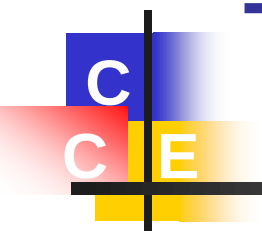


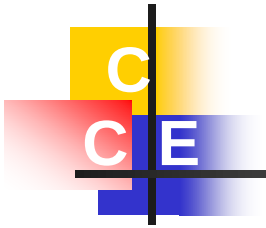
Tópico 1 - Fundamentação

Luiz Antônio M. Pereira

lpereira@luizantoniopereira.com.br

Conteúdo

- Software
- Qualidade de Software
- Engenharia de Software
- Processos de Software



Definição

- Software
  - Produtos projetados e construídos pelos Engenheiros de Software.
- Engenharia de Software
  - Estudo e aplicação de procedimentos sistemáticos, disciplinados e quantificáveis ao desenvolvimento, operação e manutenção de software (IEEE).

Engª. S/W - Motivação

- Maior dependência com relação a software e menos espaço para improvisações.
  - Desenvolver software precisa ser algo organizado, controlado, previsível...
- Há uma relação direta entre a qualidade e confiabilidade de um **produto** e a qualidade e confiabilidade do seu **processo** de produção...
  - ... Há uma chance bem maior de que uma boa solução surja em um ambiente organizado.

Desenvolvendo Software...

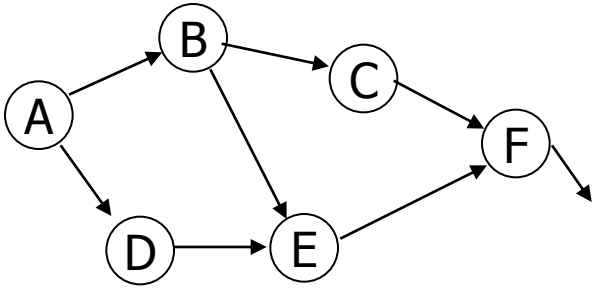
- ...Idealmente através de *processos* bem comportados, envolvendo:
  - Etapas do processo,
  - Níveis de qualidade,
  - Custos e
  - Prazos

Definidos *a priori*.

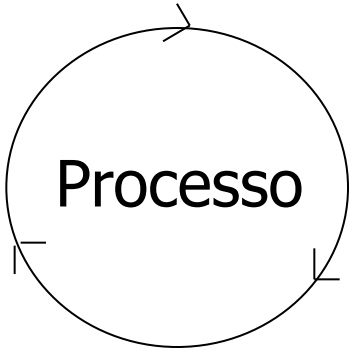
Processo de Software

Envolve:

Pessoas com habilidades, treinamento e motivação



Procedimentos e métodos definindo o relacionamento entre as tarefas



Ferramentas e equipamentos adequados

Processo de Software

- Possui uma série de tarefas definidas.
- Tarefas são organizadas de formas diferentes, de acordo com o *modelo de processo* adotado.
- Tarefas demandam marcos para verificação, documentação e garantia da qualidade do que produzem.

Qualidade em *Software*

- Ficar em conformidade com:
  - Requisitos funcionais e de desempenho explicitamente estabelecidos (são as bases para a medição da qualidade);
  - Padrões de desenvolvimento explicitamente documentados (definem um conjunto de critérios que guiam o processo de desenvolvimento);
  - Critérios de usabilidade e confiabilidade...

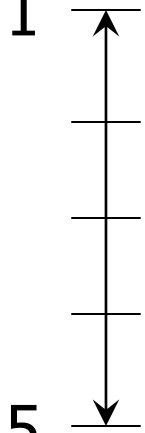
Qualidade em *Software*

- Diversos Modelos
  - ISO (IEC e 9000)
  - SPICE
  - (S/W) CMM
  - ...

Qualidade em *Software*

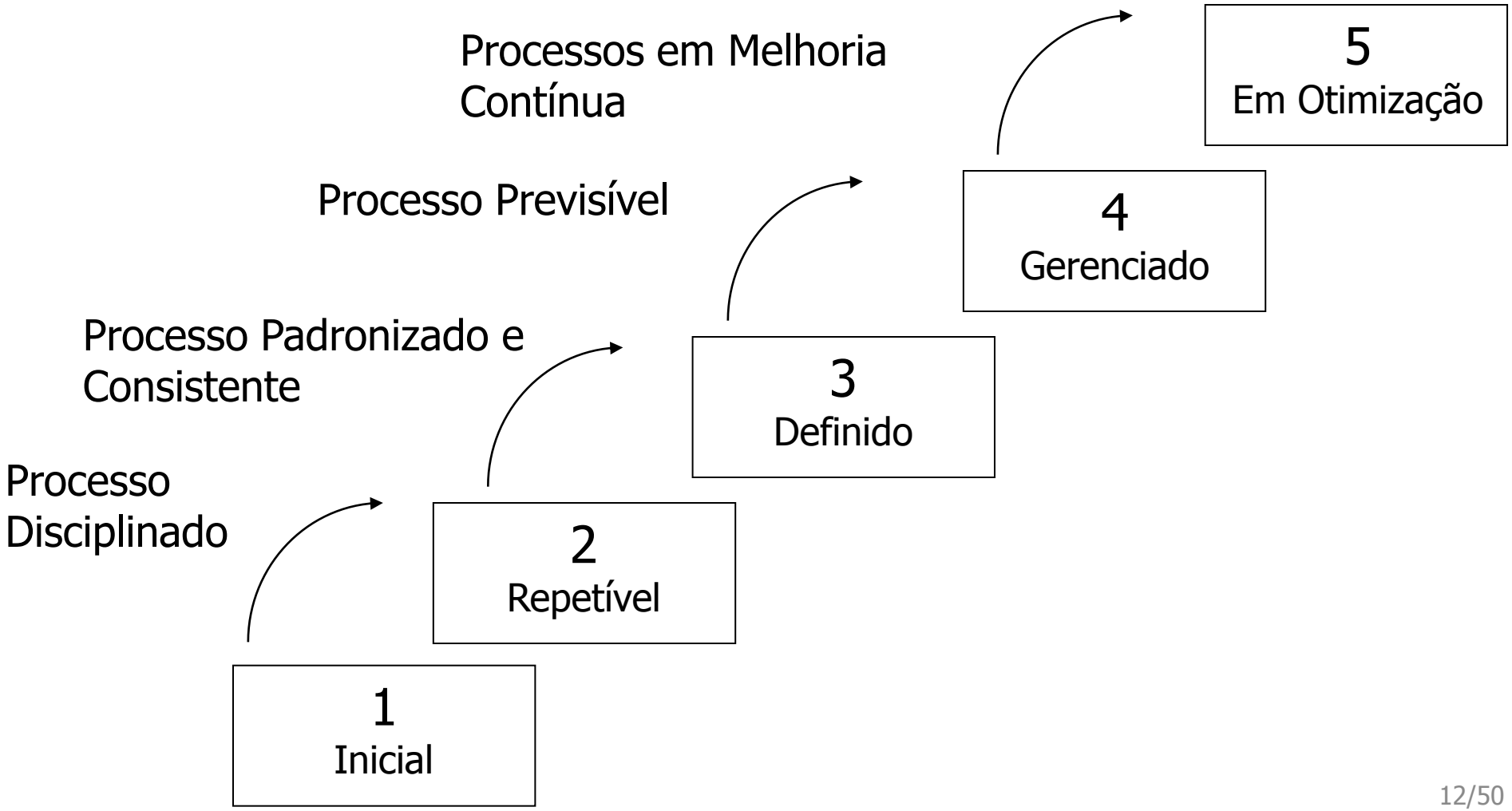
- No SW-CMM:
  - A maturidade é medida em níveis;
  - Os níveis de maturidade estabelecem o estágio atual e as etapas necessárias para melhoria dos processos de *software*.
  - Os níveis são patamares bem definidos conduzem a processos mais maduros de software;
  - Cada patamar compreende um conjunto de objetivos e comprometimentos (KPAs – *Key Process Areas*) que devem ser satisfeitos.

Qualidade em Software



- Organização imatura: o processo de software é improvisado. Mesmo que o processo seja especificado, ele não é seguido. São organizações reacionárias.
- Organização madura: possui capacidade organizada para gerenciar o desenvolvimento e manutenção de software.

Qualidade em Software

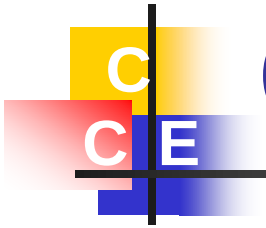


Qualidade em Software

- Níveis:
  - 1 - Inicial
    - Processo *ad-hoc*, ocasional e até caótico. Poucos ou nenhum processo está definido. Sucesso depende do esforço individual.
  - 2 - Repetível
    - Os processos básicos de gerência estão estabelecidos para acompanhamento de custo, prazos e funcionalidade. Existe a capacidade de repetir processos bem sucedidos em projetos anteriores.

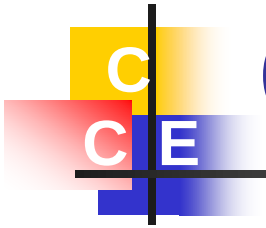
Qualidade em Software

- Níveis (cont.):
  - 3 - Definido
    - O processo de software para as atividades de gerência e engenharia estão documentados, padronizados e integrados no processo de desenvolvimento de software da organização.
    - Todos os projetos usam uma versão documentada e aprovada do processo da organização para desenvolvimento e manutenção. Inclui o nível 2.



Qualidade em Software

- Níveis (cont.):
  - 4 - Gerenciado
    - Medições detalhadas do processo e do produto são coletadas. Ambos são quantitativamente compreendidos e controlados através de métricas minuciosas. Inclui o nível 3.
  - 5 - Em Otimização (ou Otimizante)
    - Um processo contínuo de melhoria baseado nos resultados quantitativos de outros projetos e em testes de novas idéias e tecnologias está estabelecido. Inclui o nível 4.



Qualidade em Software

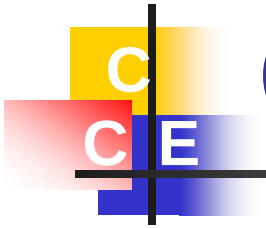
- Exemplo:
  - KPAs para nível 2:
    - Gerência dos Requisitos
    - Planejamento do Projeto de Desenvolvimento
    - Controle do Projeto de Desenvolvimento
    - Gerência da Aquisição de Software
    - Garantia da Qualidade
    - Gerência de Configuração

Ciclo de vida do Software

- Fases Genéricas ()
  - Definição: **o que** fazer para atender às necessidades dos clientes e usuários;
  - Desenvolvimento: **como** fazer;
  - Manutenção: modificar
    - corrigir,
    - adaptar,
    - evoluir,
    - prevenir.

Modelos de Processos

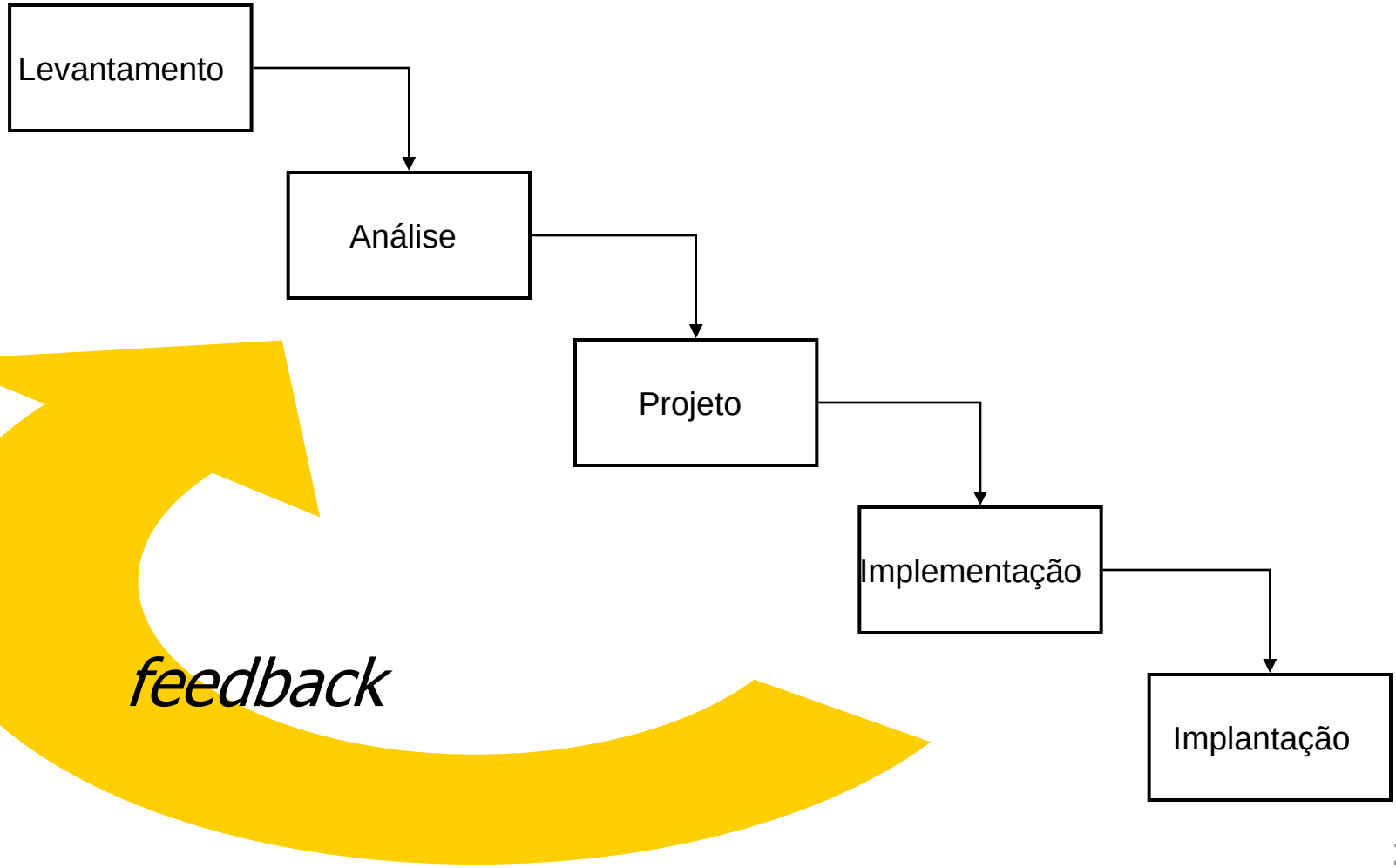
- Cascata (Clássico)
- Prototipação
- Espiral
- Processo Unificado



Cascata

- Modelo linear e sequencial
- Pode usar *feedback* ou não

Cascata - Fases



Cascata - Características

- Possui, em geral, um ciclo longo de desenvolvimento

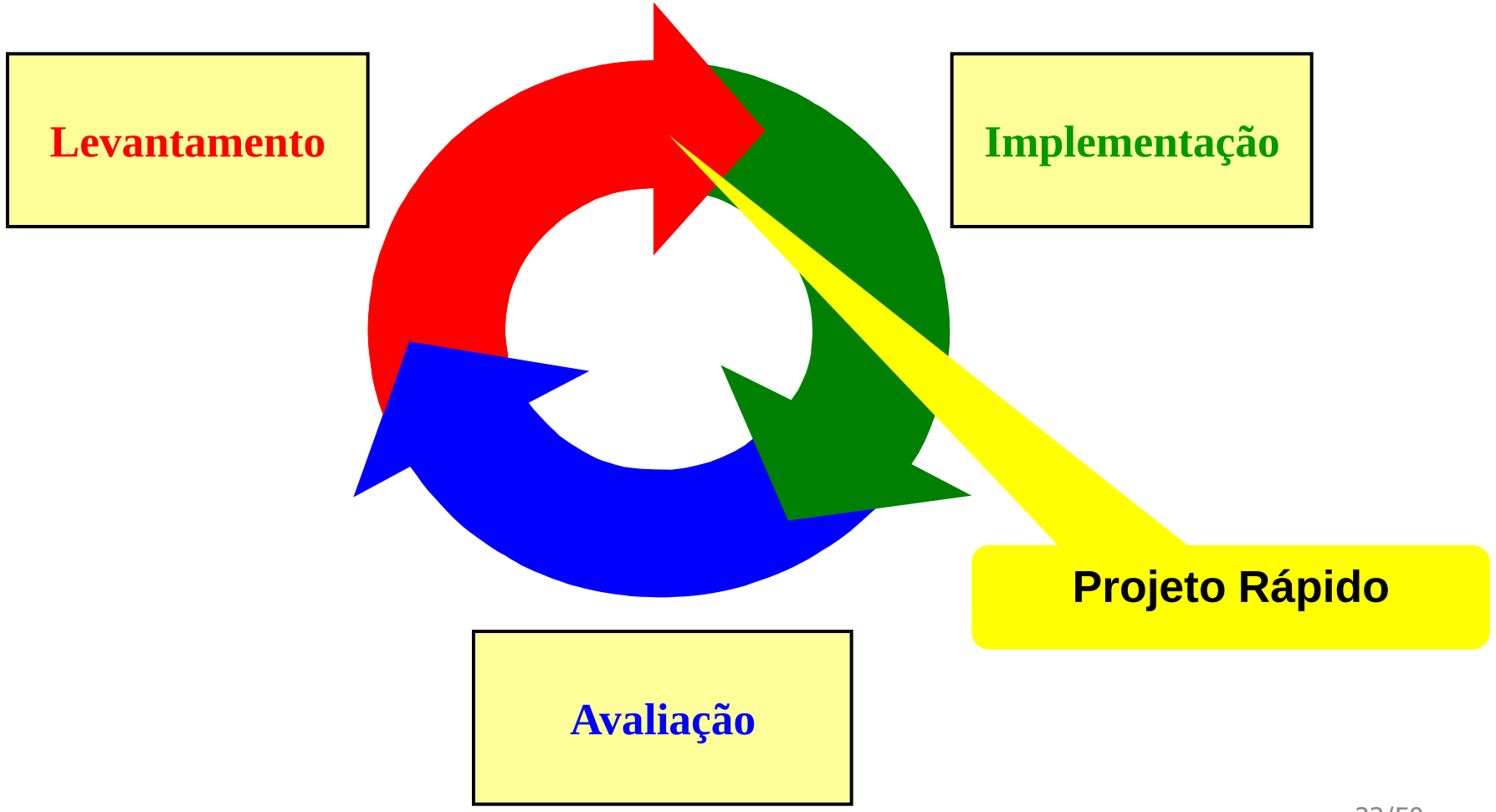
⇨ Aplicável no desenvolvimento de sistemas pouco susceptíveis a mudanças de requisitos e em organizações estáveis

- Serve como base para outros modelos de processo

Prototipação

- Construção de protótipos baseado nas informações do cliente
- Adequado para quando o "negócio" não é bem conhecido ou o cliente não sabe exatamente do que precisa
- Idealmente serve p/ identificar requisitos
- Protótipo = "1o. Sistema" (alguns autores recomendam que o joguemos fora)

Prototipação



Prototipação - Problemas

- O usuário não entende que o software desenvolvido
  - sacrifica a qualidade para obter velocidade no desenvolvimento e
  - não pode ser considerado como um produto que possa entrar em produção.
- O desenvolvedor muitas vezes toma decisões ineficientes de projeto, para facilitar o desenvolvimento, e acaba se acostumando com tais decisões, esquecendo o motivo que o levou a tomá-las.

Modelo Espiral

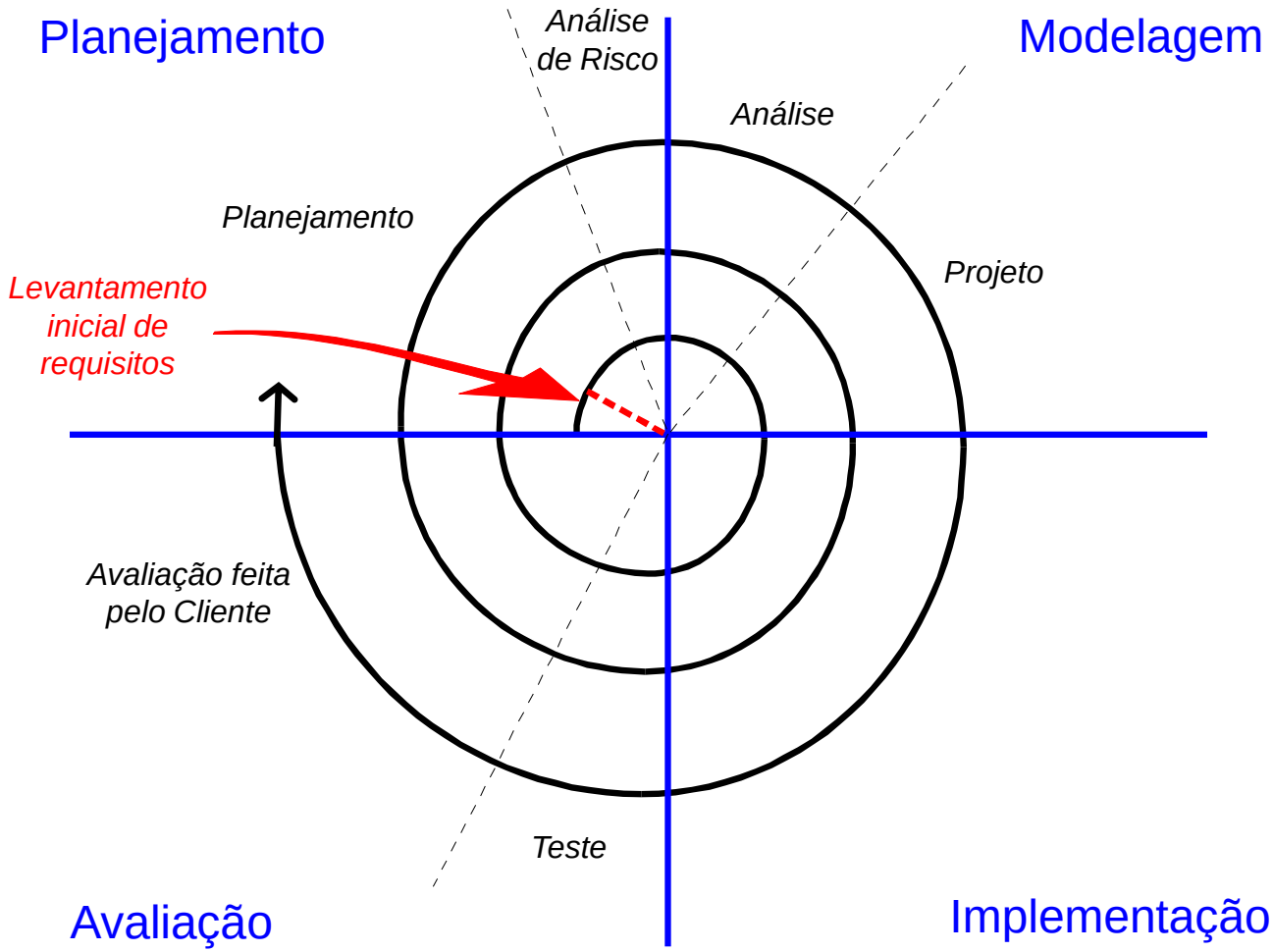
- Inicialmente publicado por Barry Boehm(*)
- Reduz sensivelmente o risco de insucesso de um projeto
- Permite uma maior interação com o cliente
- Adequado à maioria dos tipos de projetos/sistemas

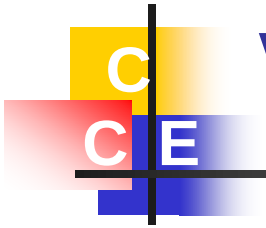
(*)”A Spiral Model for Software Development and Enhancement”, 1988

Espiral - Principais Características

- Incremental, evolutivo, iterativo
- Espiral é dividida em uma série de regiões
- Cada região contempla uma série de tarefas que são adaptadas às características do projeto a ser conduzido
- Um ciclo da espiral pode produzir, tanto uma especificação, como versões do software
- Pode ser adaptado para toda a vida do software

Espiral


Planejamento
Análise de Risco
Modelagem
Análise
Planejamento
Projeto
Levantamento inicial de requisitos
Avaliação feita pelo Cliente
Teste
Avaliação
Implementação



Vantagens do Espiral

- Adaptabilidade a mudanças de requisitos
  (cuidado com a convergência para uma solução final, no T e $ definidos)
- Redução dos riscos
- Acúmulo gradativo de experiência
- Permite a homogeneização da equipe

Espiral - Problemas

- Dificuldade para convencer o cliente de que o processo evolucionário pode ser controlável
- Depende da capacidade de análise de riscos
- Ainda não foi tanto utilizado

Processo Unificado[1] - Características

- Orientado a caso de uso

  Os casos de uso são utilizados como o principal recurso p/ o estabelecimento do comportamento desejado do sistema e p/ sua verificação e validação

- Centrado na Arquitetura

  Os módulos e a organização concebida p/ o novo sistema tem o papel importante no projeto

- Iterativo
  - Gerenciamento de sequências de versões executáveis
  - Divisão do trabalho em mini-projetos que se agregam ao longo do tempo

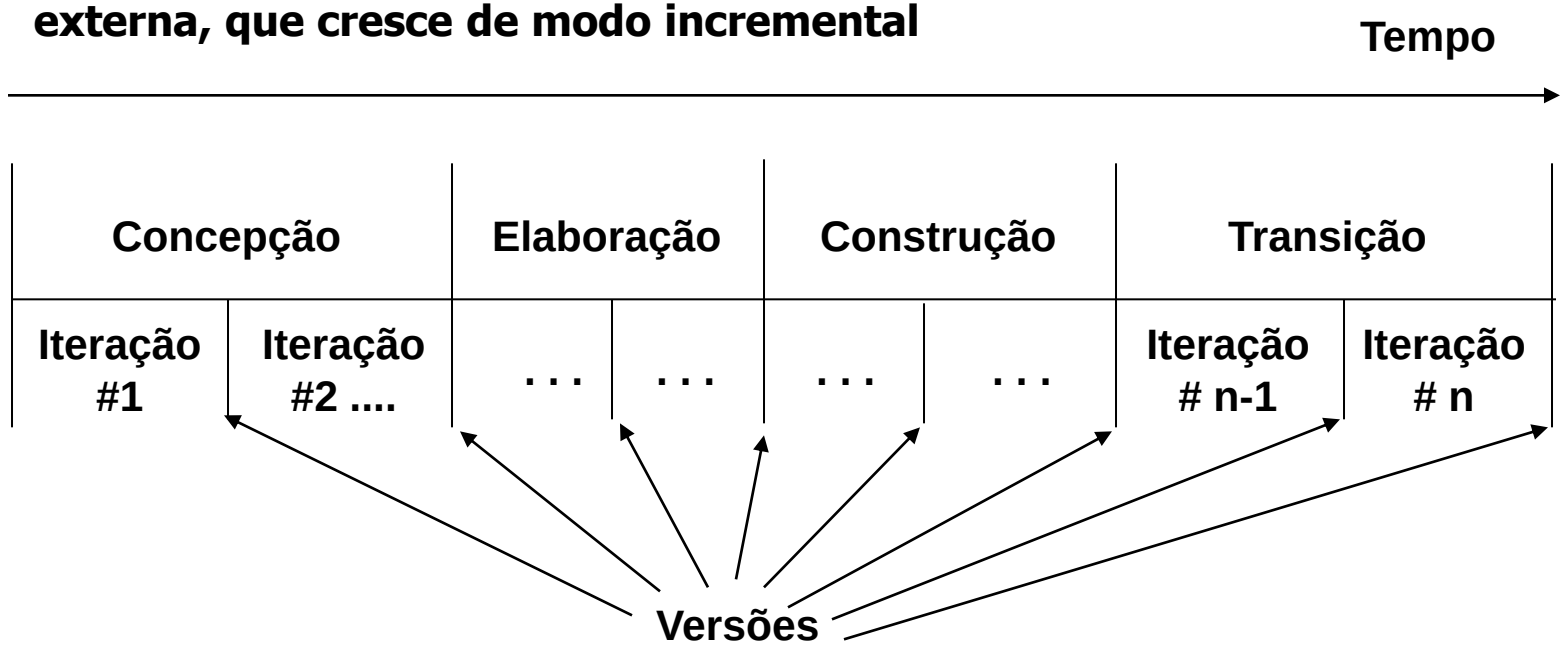
(1) Fonte: "The Unified Software Development Process", Jacobson, Booch & Rumbaugh, Addison-Wesley.

Processo Unificado - Características

- Incremental
  - Cada nova versão incorpora os aprimoramentos incrementais em relação às anteriores.
  - Cada mini-projeto incrementa o produto.

Processo Unificado - Ciclo

- Um ciclo possui 4 fases: concepção, elaboração, construção e transição
- Cada fase pode ser realizada por meio de várias iterações
- Ao final de cada iteração, temos uma versão executável, interna ou externa, que cresce de modo incremental



Concepção - Síntese

- O que o sistema deve fazer?
- Qual poderia ser a sua arquitetura?
- Qual o prazo e custo do desenvolvimento?

Elaboração - Síntese

- Análise do domínio do sistema
- Especificação da maioria dos requisitos
- Estabelecimento da arquitetura através de uma visão macro do sistema
- Detalhamento do plano do projeto
- Eliminação de riscos (pelo menos os de mais alto grau)
- Análise de resultados
- Construção de todos os modelos que não foram contemplados na Concepção

Construção - Síntese

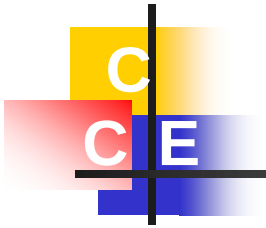
- Requisitos restantes
- Descrição dos critérios de aceitação
- Desenvolvimento iterativo e incremental do produto completo
- Testes
- Avaliação para entrada em produção

Transição - Síntese

- Distribuição/implantação do software
- Software entendido como uma versão beta
- Ajustes e correções (quase sempre)
- Avaliação do produto (software e processo)
  - Permite a aquisição de experiência
  - Indica a possibilidade de outro ciclo

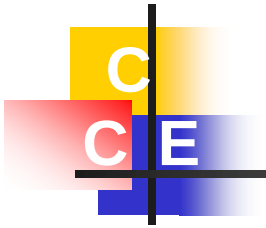
Desenvolvimento Ágil

- Contexto típico de desenvolvimento de S/W:
  - Negócio não é bem conhecido e/ou é dinâmico (necessidade constante de manutenção evolutiva e adaptativa);
  - Documentação externa permanece desatualizada em boa parte do ciclo de desenvolvimento;
  - As manutenções são feitas com base no código (*ninguém* consulta/atualiza a documentação externa);
  - Necessidade de disponibilidade rápida de código;
  - Se há processo, ele é rígido e burocratizado.



Desenvolvimento Ágil

- Classicamente se assume que
  - Usuários especificam exatamente o que querem;
  - Desenvolvedores entregam o que os usuários especificaram, no prazo e custo acertados.
- Mas o que comumente acontece é
  - Usuários não sabem o que querem;
  - Desenvolvedores não entregam o que prometeram;
  - A entrega do software no prazo e custos estabelecidos não é conseguida.
- ⇨ Desenvolvimento de software é arriscado e difícil de ser gerenciado.



Desenvolvimento Ágil

- Necessita-se de novas metodologias que, dentre outras coisas:
  - Tratem adequadamente requisitos vagos e "mutantes";
  - Produzam software de qualidade;
  - Diminuam os encargos da equipe com documentação;
  - Mantenham as expectativas dos usuários atendidas e em níveis realizáveis;
  - Mantenham os projetos gerenciáveis;
  - Tenham base científica.

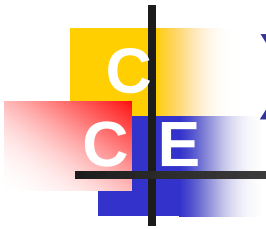
Desenvolvimento Ágil

- Manifesto Ágil (*):
  - Assinado por 17 desenvolvedores (Kent Beck *et al*) em Utah – EUA - em fevereiro de 2001.
  - Descreve a essência de um conjunto de abordagens para desenvolvimento de software criadas ao longo da última década.

(*)http://agilemanifesto.org

Desenvolvimento Ágil

- Manifesto Ágil – Assunções:
  - Indivíduos e interações são mais importantes que processos e ferramentas;
  - Software funcionando é mais importante que documentação completa e detalhada;
  - Colaboração com o cliente é mais importante que (re)negociação de contratos;
  - Adaptação a mudanças é mais importante que seguir o plano inicial (mudanças são inevitáveis. É importante saber lidar com elas).



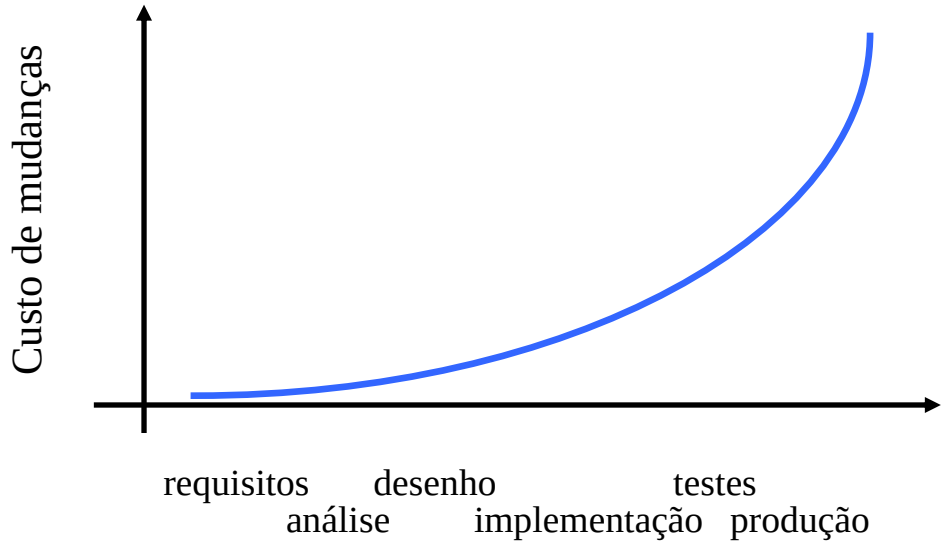
XP

- XP = *Extreme Programming*;
- Metodologia de desenvolvimento de software sendo aperfeiçoada nos últimos anos(*);
- As práticas adotadas são "as melhores práticas de engenharia de software levadas a níveis extremos";
- Originou-se das experiências de Kent Beck, Ward Cunningham e Ron Jeffries (também signatários do Manifesto Ágil) no projeto C3 (Chrysler, com Smalltalk e GemStone), 1996-1999.

(*) Correspondendo às edições 1 e 2 do livro "*Extreme Programming Explained: Embrace Change*"

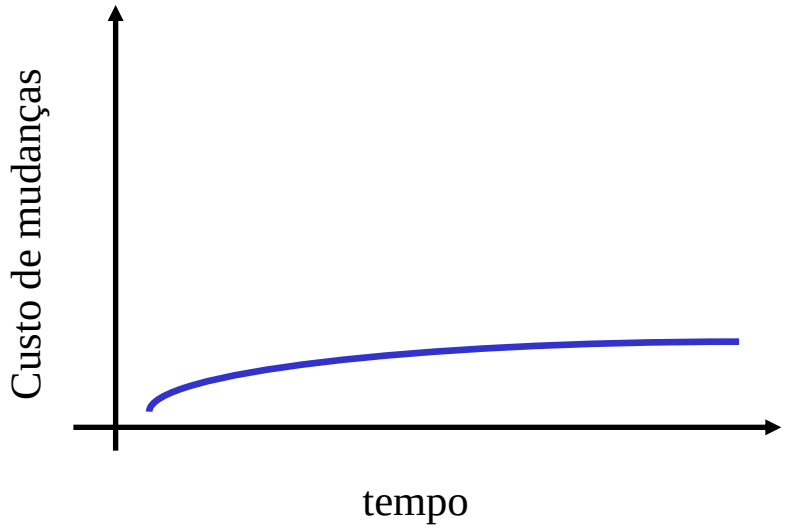
XP

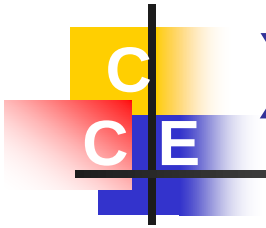
- O custo das modificações
  - Premissa *clássica*: Em processos tradicionais, os requisitos devem ser conhecidos *a priori*, já que as mudanças de requisitos têm um impacto no custo cada vez maior quanto mais tarde as mesmas ocorrerem.



XP

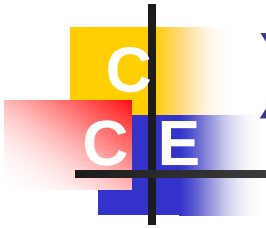
- O objetivo principal da XP é diminuir os custos de mudanças usando-se tecnologias e práticas apropriadas.





XP

- E quais são essas tecnologias e práticas?
  - Objetos são a tecnologia chave
    - Encapsulamento provê manutenibilidade;
    - Modificações de comportamento, sem alteração de código existente, podem ser (mais facilmente) implementadas através de mudanças nas trocas de mensagens entre os objetos.
  - Projeto simples, sem elementos extras (antecipação de necessidades, flexibilidades não solicitadas, etc.);
  - Testes automatizados aplicados logo após as implementações das modificações;
  - Experiência prática da equipe é valorizada, provendo autoconfiança da equipe e capacidade de definição do esforço com precisão.



XP

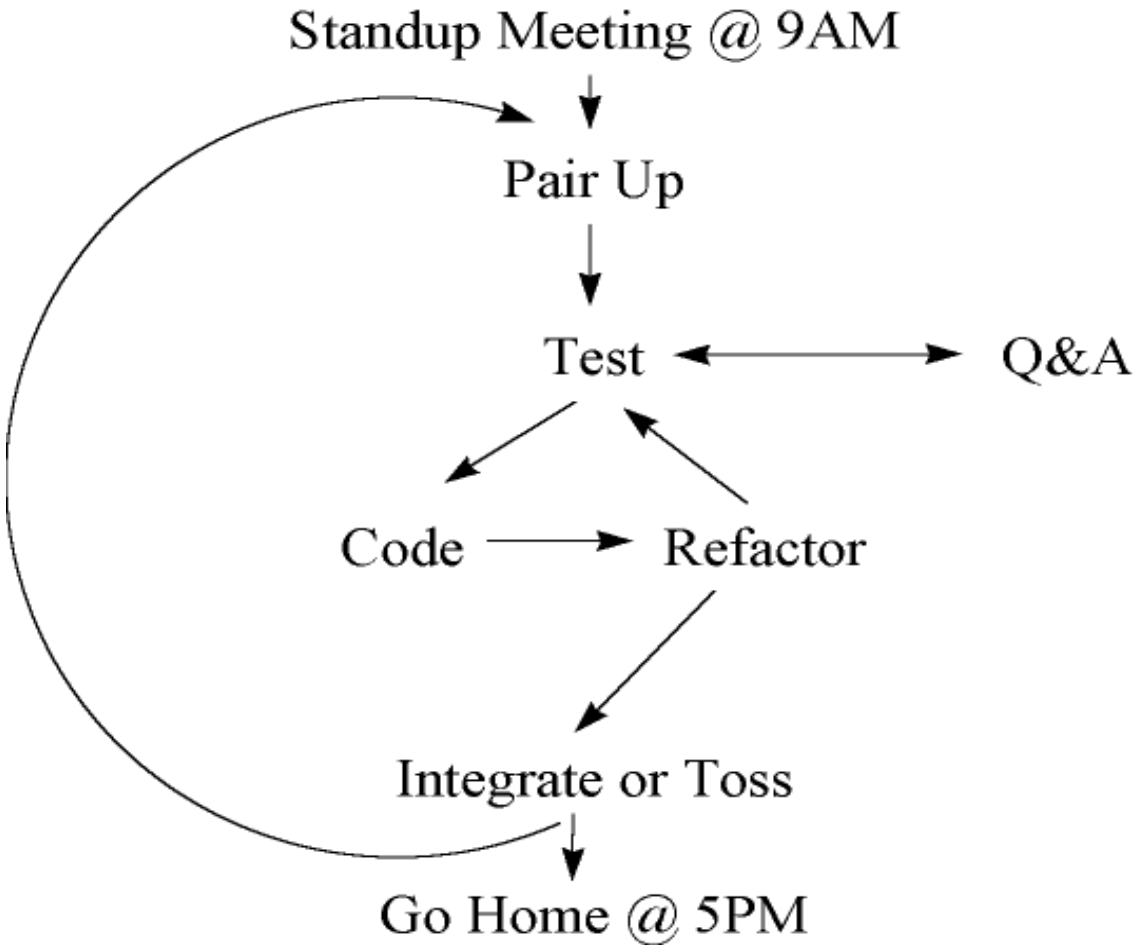
- Nesse contexto, as atitudes dos desenvolvedores ágeis são:
  - Implementam *agora* somente o que precisam *agora*;
  - Tomam as grandes decisões o mais tarde possível (quem sabe não será preciso tomá-las, de fato?);
  - Não implementam flexibilidade desnecessária num dado momento (não antecipam necessidades – quem sabe se serão mesmo necessárias?).

XP

- XP se baseia, de forma objetiva, em 24 práticas. Alguns destaques.
  - **Todos juntos**: trabalhar em um ambiente grande, para toda a equipe se ver (*War Room*).
  - **Transparência da informação:** manter as informações correntes do projeto em local de fácil visualização (e.g. um mural).
  - **Trabalho energizado**: manter ritmo de trabalho sustentável. Ter vida fora do trabalho (☺).
  - **Programação em pares**: duas pessoas programando juntas, num processo de interação constante.
  - ***User Stories***: planejar e controlar o desenvolvimento através de "pedaços" de funcionalidades do negócio, tais como vistas pelos usuários.

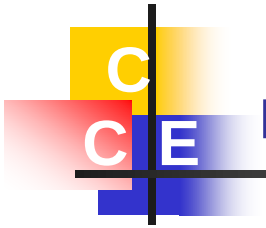
XP

O dia-a-dia de um programador XP



Fonte: http://www.xp123.com/xplor/xp0006/index.shtml

PUC-Rio



Modelos e Modelagem

Breve Discussão com a Turma

- O que é um modelo?
- Por que modelar?
- Como se constroem modelos de sistemas?

Lembrete

- Próxima aula:

  UML – Diagramas de Casos de Uso