Engenharia de Software

Modelos de Ciclo de Vida, Métodos Ágeis, Ferramentas CASE

Fernando Pedrosa – fpedrosa@gmail.com

Bibliografia

- **Pressman**, Roger S. Software Engineering: A Practiotioner's Approach. **Editora**: McGraw-Hill.
- **Sommerville,** Ian. Software Engineering. **Editora**: Addison Wesley.

Engenharia de Software: o que é?

- Disciplina de **engenharia** preocupada com todos os aspectos sobre a produção de software, incluindo:
- Processos
  - Racionalizam o desenvolvimento de Software
- Métodos
  - Conhecimento técnico; “Como” fazer
- Ferramentas
  - Suporte automatizado para processos e métodos

Desenvolver software não é só programar!

Objetivos

- Obter **software de qualidade**
- Com produtividade no seu desenvolvimento, operação e manutenção
- Dentro de custos, prazos e níveis de qualidade controlados
- Com o melhor custo-benefício entre **Qualidade e Produtividade**

Engenharia de Software versus Engenharia de Sistemas

- Engenharia de Sistemas é algo maior: preocupa-se com todos os aspectos de sistemas baseados em computador
  - Software
  - Hardware
  - Processos
  - Pessoas e outros sistemas, etc.
- Engenharia de Software é apenas parte deste processo

Exercícios [0]

(TRE/BA – CESPE 2010)

[61] A engenharia de software está relacionada com todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até sua manutenção, depois que este entrar em operação. A engenharia de sistemas diz respeito aos aspectos do desenvolvimento e da evolução de sistemas complexos, nos quais o software desempenha um papel importante.

Exercícios [0]

(IBGE – CONSULPLAN 2009)

[12] Segundo Pressman (1995), Engenharia de Software é o estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais, abrangendo um conjunto de três elementos fundamentais (métodos, ferramentas e procedimentos).

Assinale a alternativa INCORRETA:

A) Métodos de Engenharia de Software proporcionam os detalhes de "como fazer" para construir o software.

B) As ferramentas proporcionam apoio automatizado ou semi-automatizado aos métodos.

C) Procedimentos constituem o elo de ligação dos métodos e das ferramentas e possibilitam o desenvolvimento racional e oportuno de software.

Exercícios [0]

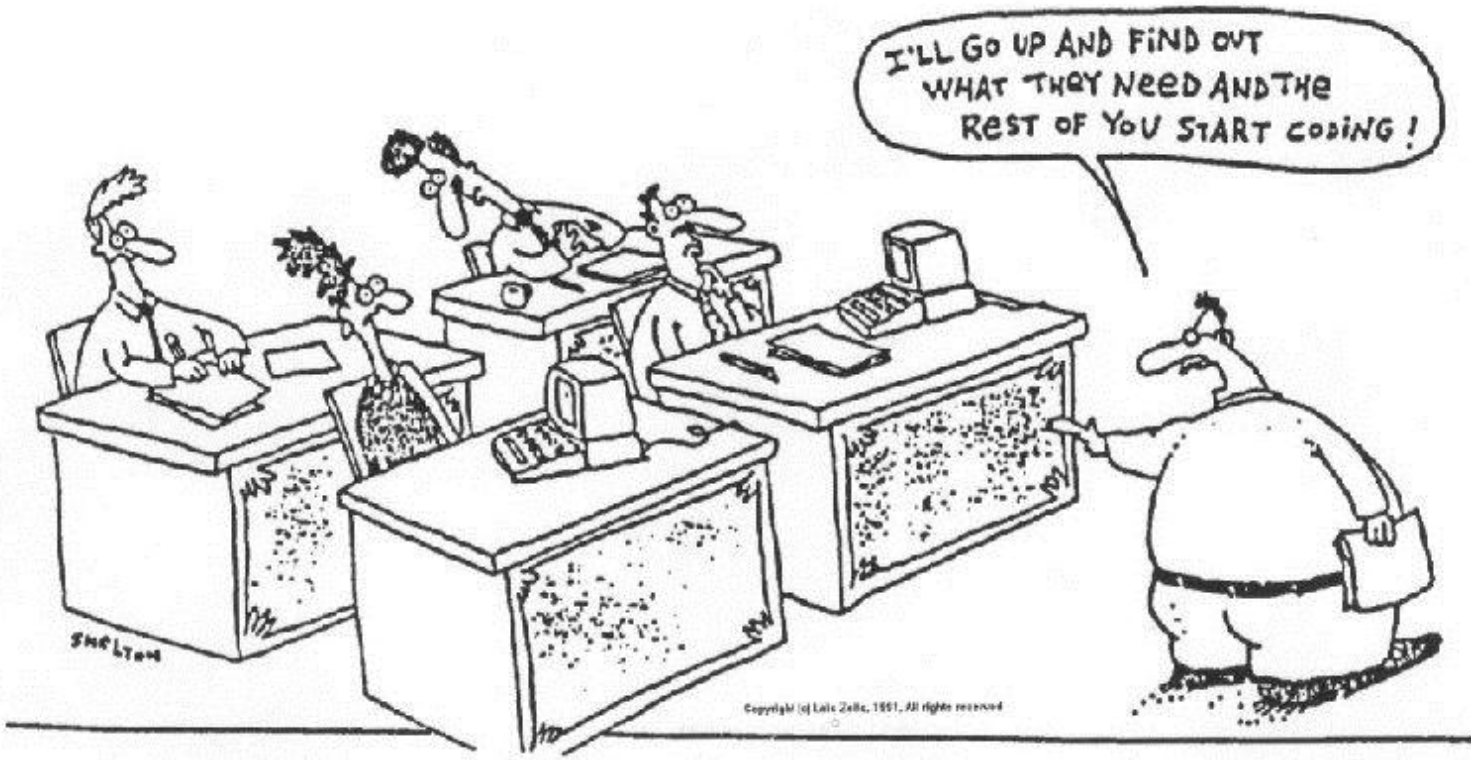
D) Métodos envolvem um amplo conjunto de tarefas que incluem: planejamento e estimativa de projeto, análise de requisitos de software e sistemas, projeto de estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste e manutenção.

E) Ferramentas são roteiros para o desenvolvimento de software

Histórico

- **Década de 60**: a chamada "Crise do Software"
  - Desenvolvimento fora de controle
  - Iniciou como um problema de Custo e Produtividade.
  - Mais importante: era um problema de Qualidade
- **Década de 70**
  - Programação Estruturada
  - Projeto Estruturado

Antigamente...



Histórico

- **Década de 80**
  - Análise Estruturada (DFDs, Dicionário de Dados, Diagrama ER, de Estados, etc.)
  - Ferramentas CASE
- **Década de 90**
  - Análise e Projeto OO.
  - Java
  - UML
  - Processo Unificado

Histórico

- **Anos 2000**
  - Metodologias Ágeis
  - Novos paradigmas: SOA, Aspectos, Model-Driven Architecture, etc.
  - Cloud Computing

Terminologia

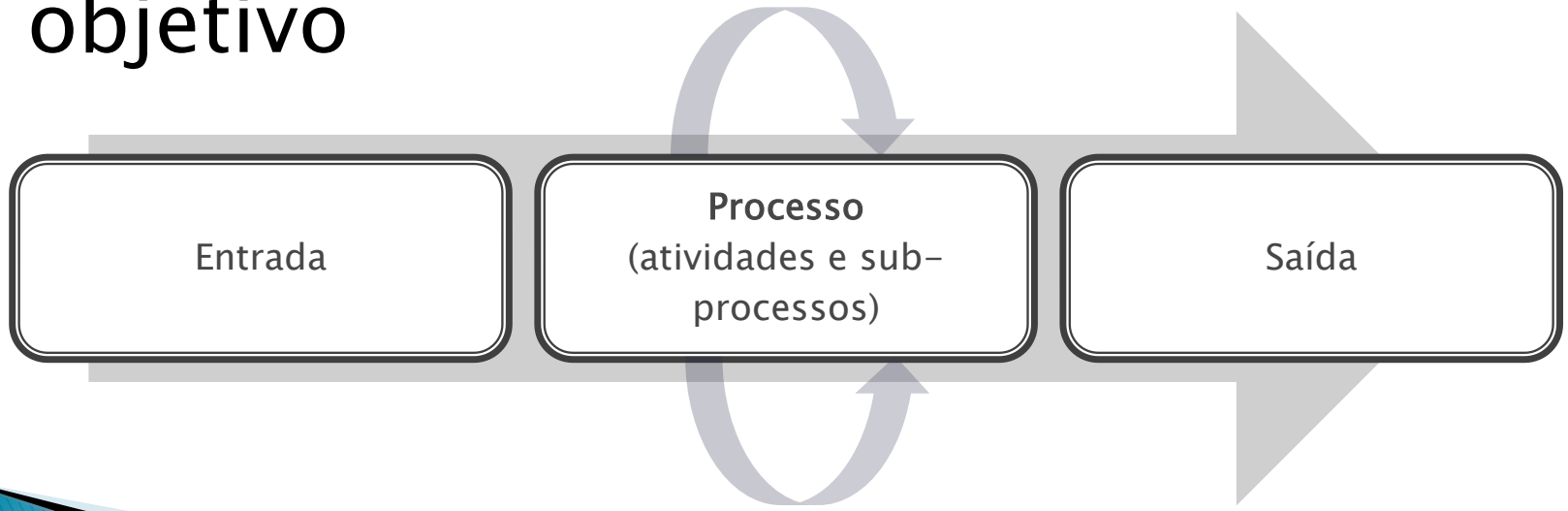
Software

- Programa de computador e documentação associada
- Produtos de software podem ser desenvolvidos para um cliente particular ou podem ser desenvolvidos para um mercado geral
- Novos softwares podem ser criados desenvolvendo-se novos programas ou reusando softwares existentes

Terminologia

Processo

- Uma série conectada de ações, com a intenção de satisfazer um objetivo
- Define quem está fazendo o quê, quando e como para atingir um certo objetivo



Terminologia

Processo de Software

- Um conjunto estruturado de atividades para desenvolver um sistema de software
  - Especificação
  - Projeto
  - Validação
  - Evolução

Modelos de Ciclo de Vida

Modelos de Ciclo de Vida

- “São uma representação abstrata e simplificada do processo de desenvolvimento software, apresentada a partir de uma perspectiva específica”
- Tipicamente contêm:
  - “Esqueleto do processo”
  - Ordem de precedência das atividades
  - Principais artefatos e produtos gerados

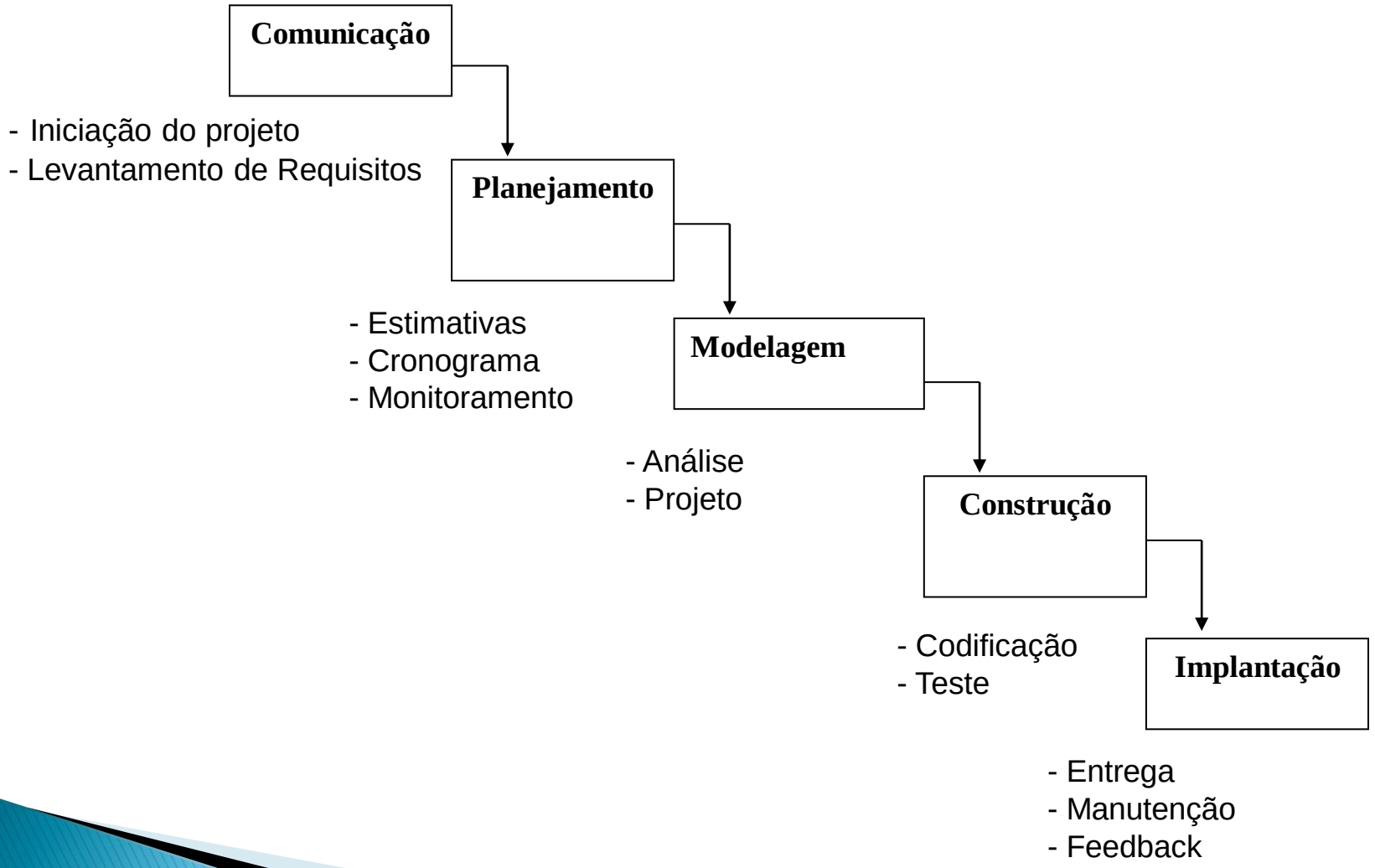
Principais modelos

- Cascata ou Clássico
- Prototipagem
- Métodos formais
- Espiral
- Incremental

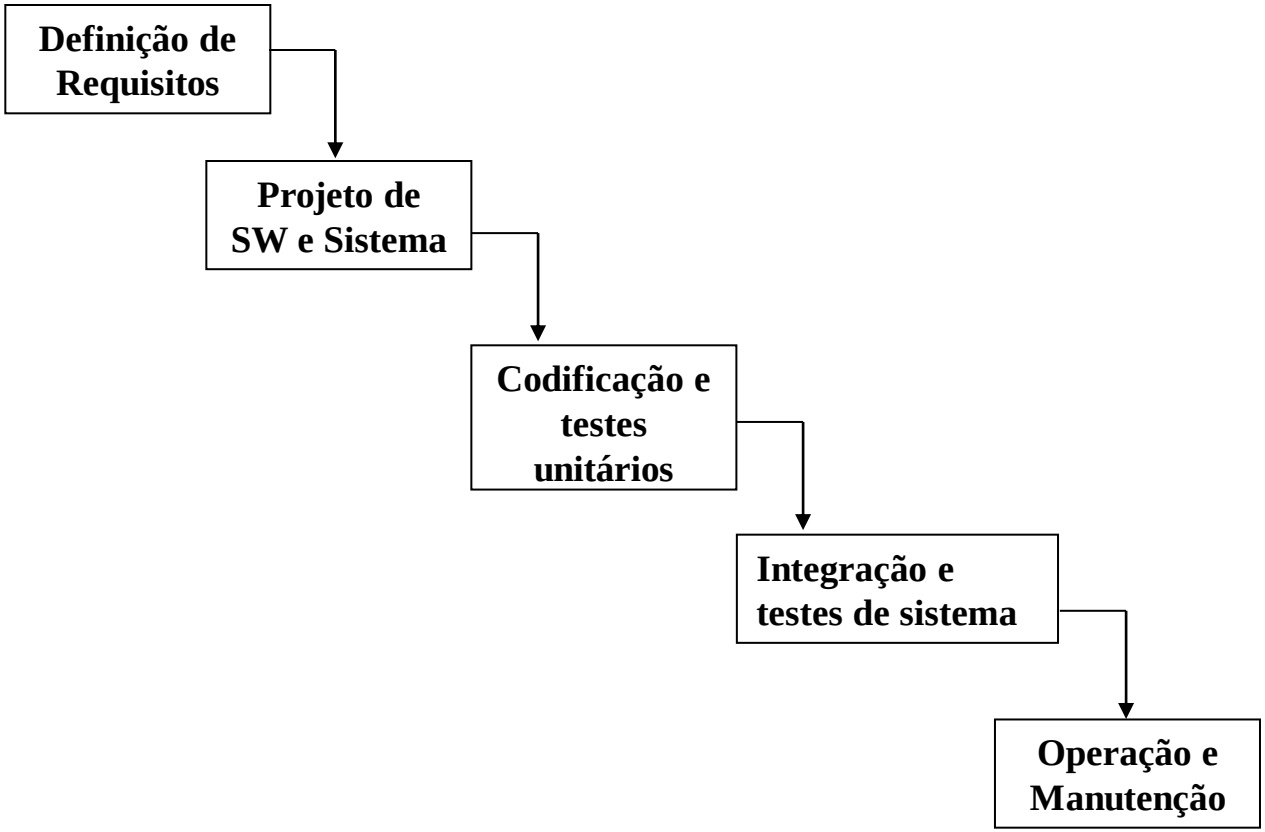
Modelo em Cascata

- Modelo “Clássico”, teve origem na indústria de manufatura e construção
- Sua estrutura é composta por várias etapas que são executadas de forma sistemática e seqüencial
- Na falta de uma abordagem estruturada, foi a primeira tentativa de formalizar uma metodologia de desenvolvimento de software

Modelo em Cascata (Pressman)



Modelo em Cascata (Sommerville)

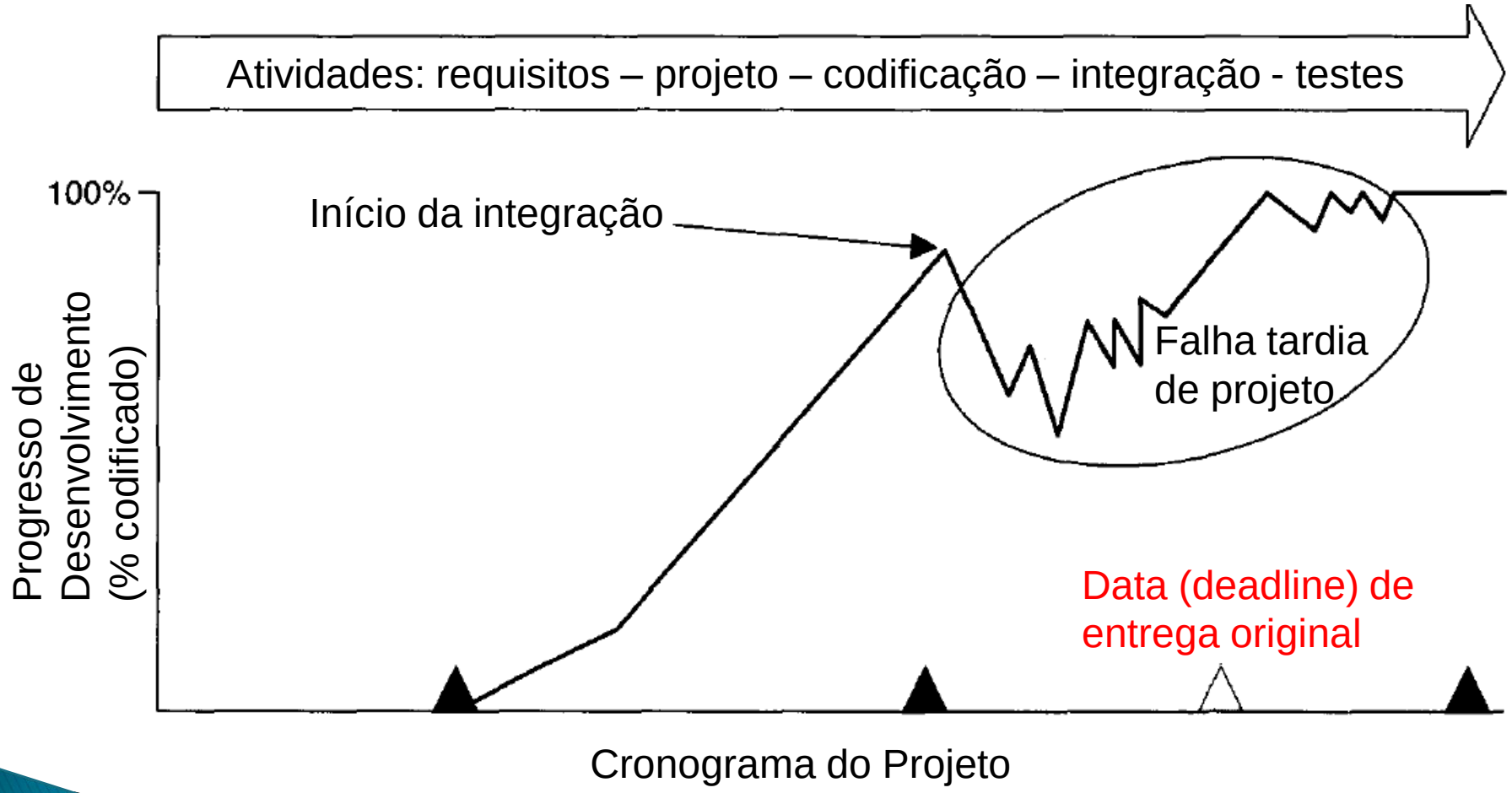


Modelo em Cascata

- Minimiza o planejamento, organiza as atividades em uma sequência com entregas bem definidas
- Funciona bem para requisitos estáveis e bem compreendidos
  - O modelo pressupõe que os requisitos ficarão estáveis ao longo do projeto
- É facilmente aplicável em equipes inexperientes

Porém, atrasa a redução de riscos!

O modelo em cascata atrasa a resolução dos riscos



Exercícios [1]

(TST – CESPE 2008)

[93] No modelo de desenvolvimento seqüencial linear, a fase de codificação é a que gera erros de maior custo de correção.

(SERPRO – CESPE 2008)

[63] O modelo em cascata consiste de fases e atividades que devem ser realizadas em seqüência, de forma que uma atividade é requisito da outra.

(INMETRO – CESPE 2009)

[85] Em um processo de desenvolvimento em cascata, os testes de software são realizados todos em um mesmo estágio, que acontece após a finalização da fase de implementação.

Exercícios [1]

[86] Em uma empresa que tenha adotado um processo de desenvolvimento de software em cascata, falhas no levantamento de requisitos têm maior possibilidade de gerar grandes prejuízos do que naquelas que tenham adotado desenvolvimento evolucionário.

(SAD/PE – CESPE 2010 – adaptada)

[58] O gerente geral de projetos da empresa decidiu, junto a um cliente, realizar algumas modificações nos requisitos de um produto de software que já se encontrava na fase de testes e comprometeu-se a incluir tais requisitos na próxima liberação do produto. Essa decisão permite inferir que o modelo de desenvolvimento de software empregado não é do tipo cascata.

Fases do ciclo de vida do Software

- **Planejamento**
  - Esboçar escopo e requisitos
  - Fazer estimativas razoáveis sobre recursos, custos e prazos

- **Análise e Especificação de Requisitos**
  - Refinar requisitos e escopo
  - Entender o domínio do problema, com comportamento e funcionalidades esperados

Fases do ciclo de vida do Software

- **Projeto**
  - Incorporar requisitos tecnológicos aos requisitos essenciais do sistema
  - Projetar a arquitetura do sistema

- **Implementação**
  - Traduzir o projeto em uma forma passível de execução pela máquina
  - Codificação

Fases do ciclo de vida do Software

- **Testes**
  - Realizar diversos níveis de teste, de forma a fazer a verificação do software.

- **Implantação, Operação e Manutenção**
  - Colocar o software em produção
  - Treinar pessoas
  - Manter o software
  - Gerenciar os serviços

Prototipagem Evolucionária

- O objetivo é trabalhar junto aos clientes para evoluir o sistema a partir de uma especificação inicial resumida
  - Entregar resultado o mais rápido possível
- Deve começar com requisitos mais bem compreendidos
- Novas funcionalidades são adicionadas à medida que o cliente as propõem
- Aplicável em sistemas pequenos ou médios com curto tempo de vida

Prototipagem Evolucionária

Problemas

- Falta de visibilidade do progresso
  - O sistema está sempre evoluindo, nunca está “terminado”
- Os sistemas acabam tornando-se pobremente estruturados
- A documentação pode ser esquecida
- Os padrões de qualidade podem ser “relaxados”

Prototipagem Descartável

- Assim como na prototipagem evolucionária, pequenas versões prototípicas são disponibilizadas ao clientes para avaliação
- Porém, o objetivo aqui é entender e clarificar os requisitos do sistema
- Deve-se começar com os requisitos mais difíceis e menos compreendidos
- Ao final, descarta-se o protótipo e a implementação do software continua

Prototipagem Descartável

- É útil para sistemas grandes e complicados, e quando o cliente não sabe exatamente o que quer
- Protótipos descartáveis podem ser aplicados no contexto de qualquer modelo de processo
  - Cascata
  - Espiral
  - Iterativo/Incremental, etc.

Prototipagem

Exercícios [2]

(TRE/MT – CESPE 2010 – adaptada)

[41] A metodologia de prototipagem evolutiva é uma abordagem que visualiza o desenvolvimento de concepções do sistema conforme o andamento do projeto, por meio de protótipos visuais.

(UNIPAMPA – CESPE 2009)

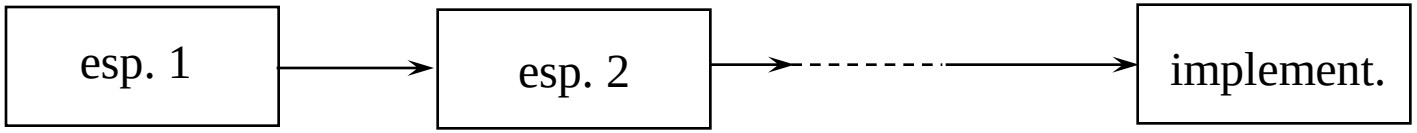
[73] No modelo de desenvolvimento prototipagem, um protótipo é desenvolvido para ajudar no entendimento dos requisitos do sistema.

(TJDFT – CESPE 2008)

[70] O modelo de desenvolvimento por prototipação é caracterizado pela ausência de métricas de controle, dada a natureza experimental do desenvolvimento e do produto obtido.

Métodos Formais

- Modelo baseado em técnicas matemáticas para especificar, desenvolver e verificar software
- O software é especificado usando técnicas formais (matemáticas), e após a "prova" da especificação é transformado em código



Métodos Formais

- O próprio processo de desenvolvimento garante que o programa faz exatamente o que foi especificado
- É possível gerar programas corretos e completos por construção
- Têm sido aplicados apenas ao desenvolvimento de sistemas críticos (por questões de segurança!)

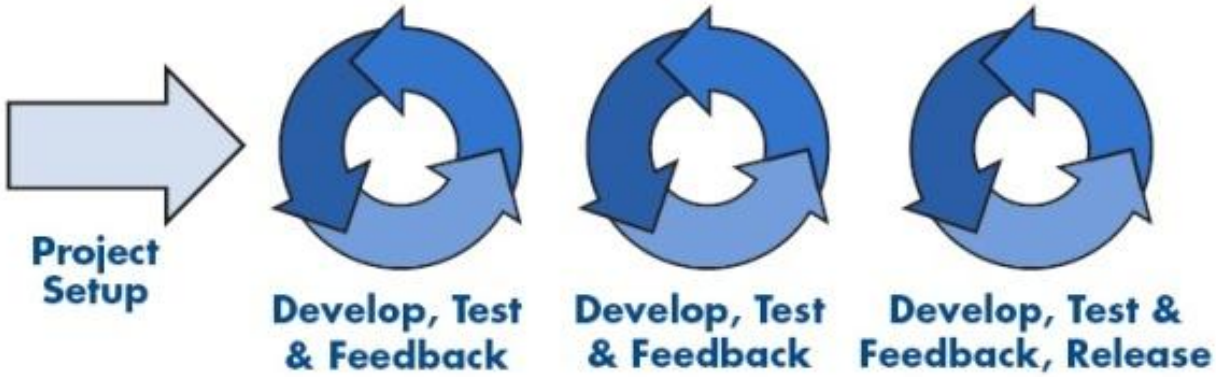
Exercícios [3]

(SERPRO – CESPE 2008)

[65] Para a especificação de software e verificação de sistemas, uma alternativa que se fundamenta na matemática discreta e na lógica é o modelo incremental.

Modelos Iterativos

- Motivação: requisitos de sistema **sempre** evoluem durante o projeto
- Deve-se dividir para conquistar
- Duas abordagens
  - Desenvolvimento Incremental
  - Desenvolvimento Espiral



Modelos Iterativos

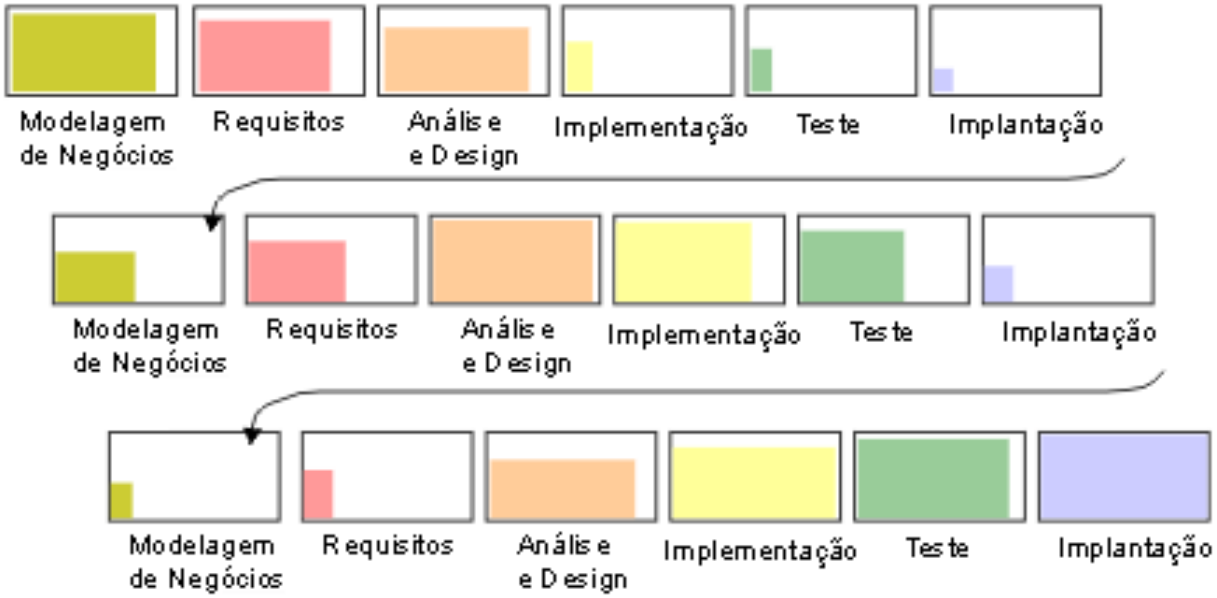
Desenvolvimento Incremental

- A idéia é de desenvolver e entregar o software em incrementos, com cada incremento entregando parte da funcionalidade requerida
- Requisitos são definidos antes do desenvolvimento do incremento, sendo os mais críticos priorizados

Modelos Iterativos

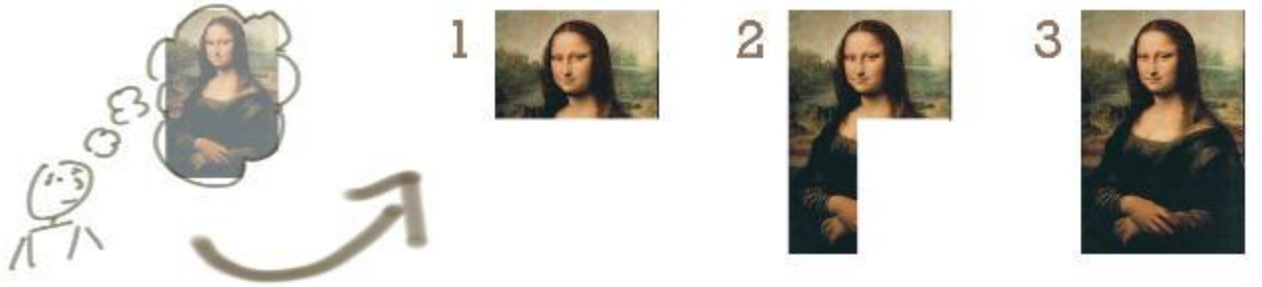
Desenvolvimento Incremental

- Um “mini projeto em cascata” é executado em cada iteração, progredindo até a entrega final do produto



Iterativo x Incremental

- Incremental: são adicionados “pedaços completos”



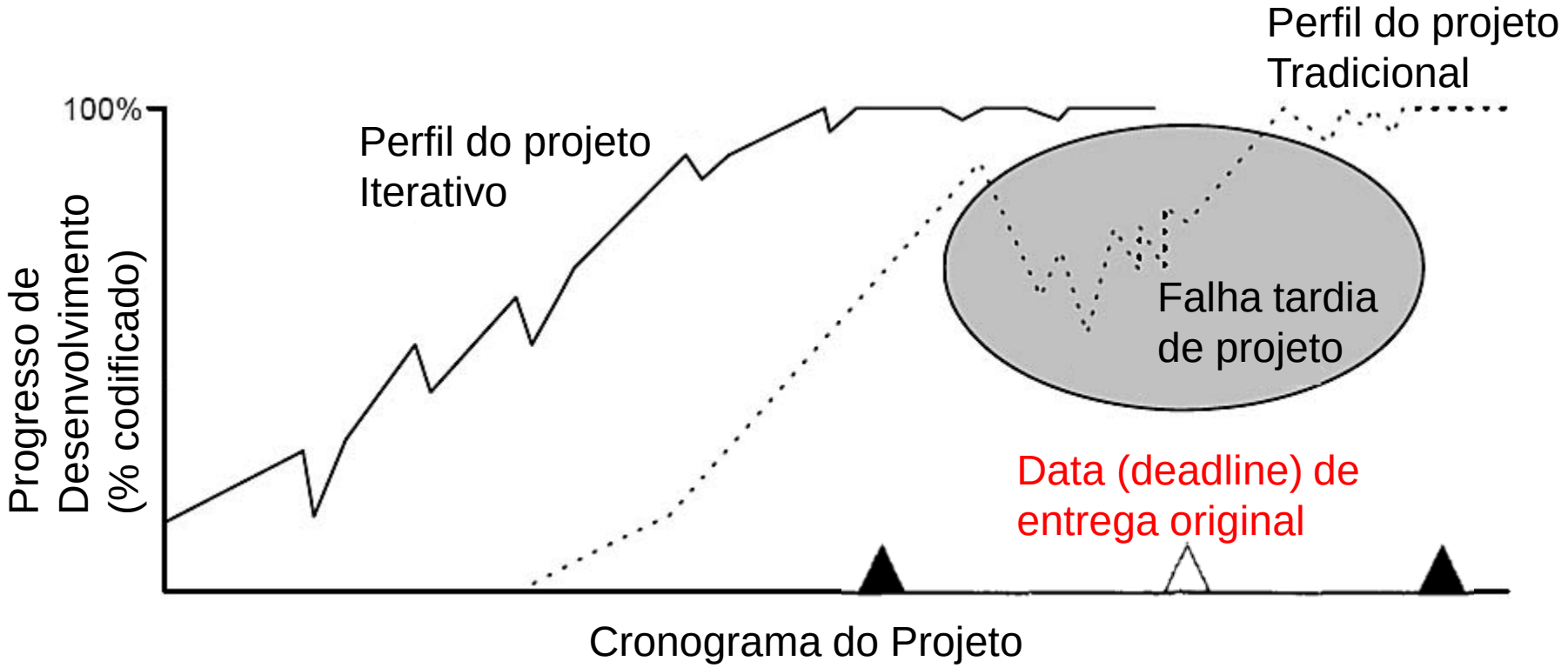
- Iterativo: esboços ou pedaços incompletos do produto são adicionados a cada iteração

Modelos Iterativos

Desenvolvimento Incremental

- Vantagens
  - O cliente pode receber e avaliar as entregas do produto mais cedo
  - Os incrementos são avaliados e os desvios identificados, sendo possível replanejar as próximas iterações de acordo
  - O risco geral do projeto fracassar diminui

Desenvolvimento Iterativo x Cascata



Modelos Iterativos

Desenvolvimento em Espiral

- O processo é representado como uma espiral em vez de uma seqüência de atividades
- Cada volta na espiral representa uma fase no processo
- Não há fases fixas, como Especificação, Projeto, etc.
  - Os loops na espiral são escolhidos dependendo do que for necessário

Modelos Iterativos

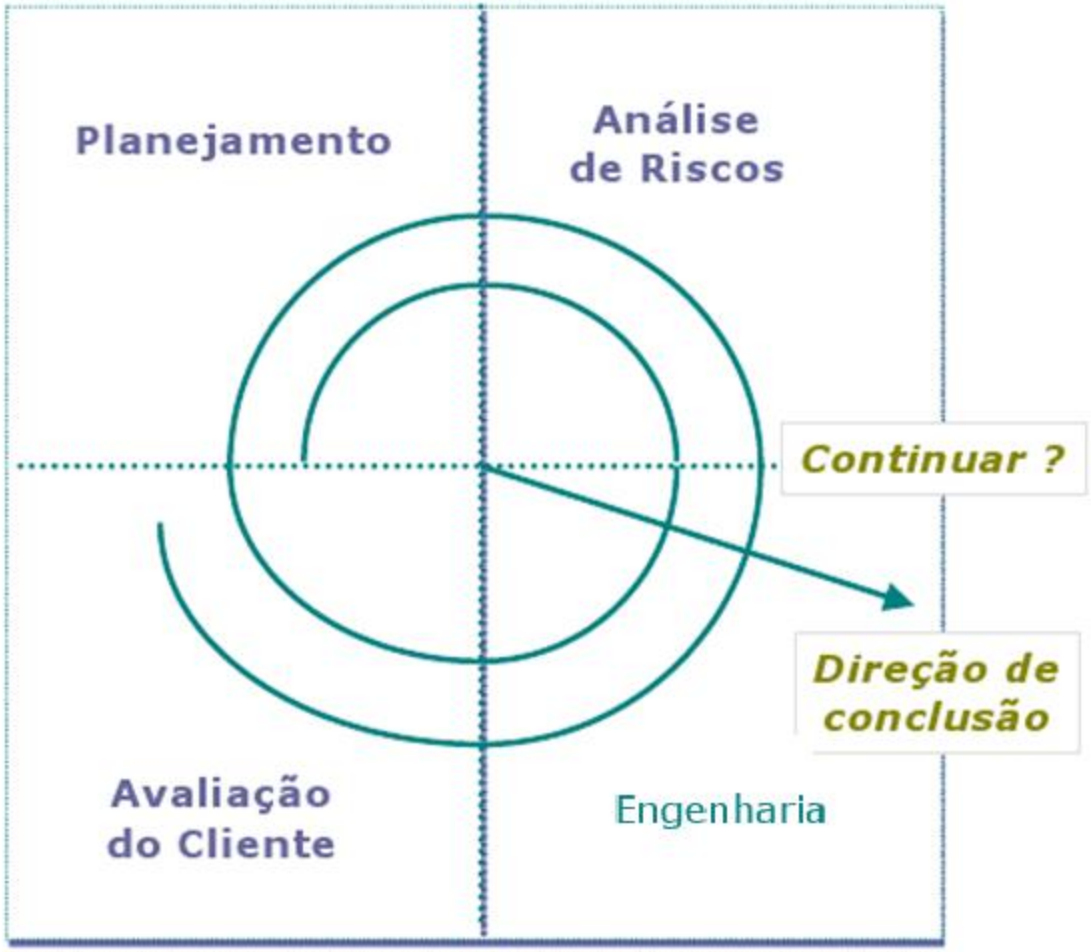
Desenvolvimento em Espiral

- Acrescenta aspectos gerenciais ao desenvolvimento de software
  - Planejamento, tomada de decisão
  - Análise de Riscos

Porém, é complexo e requer experiência na avaliação de riscos!

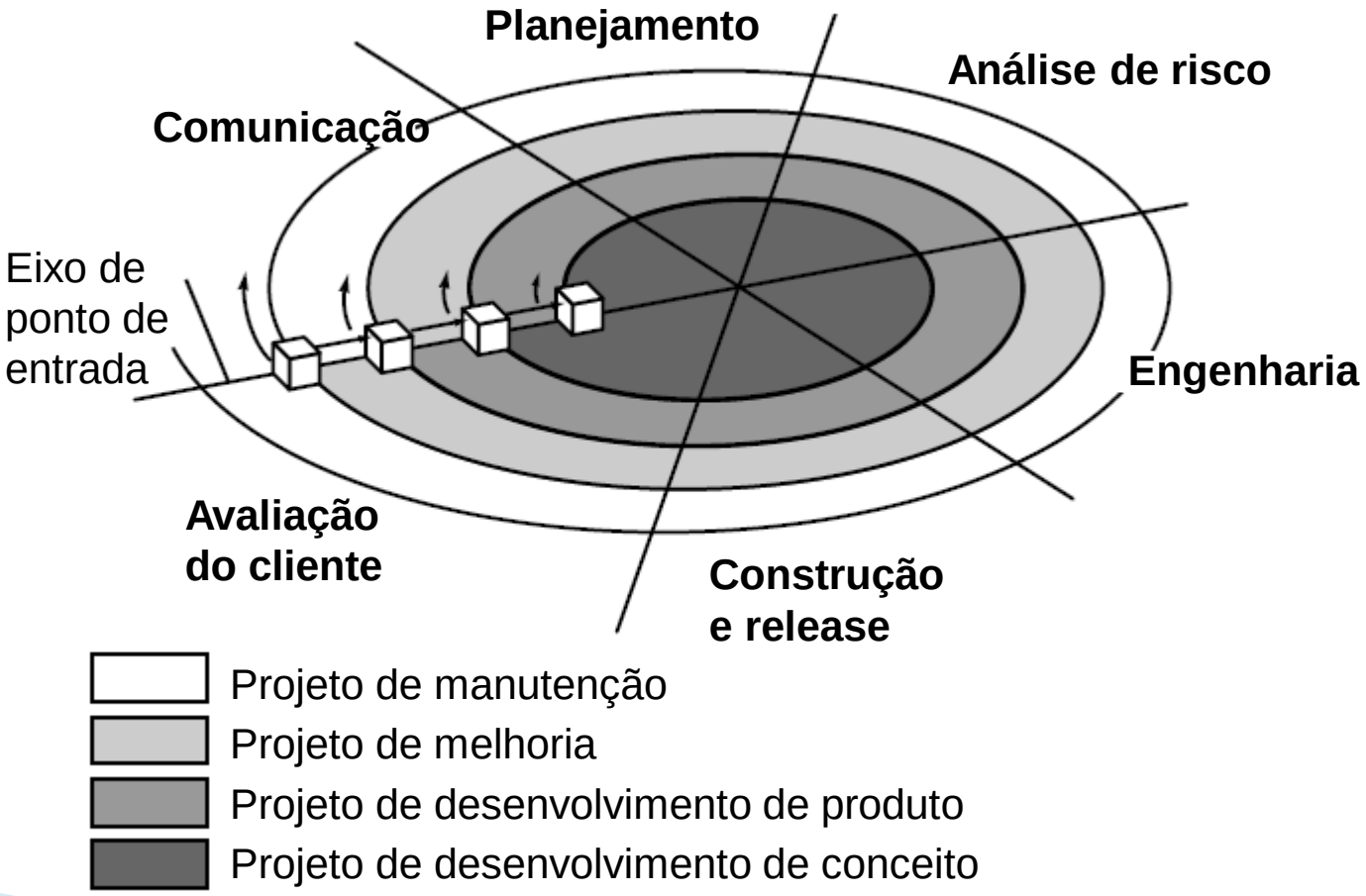
Modelos Iterativos

Desenvolvimento em Espiral [Boehm]



Modelos Iterativos

Desenvolvimento em Espiral [Pressman]



Exercícios [4]

(TST – CESPE 2008)

[94] O modelo de desenvolvimento em espiral permite repensar o planejamento diversas vezes durante o desenrolar do projeto.

(SERPRO – CESPE 2008)

[64] O modelo iterativo e o modelo em espiral possuem características semelhantes: ambos permitem que as atividades do processo sejam planejadas e avaliadas ao longo do ciclo de vida.

(IJSN – CESPE 2010)

[56] Uma vantagem do ciclo de desenvolvimento iterativo em relação ao ciclo clássico está na receptividade às mudanças inerentes ao desenvolvimento de software.

Exercícios [4]

(SAD/PE – CESPE 2010 – adaptada)

[58] Imediatamente após ter testado um protótipo evolucionário, um dos colegas da empresa iniciou a produção de uma lista de riscos aos quais o projeto está sujeito. Dessa forma, a empresa não utiliza um modelo de ciclo de vida embasado no espiral.

(SERPRO – CESPE 2010)

[69] O modelo espiral do ciclo de vida de software é iterativo e incremental, uma vez que a mesma sequência de atividades relacionadas à produção de software é realizada a cada ciclo da espiral.

Metodologias Ágeis

Bibliografia

- **Beck,** Kent. Extreme Programming Explained – Embrace Change. **Editora**: Addison-Wesley.
- **Schwaber,** Ken. Scrum Guide. **Disponível em:** www.scrum.org

Contexto

- Motivação: insatisfação com os excessos dos métodos tradicionais, considerados “pesados”
- Métodos ágeis buscam flexibilizar o desenvolvimento de software
  - Focam no código, e não no projeto
  - São baseados em abordagens iterativas
  - Têm o objetivo de entregar software funcionando o mais rápido possível

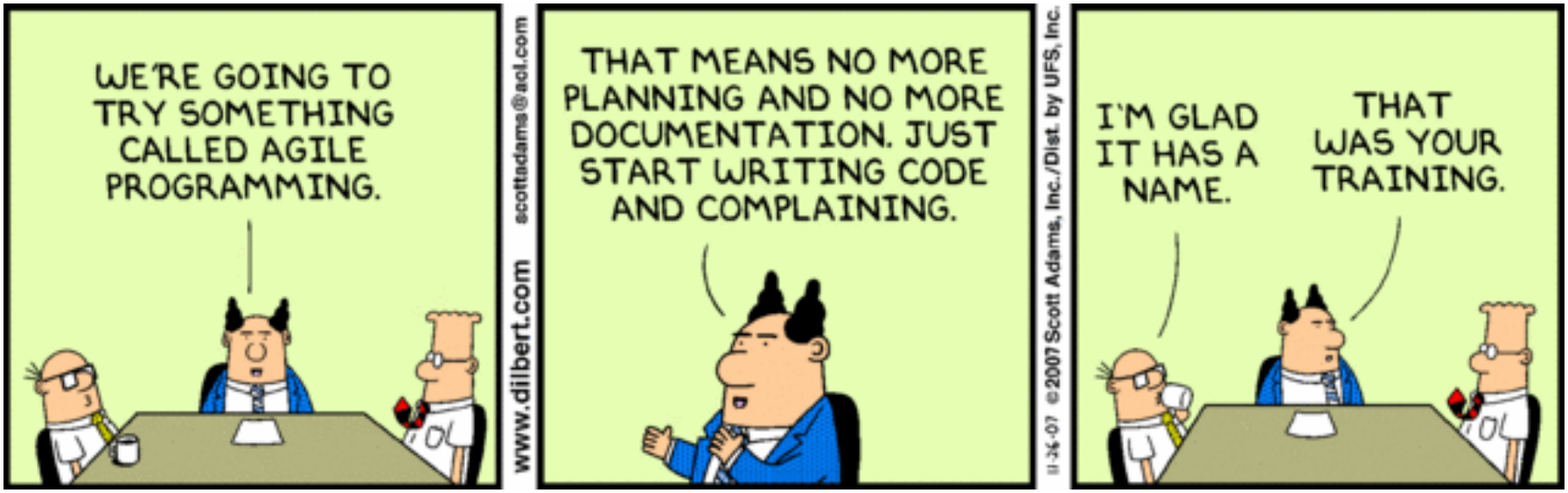
Manifesto Ágil (2001)

"Estamos evidenciando maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a entender que:

- **Indivíduos e interações** são mais importantes que processos e ferramentas.
- **Software funcionando** é mais importante do que documentação completa e detalhada.
- **Colaboração com o cliente** é mais importante do que negociação de contratos.
- **Adaptação a mudanças** é mais importante do que seguir o plano inicial.

Ou seja, mesmo tendo valor os itens à direita, valorizamos mais os itens à esquerda. "

Mito sobre Desenvolvimento Ágil...



Errado - Metodologia Ágil não é o caos!

Principais modelos

- Métodos ágeis, hoje, são considerados uma “família”
  - **eXtreme Programming**
  - **Scrum**
  - **FDD**
  - Lean Software Development
  - Crystal Family
  - (...)

Extreme Programming (XP)

Surgimento

- Em meados de 1990, **Kent Beck**, procurou formas mais simples e eficientes de desenvolver Software.
- Em Março de 1996, ele iniciou um projeto com novos conceitos que resultaram na metodologia XP – *eXtreme Programming*

O que é XP?

"Trata-se de uma metodologia ágil para equipes pequenas e médias desenvolvendo software com requisitos vagos e em constante mudança" – Kent Beck

O que é programar ao extremo?

- Levar todas as boas práticas ao **extremo**
  - Se revisar código é bom, vamos revisá-lo toda hora (***pair programming***)
  - Se testar é bom, vamos testar toda hora (**testes funcionais**)
  - Se projetar é bom, vamos fazer disso parte do trabalho diário de cada pessoa (***refactoring***)
  - Se integrar é bom, vamos integrar a maior quantidade de vezes possível (**integração contínua**)
  - Se simplicidade é bom, vamos deixar o sistema na forma mais simples possível (**projeto simples**)

Práticas do XP

- **Metáfora**
  - Uma história que todos – programadores, clientes e gerentes – podem contar acerca de como funciona o sistema
  - Facilita a comunicação entre os interessados
- **Projeto simples**
  - O código está, a qualquer momento, na forma mais simples que passe todos os testes
- **Pequenas versões**
  - As entregas são feitas através de pequenos releases (pedaços) de software **funcionando**
  - Dá confiança ao cliente sobre o progresso geral

Práticas do XP

- **Refatoração**
  - O código deve ser constantemente melhorado, tornando-o mais simples e mais genérico, removendo redundâncias e duplicidades
- **Programação em pares**
  - Os programadores trabalham em pares, checando (validando) mutuamente o trabalho feito
  - Mesma máquina, mesmo mouse, mesmo monitor
- **Propriedade coletiva do código**
  - Todos são responsáveis por todo o código e qualquer pessoa está autorizada a realizar mudanças nele

Práticas do XP

- **Padrão de codificação**
  - Todo código é desenvolvido de acordo com um estilo e formato consistentes (padrão)
- **Ritmo sustentável**
  - Cada programador trabalha 40 horas por semana, no máximo
- **Reuniões em pé**
  - Reuniões rápidas e diárias com a equipe, para discutir apenas o essencial
- **Cliente sempre presente**
  - O cliente, com conhecimento sobre o negócio, deve estar disponível em tempo integral para a equipe

Práticas do XP

- **Desenvolvimento Orientado a Testes**
  - Uma estrutura de testes unitários automatizada é criada e os testes são escritos antes mesmo das funcionalidades serem implementadas
- **Integração Contínua**
  - Os diversos módulos do software são integrados o mais cedo possível, para evitar problemas de integração no futuro
- **Planejamento Incremental**
  - Requisitos são registrados como Estórias dos Usuários e priorizados para serem incluídos em uma determinada iteração

Valores do XP

- **Comunicação**
  - Métodos para rapidamente construir e disseminar conhecimento
- **Simplicidade**
  - XP encoraja que você comece, sempre, pela solução mais simples que funcione
- **Feedback**
  - Do cliente, do sistema e da equipe
- **Coragem**
  - Design simples, refatoração…
- **Respeito**
  - Respeito da Equipe, do Cliente, dos Usuários…

Exercícios [5]

(TRE/BA – CESPE 2010)
[109] Em XP, a prática denominada programação em pares (pair programming) é realizada por um desenvolvedor em dois computadores, com o objetivo de aumentar a produtividade.

(SERPRO – CESPE 2010)
[70] Metodologias ágeis como a XP enfatizam a documentação de software no próprio código, que deve ser escrito por meio de ferramentas CASE voltadas ao desenvolvimento rápido de aplicações

(ANAC – CESPE 2009)
[115] A técnica conhecida como refactoring é constantemente aplicada no desenvolvimento baseado no método ágil extreme programming.

Exercícios [5]

(ANAC – CESPE 2009)

[116] No modelo extreme programming, os testes de software só são realizados na etapa, final de desenvolvimento do software e, somente nessa etapa, os programadores trabalham, obrigatoriamente, em pares, utilizando cada um o próprio computador.

Exercícios [5]

(BNDES – CESGRANRIO 2009)

[47] Determinado projeto de software utiliza XP (eXtreme Programming) como metodologia de desenvolvimento. A esse respeito, é **INCORRETO** afirmar que

a) o cliente participa ativamente e acompanha os passos dos desenvolvedores diariamente.
b) os integrantes da equipe se reúnem rapidamente no início do dia, de preferência em pé.
c) a equipe de desenvolvimento concentra esforços naquilo que gera maior valor para o cliente.
d) a programação em pares dispensa o desenvolvimento orientado a testes no projeto.
e) as funcionalidades do software são descritas em histórias, da forma mais simples possível

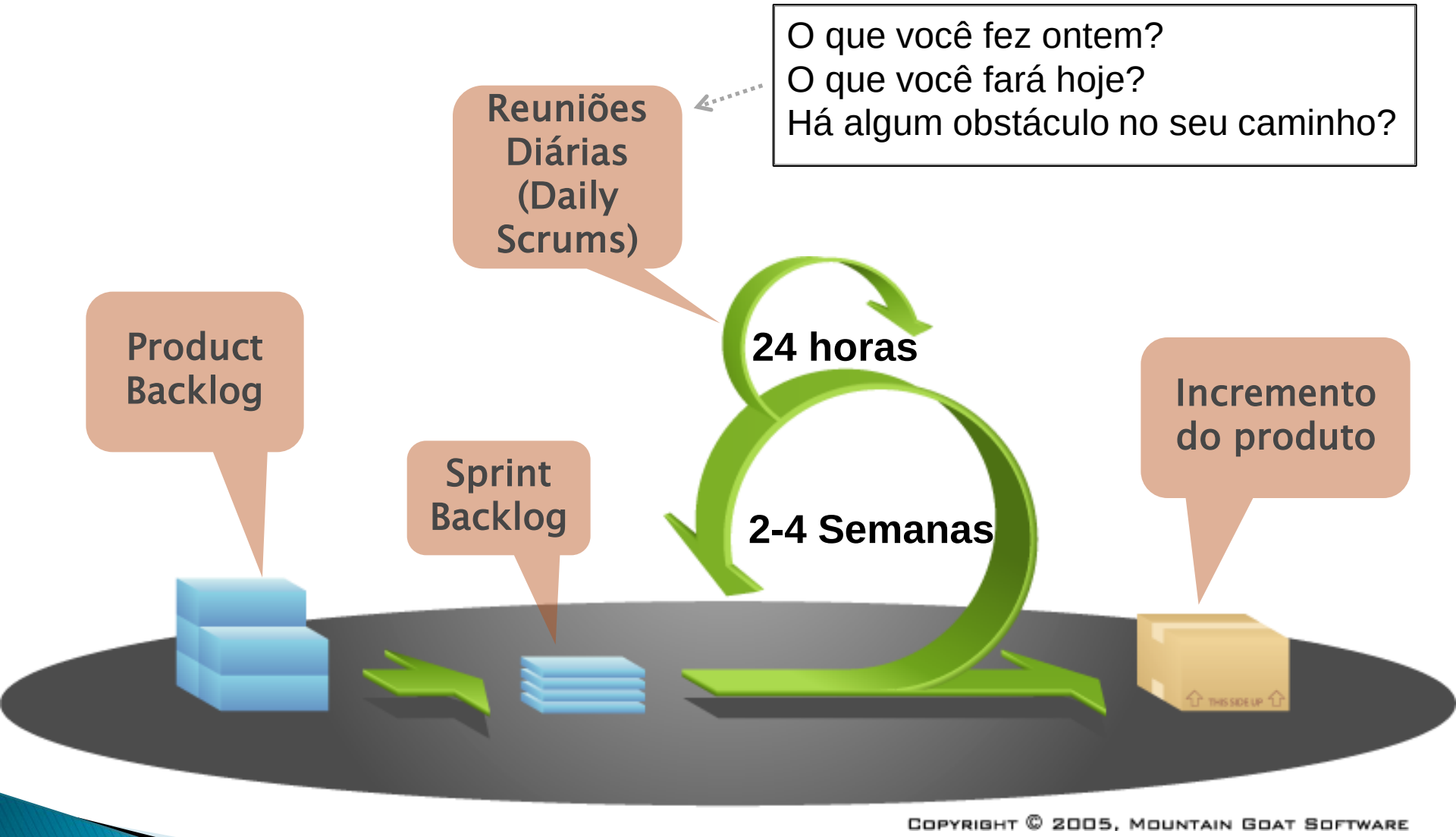
Scrum

- O nome é derivado de uma atividade que acontece em um jogo de Rugby
- É um framework de processo dentro do qual podem ser empregados processos e técnicas variadas
  - É possível adicionar papeis, artefatos, atividades e “cerimônias” de acordo com a sua necessidade
- Scrum pode ser aplicado em qualquer contexto no qual um grupo de pessoas trabalhe junto para atingir algum objetivo

Características

- As equipes se auto-organizam para maximizar a comunicação e diminuir a supervisão
- O produto evolui em uma série de “sprints”
- Os requisitos (funcionalidades dos clientes) são listados em um “product backlog”
- Não há prática de engenharia prescrita (Scrum se adapta a todas elas)
  - Scrum é um processo essencialmente **gerencial**

Visão Geral


O que você fez ontem?
O que você fará hoje?
Há algum obstáculo no seu caminho?
Reuniões Diárias (Daily Scrums)
24 horas
Product Backlog
Sprint Backlog
2-4 Semanas
Incremento do produto
Copyright © 2005, Mountain Goat Software

Artefatos

Product Backlog

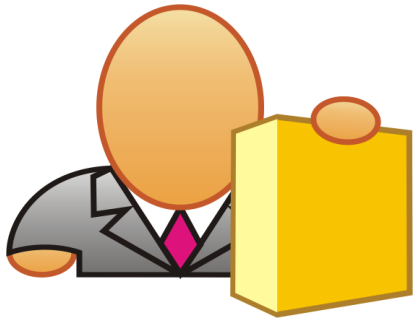
- Uma lista ordenada de tudo o que é necessário no produto
- Idealmente, cada item deve ter seu peso (prioridade) de acordo com a vontade do cliente
- É replanejado (repriorizado) no início de cada Sprint

Artefatos

Sprint Backlog

- Uma lista de tarefas que a equipe se compromete a completar dentro de uma determinada Sprint
- Os itens são derivados a partir do Product Backlog
- São considerados
  - A prioridade que o cliente deu aos itens
  - O tempo e esforço estimados pela equipe para completar os vários itens

Papéis



Product Owner

- Define as funcionalidades do produto
- Decide as datas de lançamento e conteúdo
- Prioriza as funcionalidades de acordo com o valor para a empresa
- Aceita ou rejeita os resultados dos trabalhos

Papéis

Team

- Contém tipicamente entre 5 e 9 pessoas
- Multi-funcional
  - Programadores, testadores, desenvolvedores de interfaces, etc.
- Dedicação integral
  - Raras Exceções (ex: DBA)
- Auto-organizável (sem títulos)
- Trocas só na mudança de Sprints

Papéis

Scrum Master

- Responsável pela aplicação dos valores e práticas Scrum
- Remove obstáculos, facilita resultados
- Garante a plena funcionalidade e produtividade da equipe
- Escudo para interferência externas

Papéis

Scrum Master

- Obstáculos a serem removidos
  - “O meu ___ quebrou e eu preciso de um novo”
  - “Eu preciso para debugar um problema no ___”
  - “Eu preciso de ajuda para aprender ___”
  - “O cliente ___ não teve tempo de se reunir conosco no planejamento e por isto estou parado”
  - “O presidente da empresa pediu para eu resolver um problema para ele em outro projeto, por um dia ou dois…”

Eventos

- **Planejamento da Sprint**
  - Selecionam-se itens do Product Backlog, e as tarefas são identificadas e estimadas
  - De forma colaborativa, não apenas feito pelo Scrum Master
- **Reuniões Diárias (Daily Scrums)**
  - Apenas os membros da equipe, todos os dias, em pé, durante 15 minutos
- **Revisão do Sprint**
  - Apresentação dos resultados obtidos
  - Todo o time participa, informalmente, 2 horas de preparação, no máximo, sem slides

Eventos

- **Retrospectiva da Sprint**
  - Ocorre após a revisão da sprint e antes da próxima reunião de planejamento
  - Inspeciona como foi a última Sprint em termos de:
    - Pessoas e Relações
    - Processos e Ferramentas
  - Enquanto a revisão da sprint analisa o produto, a retrospectiva analisa o processo

Exercícios [6]

(BASA – CESPE 2010)

[78] O Scrum é utilizado, como função primária, para o gerenciamento de projetos de desenvolvimento de software, mas também tem sido usado como extreme programming e outras metodologias de desenvolvimento. Teoricamente, o Scrum pode ser aplicado em qualquer contexto no qual um grupo de pessoas necessite trabalhar juntas para atingir um objetivo comum.

(SERPRO – CESPE 2010)

[54] Para que o plano do projeto de um processo seja efetivo, uma das premissas é que o product backlog esteja completo.

Exercícios [6]

(TRE/BA - CESPE 2010)

[68] Um princípio chave do Scrum é o reconhecimento de que desafios fundamentalmente empíricos não podem ser resolvidos com sucesso utilizando-se uma abordagem tradicional de controle. O Scrum adota uma abordagem empírica, aceitando que o problema não pode ser totalmente entendido ou definido, focando na maximização da habilidade da equipe de responder de forma ágil aos desafios emergentes.

[101] A metodologia Scrum é facilitada por um scrum master, que atua como um mediador entre a equipe e qualquer influência desestabilizadora, além de assegurar que a equipe esteja utilizando corretamente as práticas de Scrum, motivando e mantendo o foco na meta da sprint.

Feature Driven Development

- Início: Jeff de Luca e Peter Coad em 1997 – projeto para um banco de Singapura
- A FDD é focada na entrega regular de **funcionalidades** valiosas para o cliente
- Tem mais estrutura do que o XP
  - As equipes podem variar de 10 a 250 programadores (aproximadamente)
- Porém é mais enxuto do que o RUP
  - Não necessita de tanta documentação, nem é tão detalhado

Feature Driven Development

Papéis

- Project Manager
  - O líder administrativo do projeto
  - Planeja orçamento, elabora relatórios e protege a equipe de distrações externas
- Chief Architect
  - Responsável pelo projeto geral do sistema
  - Tem a palavra técnica final sobre o software e sua arquitetura

Feature Driven Development

Papéis

- Development Manager
  - Lidera as atividades diárias de desenvolvimento
  - Lidera a equipe de desenvolvimento através de desafios técnicos
- Chief Programmers
  - Desenvolvedores experientes
  - Lideram pequenas equipes de desenvolvedores individuais

Feature Driven Development

Papéis

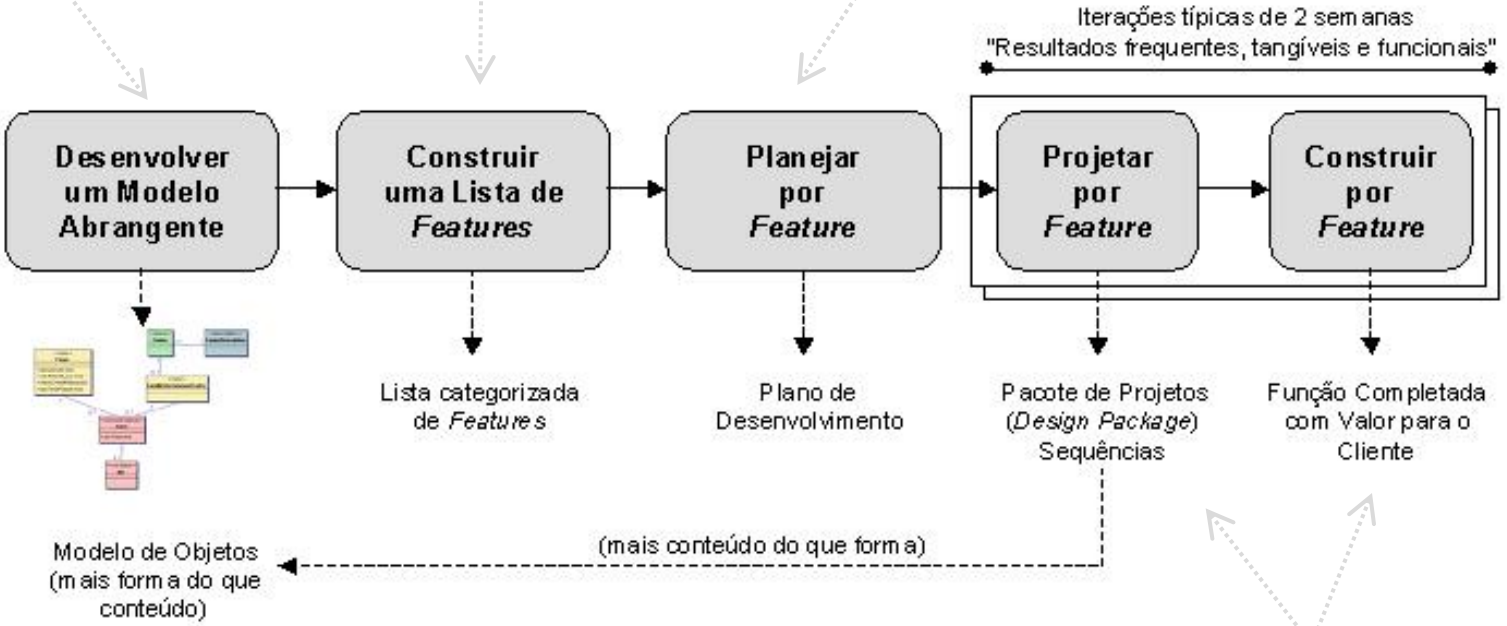
- Class Owners
  - São os desenvolvedores individuais
  - Projetam, codificam, testam e documentam as funcionalidades
- Domain Experts
  - Usuários, clientes, patrocinadores, etc.
  - A base de conhecimento para os desenvolvedores

Processos

Entendimento do domínio e estabelecimento do escopo

Funcionalidades que podem ser implementadas em até 2 semanas

Quem implementará a funcionalidade? Qual é a sequência de implementação?



Projeto e Codificação (tarefas de engenharia)

Exercícios [7]

(DPE/SP – FCC 2010)

[64] Na engenharia de software, um processo iterativo denominado sprint, que segue o ciclo PDCA para entregar, num período de 30 dias aproximadamente, um incremento do software pronto, caracteriza a metodologia ágil

(A) SCRUM.
(B) DSDM.
(C) Crystal.
(D) FDD.
(E) XP.

Exercícios [7]

(TRF4 – FCC 2010)
[61] A Feature Driven Development (FDD) é uma metodologia ágil de desenvolvimento de software, sobre a qual é correto afirmar:

a) Não pode ser combinada a outras técnicas para a produção de sistemas.
b) Possui cinco processos: Desenvolver um Modelo Abrangente, Construir a Lista de Funcionalidades, Planejar por Funcionalidade, Detalhar por Funcionalidade e Implementar por Funcionalidade.
c) Divide os papéis em dois grupos: papéis chave e papéis de apoio. Dentro de cada categoria, os papéis são atribuídos a um único participante que assume a responsabilidade pelo papel.
d) Mantém seu foco apenas na fase de modelagem.
e) Mantém seu foco apenas na fase de implementação.

Exercícios [7]

(TJ/PE – FCC 2007)
[36] Considere:
I. Desenvolvimento de um modelo geral.
II. Construção da lista de funcionalidades.
III. Plano de liberações com base nas funcionalidades a implementar.
IV. Projetar com base nas funcionalidades.
V. Implementar com base nas funcionalidades.

São fases de projetos que seguem o processo projetado por Peter Coad, Erich Lefebvre e Jeff De Luca chamado de

(A) MDA
(B) XP
(C) FDD
(D) RUP
(E) MVC

Ferramentas CASE

Um breve histórico...

- Antes da década de 90 – "*casa de ferreiro, espeto de pau*"
- Hoje em dia as ferramentas CASE ainda não são tão variadas nem fornecem tudo aquilo que os desenvolvedores queriam, mas **são um aparato essencial para o engenheiro de software**

CASE – Visão Geral [Pressman]

- **O que são?**
  - São ferramentas que auxiliam o engenheiro de SW em cada atividade associada ao desenvolvimento de SW
- **Quem usa?**
  - Gerentes de projeto e engenheiros de SW
- **Por que são importantes?**
  - Reduzem o esforço necessário para produzir artefatos e alcançar metas
  - Aumentam a qualidade do software

CASE – Visão Geral [Pressman]

- **Quais são os passos?**
  - Ferramentas CASE são usadas em conjunto com o modelo de processo adotado. Se for escolhida uma ferramenta completa, pode passar por quase todos os passos do desenvolvimento de SW

CASE – Visão Geral [Pressman]

- **Como são usadas?**
  - Como complemento às boas práticas de Engenharia de Software. Ferramentas CASE **não** substituem uma metodologia de desenvolvimento de software sólida

  "Um tolo com ferramentas, ainda é apenas um tolo"

CASE – Categorização

- **Horizontais**
  - São utilizados durante todo o processo de desenvolvimento de software
- **Verticais**
  - São específicas para uma disciplina de software
- Por funções [Pressman]
  - Processos de negócio, Planejamento de projeto, Análise de Riscos, Rastreamento de Requisitos, IDEs, Gerenciamento de BDs, Análise Estática, Análise Dinâmica, ...

CASE – Categorização

- Como não há um padrão para categorizar ferramentas CASE, a seguinte proposta foi feita:
  - **Front-end ou Upper CASE**
    - apóiam as etapas iniciais da criação dos sistemas: as fases de planejamento, análise e projeto da aplicação
  - **Back-end ou Lower CASE**
    - dão apoio à parte física, i.e, código, testes e manutenção
  - **I-CASE ou Integrated CASE**
    - cobrem todo o ciclo de vida, do início ao fim

Exercícios [8]

(TRT5 – CESPE 2009)

[76] As ferramentas CASE podem ser verticais ou horizontais. As primeiras oferecem serviços utilizados durante todo o processo de software, enquanto as segundas são utilizadas em fases específicas do processo de software.

(TJ/CE – CESPE 2008)

[53] Uma ferramenta CASE pode ser utilizada durante todo o projeto de um software ou apenas em fases específicas do projeto.

(MDS – CESPE 2009)

[118] Uma ferramenta CASE (computer-aided software/system engineering) integrada, também chamada de I-CASE,permite a transferência de informação, como modelos, programas e documentos, de uma ferramenta para outra. Entretanto, uma I-CASE não permite a mudança de um estágio do processo de engenharia de software para outro.

Gabarito dos Exercícios

- [0] 61 C, 12 E
- [1] 93 E, 63 C, 85 E, 86 C, 58 C
- [2] 41 C, 73 C, 70 E
- [3] 65 E
- [4] 94 C, 64 C, 56 C, 58 E, 69 E
- [5] 109 E, 70 E, 115 C, 116 E, 47 D
- [6] 78 C, 54 E, 68 C, 101 C
- [7] 64 A, 61 B, 36 C
- [8] 76 E, 53 C, 118 E

FIM