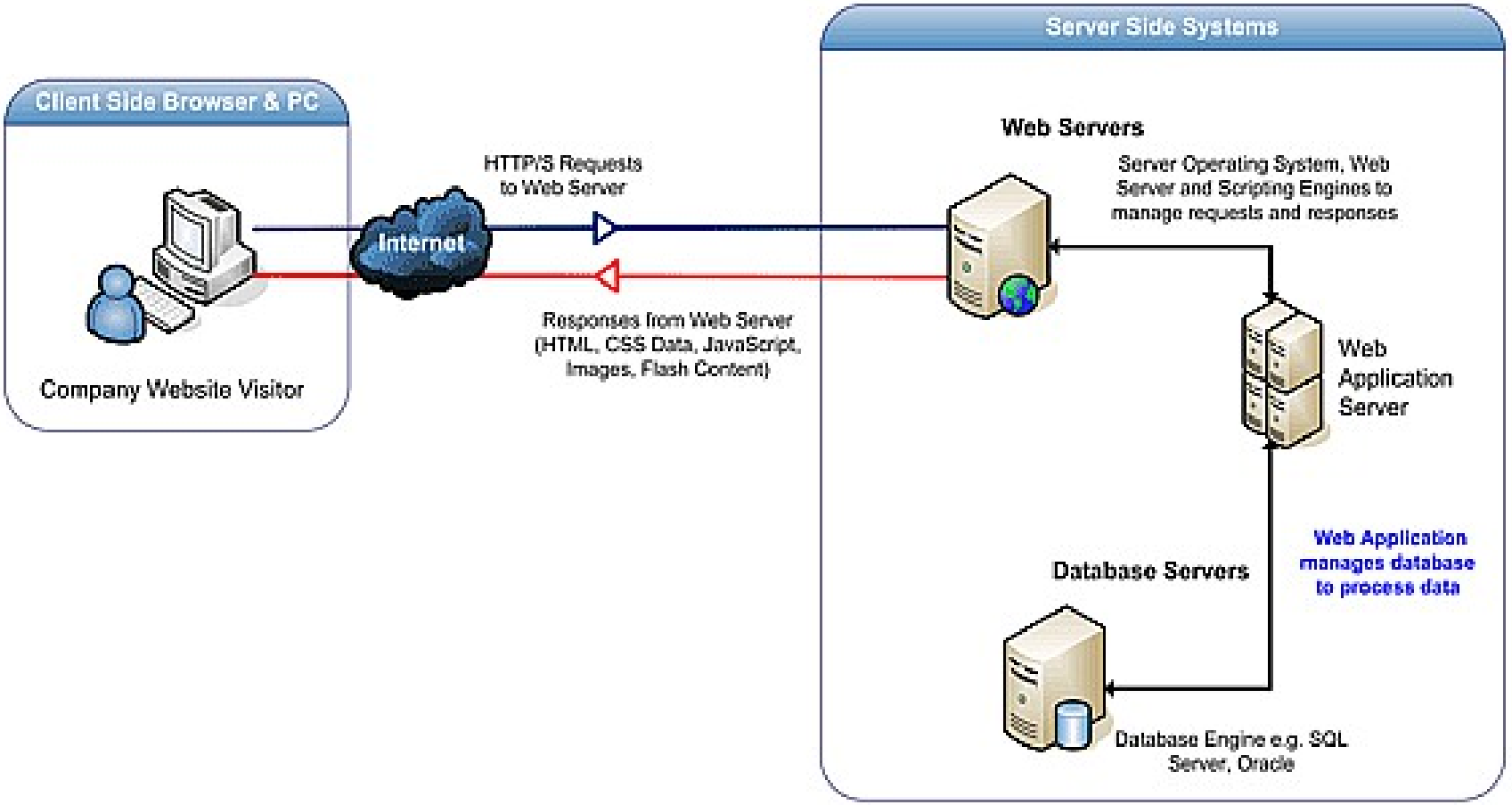
- Costuma gerar muito custo para as organizações, especialmente pela ausência ou fragilidade da documentação

  → **É dispendioso manter e arriscado evoluir**

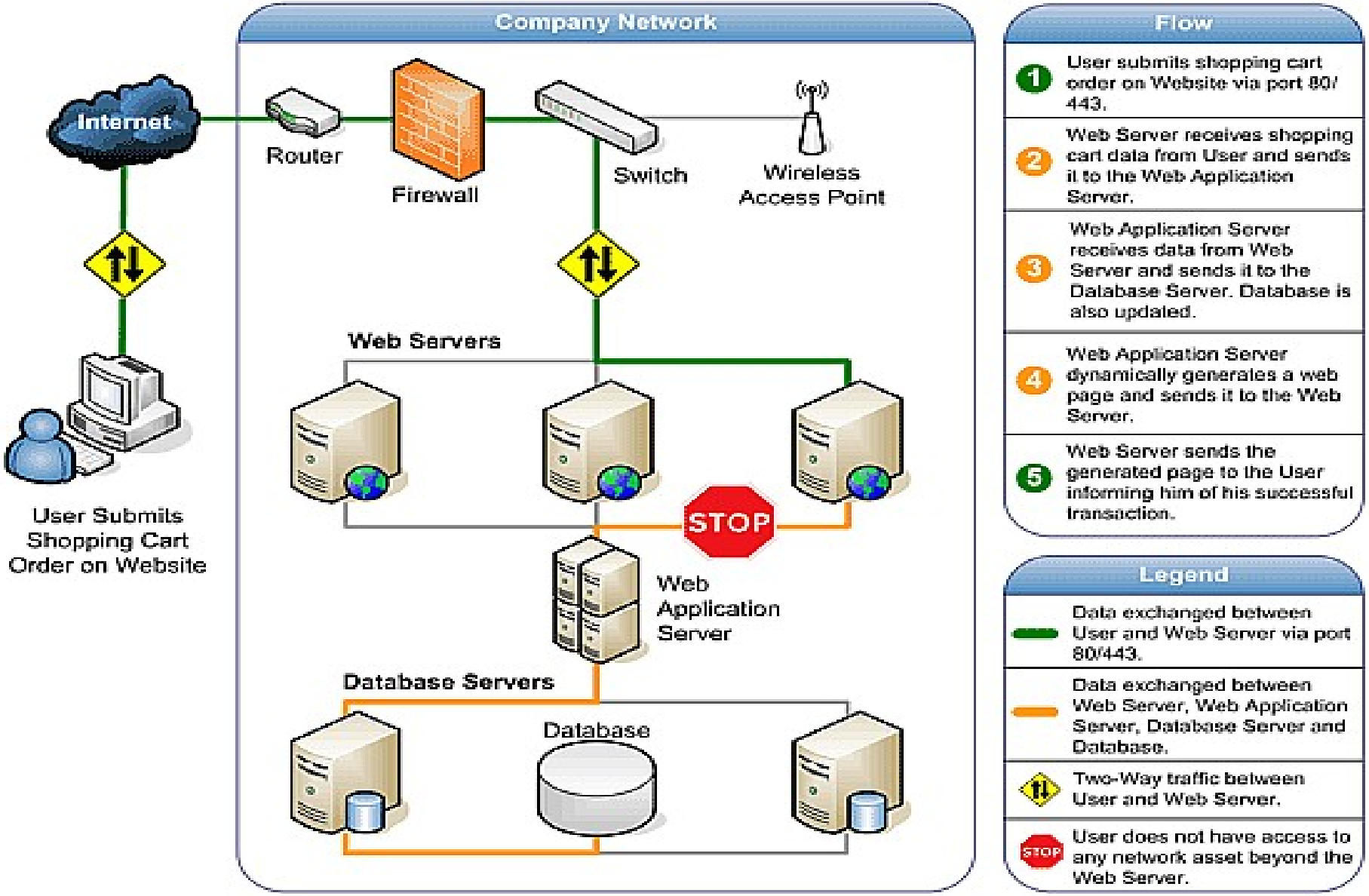
Natureza única das WebApps

Evolução → HTML somente → ontem

→ hoje: XML, Java, PHP, ASP, JavaScript, Perl, VBScript, etc.

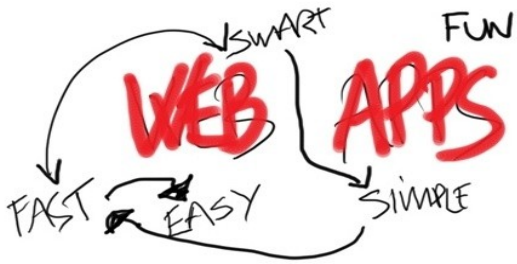


Natureza única das WebApps



Natureza única das WebApps

- Uso intensivo de redes
- Simultaneidade
- Alta disponibilidade
- Alta exigência de desempenho
- Carga de processamento e de acesso não previsível
- Sensibilidade no conteúdo e estética
- Orientado a dados → hipertexto
- Evolução contínua
- Criticidade de segurança
- Curto prazo para o *deployment*



Engenharia de *Software*

- É uma disciplina da engenharia computacional que se ocupa de todos os aspectos da produção de *software*, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema

- Geralmente, os engenheiros de *software* adotam uma **abordagem sistemática e organizada** em seu trabalho, pois essa, com certeza, é a maneira mais eficaz de produzir software de alta qualidade

Engenharia de *Software*

- Tem como meta o desenvolvimento de sistemas de *software* com **boa relação custo-benefício**

- O termo foi mencionado pela 1ª vez em 1968 → **crise do *software*** (hardware poderoso, portanto o software resultante era maior e mais complexo)

- Dessa forma, uma abordagem informal para a construção desses sistemas não era o bastante. Os projetos atrasavam, apresentavam custos elevados, não eram confiáveis, eram de difícil manutenção e tinham desempenho ruim

  → o desenvolvimento de *software* estava em crise

Engenharia de *Software*

- Novas técnicas e novos métodos eram necessários para controlar a complexidade inerente aos sistemas de *software*
  - Essas técnicas se tornaram parte da ES

- Ainda existem problemas, porém houve um grande progresso desde 1968 e o desenvolvimento da ES melhorou de modo marcante o *software* produzido

Engenharia de *Software*

- **Quando um *software* é bem-sucedido?**
  - quando satisfaz as necessidades das pessoas que o usam, tem desempenho sem falhas por um longo período, é fácil de modificar e ainda mais fácil de usar – ele pode e efetivamente modifica as coisas para melhor

- **Mas quando o *software* falha?**
  - quando seus usuários ficam insatisfeitos, quando apresenta erros, quando é difícil de modificar e ainda mais difícil de usar

- **Para obter sucesso**, precisamos de disciplina quando o software é projetado e construído
  → precisamos de uma abordagem de engenharia

Engenharia de *Software*

Pontos-chave

- Compreender o problema antes de pensar na solução
- Projetar é uma atividade fundamental na engenharia de *software*
- Qualidade e facilidade de manutenção são resultantes de um projeto bem feito
- A engenharia de *software* engloba um processo, métodos de gerenciamento e desenvolvimento de *software*, bem como ferramentas
- Atividades de apoio ocorrem ao longo do processo de software e se concentram, principalmente, no gerenciamento, acompanhamento e controle do projeto
- A adaptação do processo de *software* é essencial para o sucesso de um projeto

Engenharia de *Software*

"Mais do que uma
simples disciplina
ou ramo do conhecimento,
engineering é um verbo
[engenhar, engedrar],
uma palavra de ação,
uma maneira de abordar
um problema"

Scott Whitmire

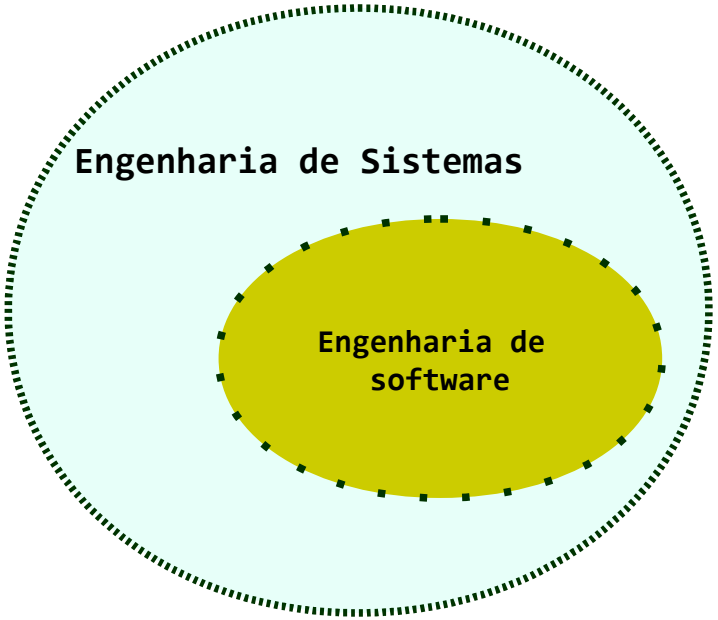
Qual a diferença entre Engenharia de *Software* e Ciência da Computação?

- **Ciência da computação** → ocupa-se da teoria e dos fundamentos referentes aos computadores e sistemas de *software*
- **Engenharia de *software*** → dedica-se aos problemas práticos da produção de software

Teorias mais refinadas da ciência da computação nem sempre podem ser aplicadas a problemas reais e complexos, que requerem uma solução de *software*

Qual a diferença entre Engenharia de *Software* e Engenharia de Sistemas?

- **Engenharia de sistemas** → ocupa-se de todos os aspectos relacionados ao desenvolvimento de sistemas, incluindo *hardware* e *software*
- Engenharia de *software* → é parte desse processo



Processo de *Software*

- É um conjunto de atividades e resultados que geram um produto de *software*
- Há 4 atividades fundamentais – comuns a todos os processos de *software*

    1. Especificação: definição das funcionalidades e restrições
    2. Desenvolvimento: o *software* deve ser produzido de forma a atender as especificações
    3. Validação: garantia de que o *software* faz o que o cliente deseja
    4. Evolução: o *software* deve evoluir para atender as necessidades mutáveis dos clientes

- Diferentes organizações podem utilizar processos diferentes para produzir o mesmo tipo de produto

Processo de *Software*

- É uma representação simplificada de um processo de software, apresentada a partir de uma perspectiva específica
- Um modelo de processo de *software* define a sequência em que as atividades do processo serão realizadas
- Exemplos de Modelos de Processo de *Software*
  - Modelo em Cascata (Ciclo de Vida Clássico)
  - Desenvolvimento Formal (Transformação formal)
  - Desenvolvimento Evolucionário
  - Desenvolvimento Orientado a Reuso (Montagem de um sistema a partir de componentes reutilizáveis)

Modelo Cascata

- Considera as atividades de especificação, desenvolvimento, validação e evolução, que são fundamentais ao processo, e as representa como fases separadas do processo, como a especificação de requisitos, o projeto de *software*, os testes
e assim por diante

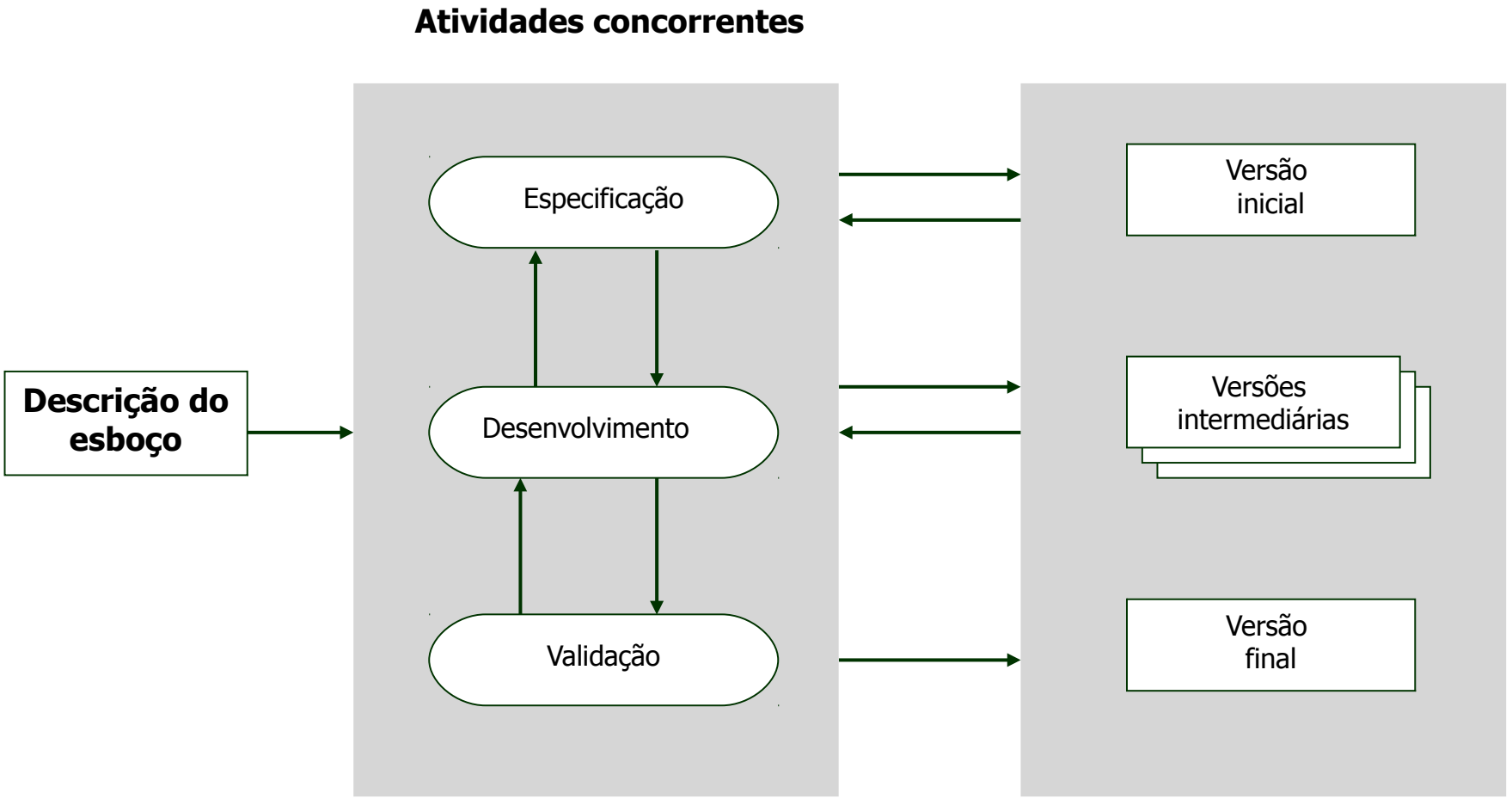
- Após cada estágio ter sido definido, ele é “aprovado” e o desenvolvimento prossegue para o estágio seguinte

Desenvolvimento Evolucionário

- Tem como base a ideia de desenvolver uma implementação inicial, expor ao comentário do usuário/cliente e fazer seu aprimoramento por meio de muitas versões, até que um sistema adequado tenha sido desenvolvido

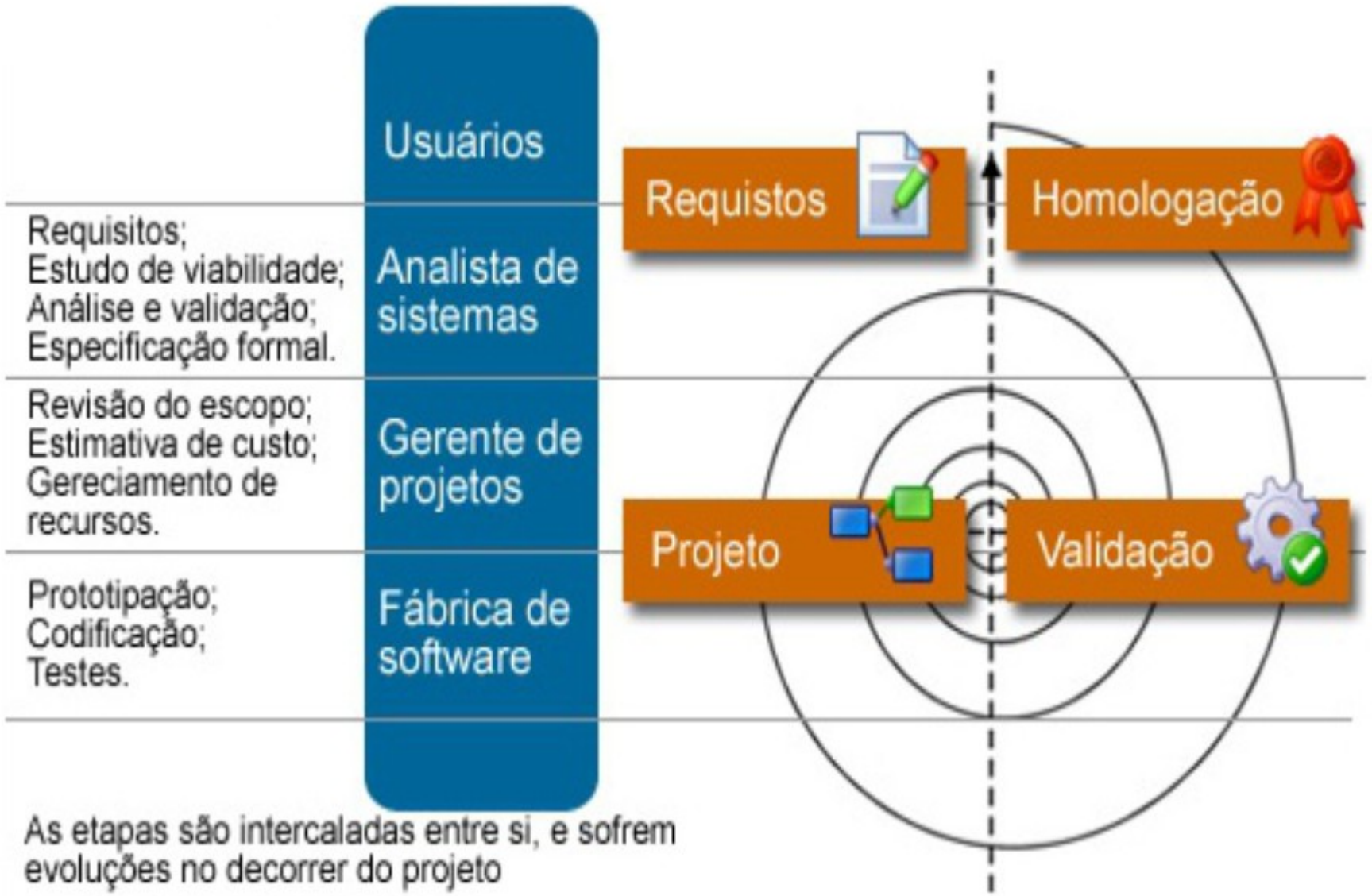
- Em vez de ter as atividades de especificação, desenvolvimento e validação em separado, todo esse trabalho é realizado concorrentemente

Desenvolvimento Evolucionário



- É conhecido também como prototipagem

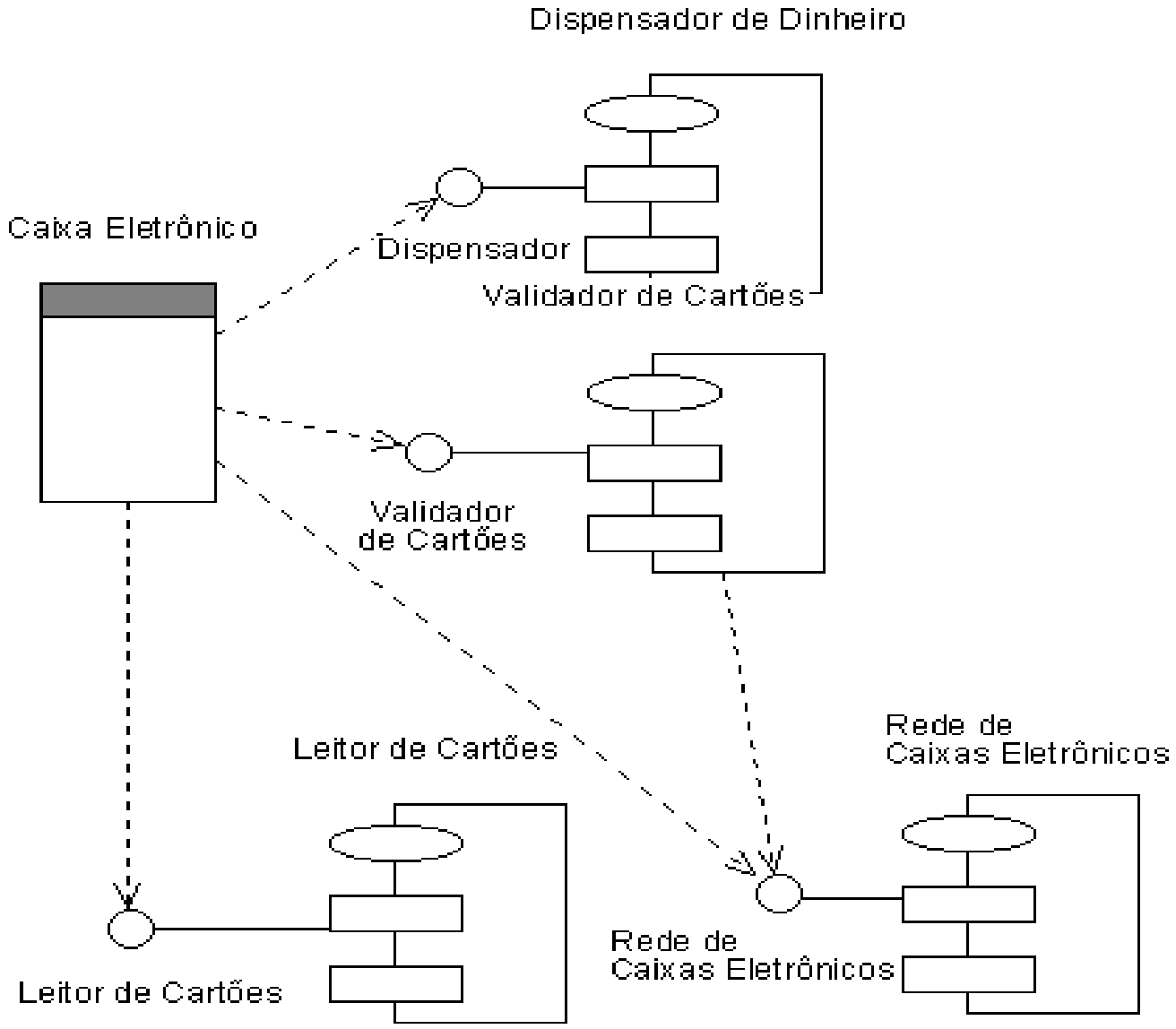
Desenvolvimento Evolucionário



Engenharia de *Software* baseada em componentes

- Baseia-se na existência de um número significativo de componentes reutilizáveis
- O processo de desenvolvimento de sistemas se concentra na integração desses componentes em um sistema, em vez de partir do zero
- Desenvolvimento Baseado em Componentes

Engenharia de *Software* baseada em componentes



Custos da Engenharia de *Software*

- A distribuição dos custos ao longo do processo de *software* depende do modelo de processo utilizado e do tipo de *software* que está sendo desenvolvido
- Aproximadamente 60% dos custos são relacionados ao desenvolvimento e 40% são referentes aos testes
- Para *software* personalizado (com um longo ciclo de duração), os custos de evolução normalmente excedem os custos de desenvolvimento
- *Software* genérico → normalmente tem custo de especificação baixo, porém devem ser extensivamente testados

Métodos da Engenharia de *Software*

- Baseiam-se na ideia de desenvolver modelos de um sistema que possam ser representados graficamente e de utilizar esses modelos como uma especificação ou projeto de *software*

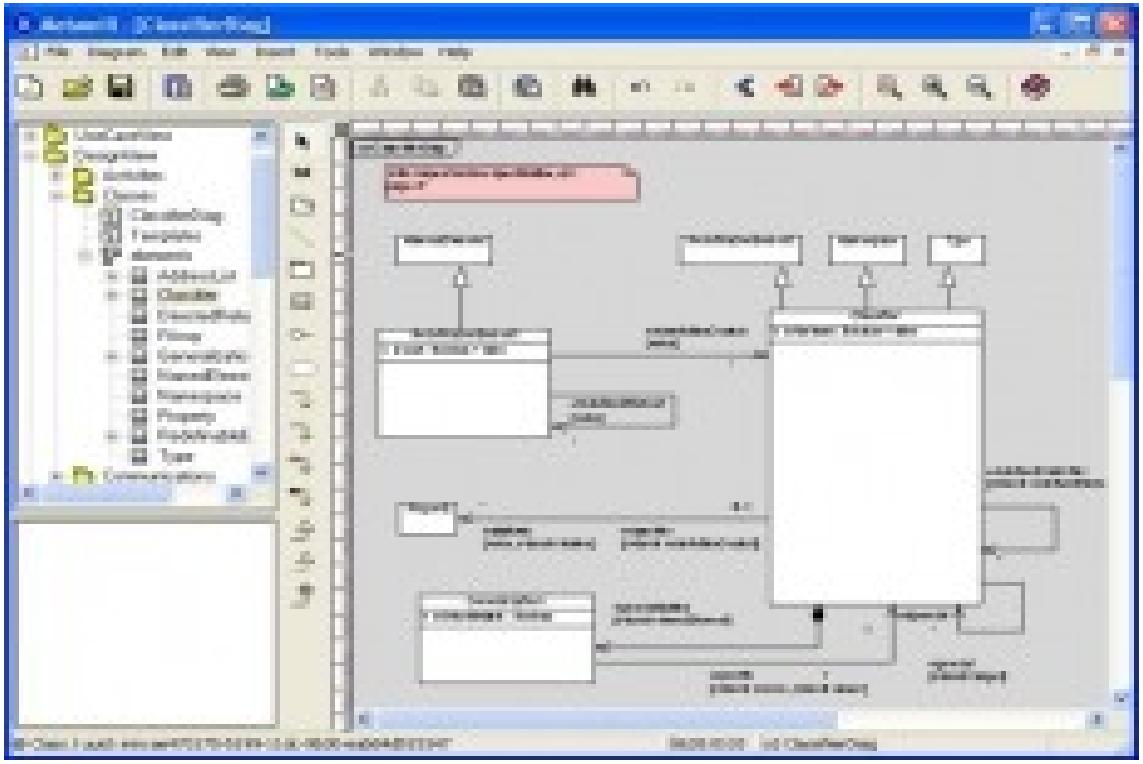
- Métodos
  - Análise Estruturada
  - Orientados a Objetos (UML – *Unified Modeling Language* – Linguagem de Modelagem Unificada)

Ferramenta CASE

(*Computer-Aided Software Engineering*)

- É um *software* destinado a proporcionar apoio automatizado às atividades de processo de software, como a análise de requisitos, modelagem do sistema, depuração e testes

- Alguns exemplos de ferramentas CASE
  - RequisitePro
  - DrCase
  - Rational Rose
  - Astah



Quais são os principais desafios enfrentados pela engenharia de *software*?

- **Desafio da heterogeneidade** → desenvolver técnicas para construir *softwares* confiáveis, que sejam flexíveis para atender diferentes tipos de computadores e diferentes sistemas de apoio

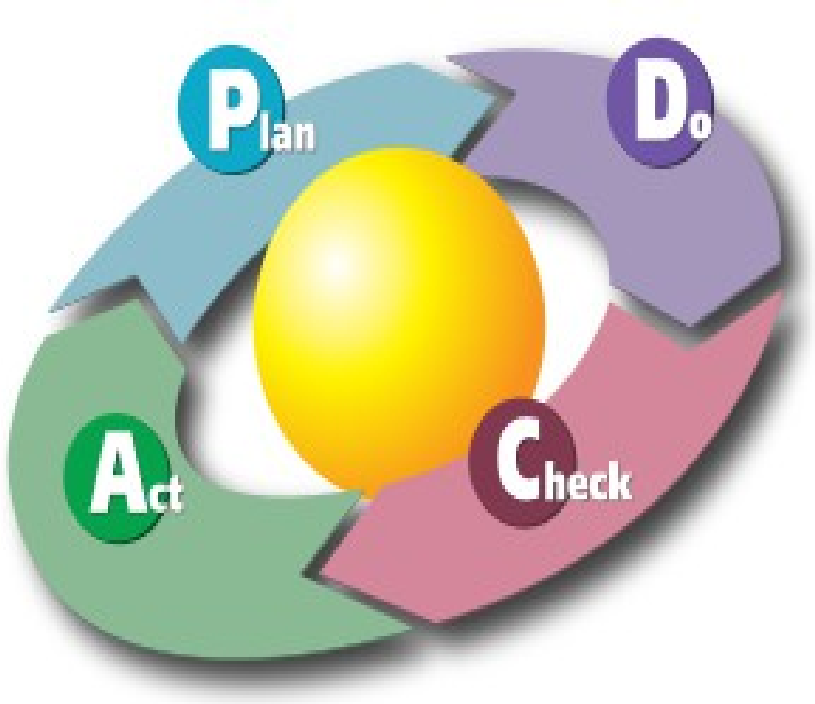
- **Desafio da entrega** → reduzir tempo para entrega de sistemas grandes e complexos, sem comprometer a qualidade

- **Desafio da confiança** → desenvolver técnicas que demonstrem que o *software* pode ter a confiança dos seus usuários

A prática da Engenharia de *Software*

A essência da prática:

- Compreenda o problema
- Planeje a solução
- Execute
- Monitore
- Examine o resultado



A prática da Engenharia de *Software*

Princípios gerais

(1)A razão de existir → usuário e suas demandas

(2)KISS (*Keep It Simple, Stupid!*) → o ótimo é inimigo do bom!

(3)Mantenha a visão → foco no escopo do projeto e preserve a convergência dos artefatos

(4)O que um produz, outros consomem

(5)Pensem no futuro → software não é produto perecível!

(6)Planeje, reutilize

(7)Pense antes de agir

Mitos relativos ao *Software*

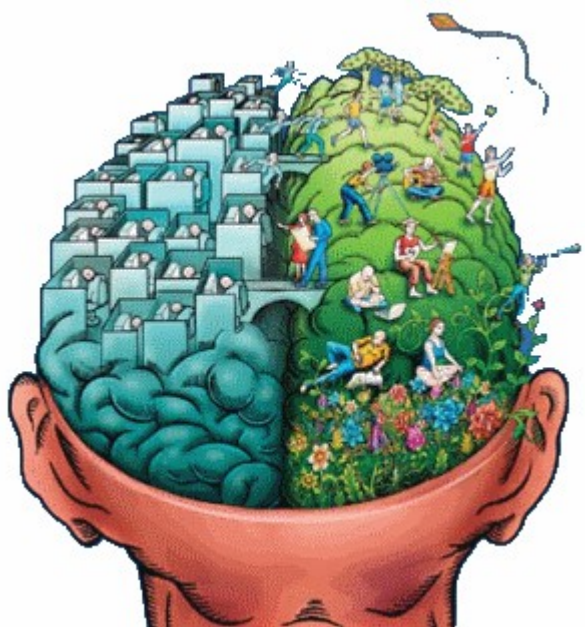
Mitos de gerenciamento:

- Já temos um livro sobre padrões e procedimentos → isso supre meu pessoal do que eles precisam saber
- Se o cronograma atrasar, poderemos acrescentar mais programadores e ficamos em dia → horda mongol
- Se eu decidir terceirizar o projeto de *software*, posso simplesmente relaxar e deixar essa empresa realizá-lo

Mitos relativos ao Software

Mitos dos clientes:

- Uma definição geral dos objetivos é suficiente para começar a escrever os programas → podemos preencher detalhes posteriormente
- Os requisitos de software mudam continuamente, mas as mudanças podem ser facilmente



Mitos relativos ao Software

Mitos dos profissionais da área:

- Uma vez desenvolvido o *software* e colocado em uso, nosso trabalho está terminado
- Até que o software entre em operação, não há maneira de avaliar sua qualidade
- O único produto passível de entrega é código executável
- A engenharia de *software* nos fará criar documentação volumosa e desnecessária, e invariavelmente, produzirá atraso no cronograma

Os profissionais da Engenharia de *Software*

Os profissionais da Engenharia de *Software*

- Analista de Negócios
- Analista de Sistemas
- Programador / Analista de Desenvolvimento
- Analista de Testes
- Analista de Qualidade
- Analista de Suporte / Analista de infraestrutura
- Administrador de Banco de Dados
- Administrador de Rede
- Gerente de Projeto

Como tudo começou...

- Todo projeto de *software* é motivado por alguma necessidade de negócio, por exemplo:
  - Automatizar determinado processo de negócio
  - Corrigir defeitos numa aplicação existente
  - Adaptar um sistema legado a um ambiente de negócios em constante transformação
  - Estender funções/recursos de uma aplicação existente
  - Criar novo produto ou serviço

Sintetizando...

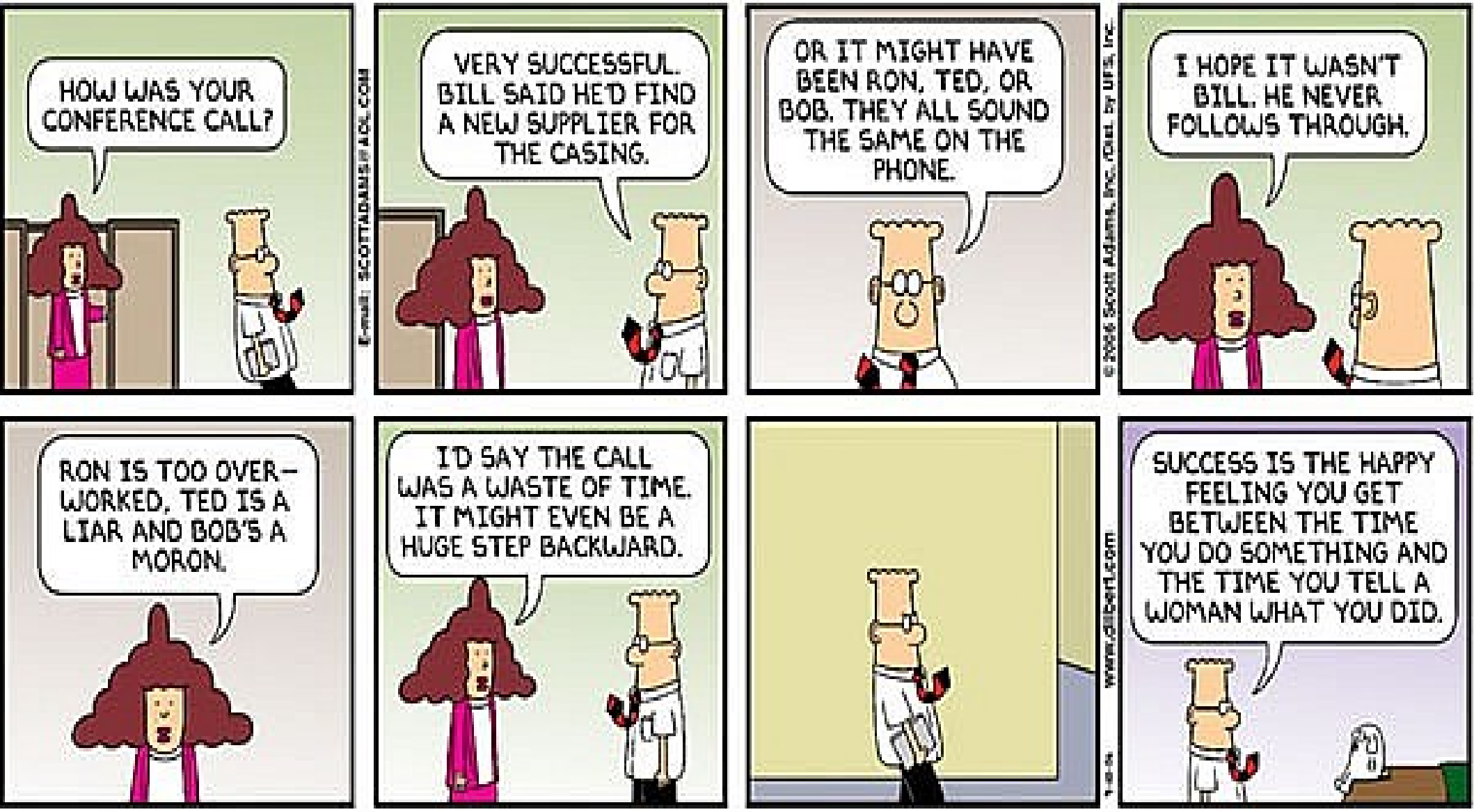
Software é o elemento-chave na evolução de produtos e sistemas baseados em computador e uma das mais importantes tecnologias no cenário mundial.

Ao longo dos últimos 50 anos, o *software* **evoluiu** de uma ferramenta especializada em análise de informações e resolução de problemas para uma indústria propriamente dita.

Mesmo assim, **ainda temos problemas para desenvolver *software* de boa qualidade dentro do prazo e orçamento estabelecidos.**

Sintetizando...

- O ***software* legado** continuar a representar desafios especiais àqueles que precisam fazer sua manutenção.
- **A engenharia de *software* engloba processos, métodos e ferramentas** que possibilitam a construção de sistemas complexos baseados em computador dentro do prazo e com qualidade.
- Inúmeros **mitos** em relação a *software* e seu desenvolvimento continuam a levar gerentes e profissionais para o mau caminho, mesmo com todo o conhecimento coletivo de *software* e das tecnologias necessárias para construí-los.



Para refletir...

Ao afirmarmos que as atividade de modelagem se aplicam a todos os projetos, isso significa que as mesmas tarefas são aplicadas a todos os projetos, independentemente de seu tamanho e complexidade?

Para refletir...

À medida que o *software* invade todos os setores, riscos ao público (devido a erros) passam a ser uma preocupação cada vez maior.

Que cenários negativos poderiam se constituir, caso o *software* não tivesse sido desenvolvido seguindo os padrões de qualidade?

Referências

- PRESMANN, R. **Engenharia de Software**: uma abordagem profissional. 7. ed. Rio de Janeiro: Mc Graw Hill, 2011. Cap. 1
- SOMMERVILLE, I. **Engenharia de Software**. 8. ed. Rio de Janeiro: Pearson, 2007. Cap. 1 e 2.

Vídeos sugeridos

- Ciência da Computação, Sistemas de Informação ou Engenharia da Computação?
  - http://olhardigital.uol.com.br/negocios/central_de_videos/ciencia_da_computacao_sistemas_de_informacao_ou_engenharia_da_computacao
- Introdução Engenharia de Software por Adriano Soares
  - www.youtube.com/watch?v=utujCSeBnUI
- Aula 1 - Engenharia de Software (Introdução) -> aulas desenvolvida por alunos
  - Parte 01 - www.youtube.com/watch?v=FmBWDwTqMbo
  - Parte 02 - www.youtube.com/watch?v=-7jXQlAWaQQ
- Videoaulas ITnerantes: ESW em Exercicios - FCC 2/2 - Fernando Pedrosa
  - www.youtube.com/watch?v=u5aY2gObt0A