Engenharia de Software I

Aula 04: Processos de Desenvolvimento de Software Tradicionais

Breno Lisi Romano

http://sites.google.com/site/blromano

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Bacharelado em Ciência da Computação – BCC (ENSC5)

Tecnologia em Sistemas para Internet – TSI (ESWI5)

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Sumário

- Revisão / Motivação
- Processos de Desenvolvimento de Software
- Modelos Prescritivos / Processos Tradicionais:
  - Clássicos:
    - Codifica e Remenda
    - Cascata
  - Evolucionários:
    - Prototipagem Evolutiva
    - Incremental / Evolucionário
    - Espiral
  - Processo Unificado

Revisão: História

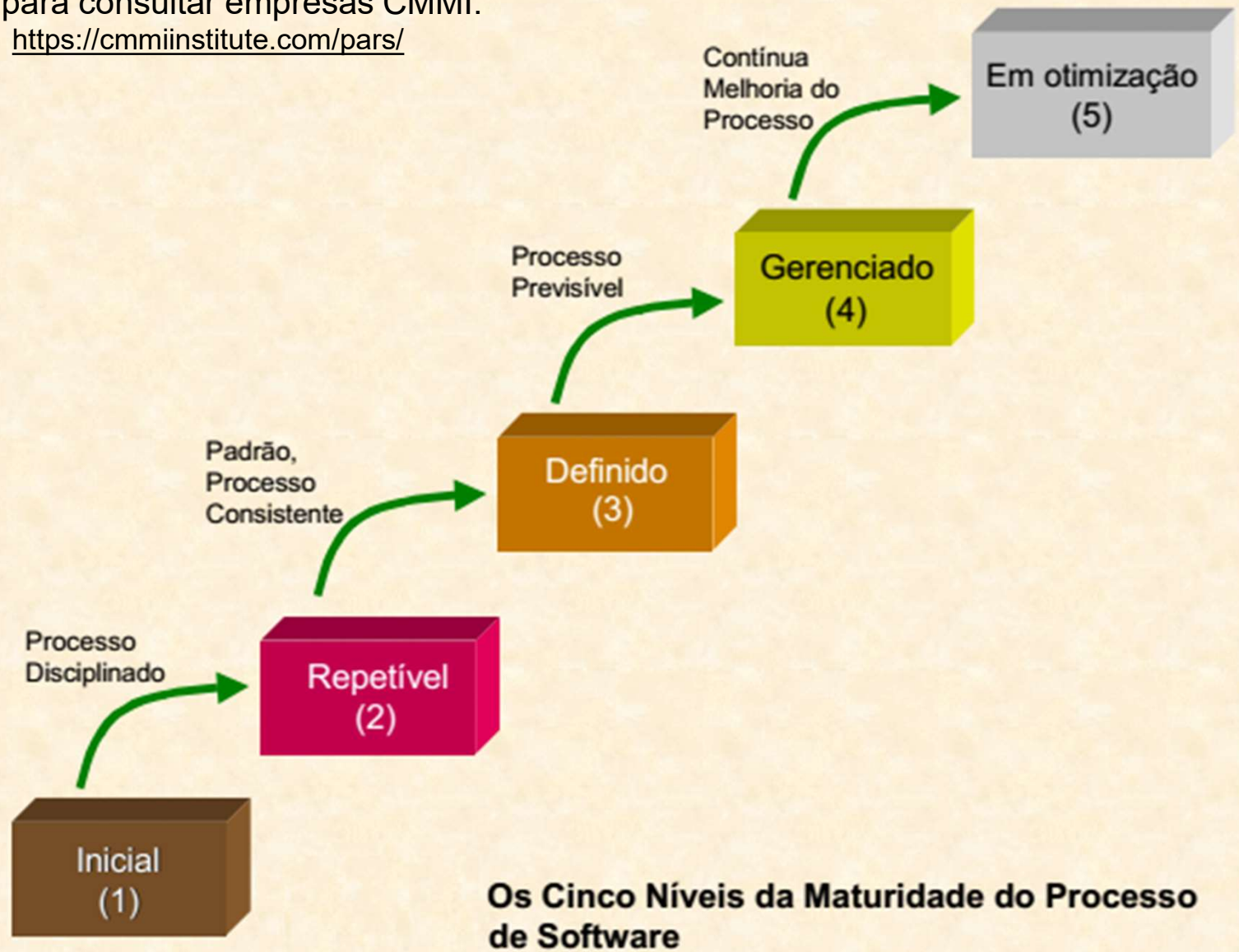
- Departamento de Defesa dos USA
  - Quase 90% dos projetos falham. Porquê?
    - Pesquisa pela Carnegie-Mellon
    - Depois de muito tempo a resposta:
      - Projetos falham por gerência inadequada!
        - » Ausência de processos que auxiliem no controle das atividades de desenvolvimento de software
    - Nascimento do *Capability Maturity Model* - CMM (atual CMM-I)
      - Níveis de maturidade de uma empresa de desenvolvimento
      - Classificação de 1 a 5

Revisão: *Capability Maturity Model* – CMM

- **Definição:**
  - CMM **fornece** às **organizações** de **software** um **guia** de como **obter controle em seus processos para desenvolver e manter software** e **como evoluir em direção a uma cultura de Engenharia de Software e excelência de gestão**
  - Objetivo de **auxiliar** na **seleção** das **estratégias** de **melhoria**, determinando a **maturidade atual** do processo e **identificando** as **questões** mais **críticas** para a **qualidade e melhoria do processo** de software
- **CMM não é um processo de desenvolvimento de software**

Revisão: Níveis de Maturidade

Site para consultar empresas CMMI:
https://cmmiinstitute.com/pars/



Os Cinco Níveis da Maturidade do Processo de Software

Revisão: Processos de Desenvolvimento

- Fundamentais para qualidade
  - Processos ajudam mas...
    - No *silver bullet*!
  - Modelos de processos
    - Codifica-Remenda
    - Cascata
    - Evolucionário
    - Prototipagem Evolutiva
    - Espiral
    - Processo Unificado
    - Orientado à Modelos
    - Ágeis

O que é um *processo de desenvolvimento de software*?

É um **conjunto de atividades** necessárias para **transformar as necessidade** dos usuários e suas expectativas em um sistema baseado em **software**

O que é um *modelo* de processo de desenvolvimento de software?

- É uma **referência**, abordagem que caracteriza e guia quanto as atividades de melhoria de um processo de desenvolvimento de software
- Possibilita a **avaliação da maturidade** do processo de desenvolvimento de software
- Referência e estabelece parâmetros de mercado quanto a **capacidade em produzir software** de qualidade
- Um modelo de processo é um **conjunto estruturado** de **boas práticas** que descrevem as **características** de **efetividade** de um processo
  - **Boas práticas** referem-se aquelas comprovadas pela **experiência**

Tipos de Processos de Desenvolvimento de Software

- Existem **duas grandes** Escolas:
  - **Modelos Prescritivos / Processos Tradicionais:**
    - São modelos que definem um **conjunto distinto de atividades**, **tarefas**, **marcos** e **produtos** de trabalho que são **necessários** para fazer **Engenharia de Software** com **alta qualidade**
    - **Processos** que se baseiam em uma **descrição de como as atividades são feitas**
  - **Metodologias Ágeis (Próximas Aulas):**
    - Modelos que definem um **conjunto de valores, princípios e práticas** que **auxiliam** a **equipe de projeto** a **entregar produtos** ou **serviços** de **valor** em um ambiente
    - **Valor central:** As **respostas às mudanças** são mais importantes que o cumprimento de um plano

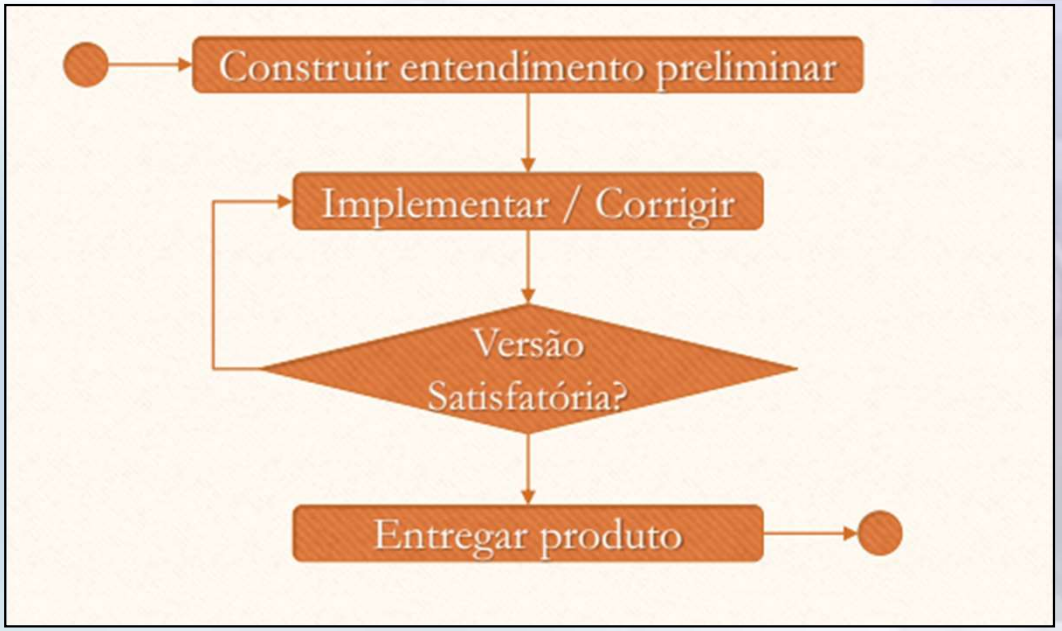
Modelos Prescritivos / Processos Tradicionais

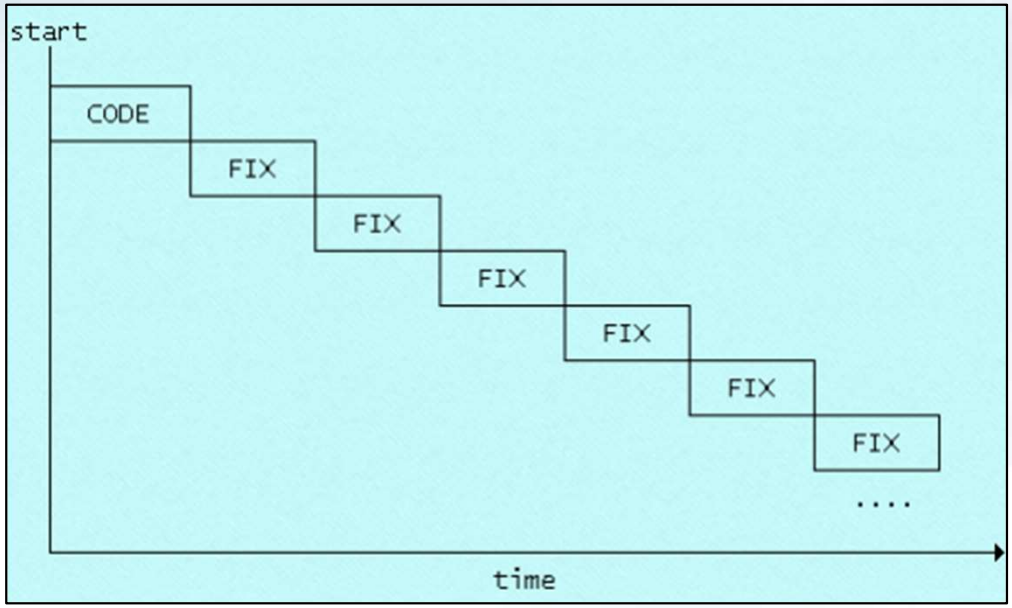
Clássicos

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Modelo Codifica-Remend (*Code and Fix*) (1)

Modelo Codifica-Remend (*Code and Fix*) (2)





Modelo Codifica-Remend (*Code and Fix*) (3)

- **Em Resumo:**
  - Construir com o cliente um **entendimento preliminar** sobre o sistema que deve ser desenvolvido
  - **Implementar** uma primeira versão desse sistema
  - **Interagir** com o **cliente** de forma **a corrigir a versão preliminar** até que esta **satisfaça** o cliente
  - Fazer **testes e corrigir** os erros inevitáveis
  - **Entregar** o produto
- Muito usado (infelizmente...)
- **Não** exige **gerência** complexa
  - Nenhuma documentação
  - Nenhum controle gerencial
  - Atraente para alguns desenvolvedores

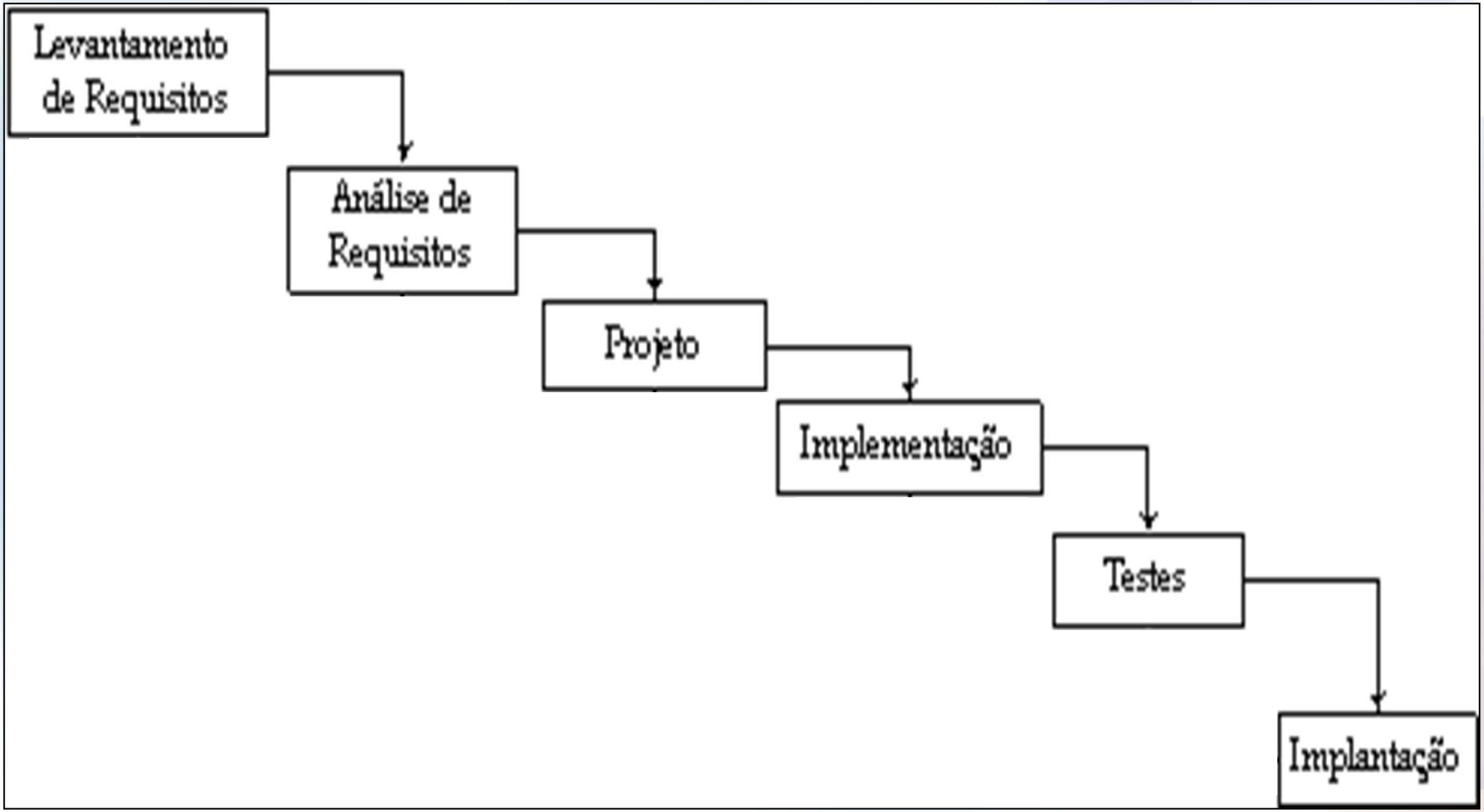
Modelo Codifica-Remenda (*Code and Fix*) (4)

- **Conclusão:** Não é um modelo de processo
  - **Não** há **previsibilidade** em relação às **atividades**
  - **Não** há **previsibilidade** em relação aos **resultados** obtidos
- Deve ser entendido mais como uma maneira de **pressionar** os **desenvolvedores** do que como um processo organizado
- **(Des) Vantagens:**
  - Não há tempo em documentação, planejamento ou projeto: “Direto ao código!”
  - O progresso é facilmente visível a medida que o software vai ficando “pronto”
  - Não há necessidade de treinamento ou conhecimentos especiais. Qualquer desenvolvedor pode implementar software
  - É muito difícil avaliar a qualidade e os riscos do projeto
  - Se no meio do projeto a equipe descobrir que as decisões arquiteturais estavam erradas, não há solução, a não ser começar tudo de novo

Modelo em Cascata / *Waterfall* (1)

- O avô de todos os modelos
- Começou a ser definido nos anos 1970
- Filosofia:
  - BDUF – *Big Design Up Front*:
    - **Antes** de **linhas de código**, é preciso fazer um **trabalho** detalhado de **análise e projeto** → Quando o **código** for **efetivamente produzido**, esteja o mais **próximo** possível dos **requisitos**
    - Antes de avançar à próxima fase deve haver uma revisão da fase anterior

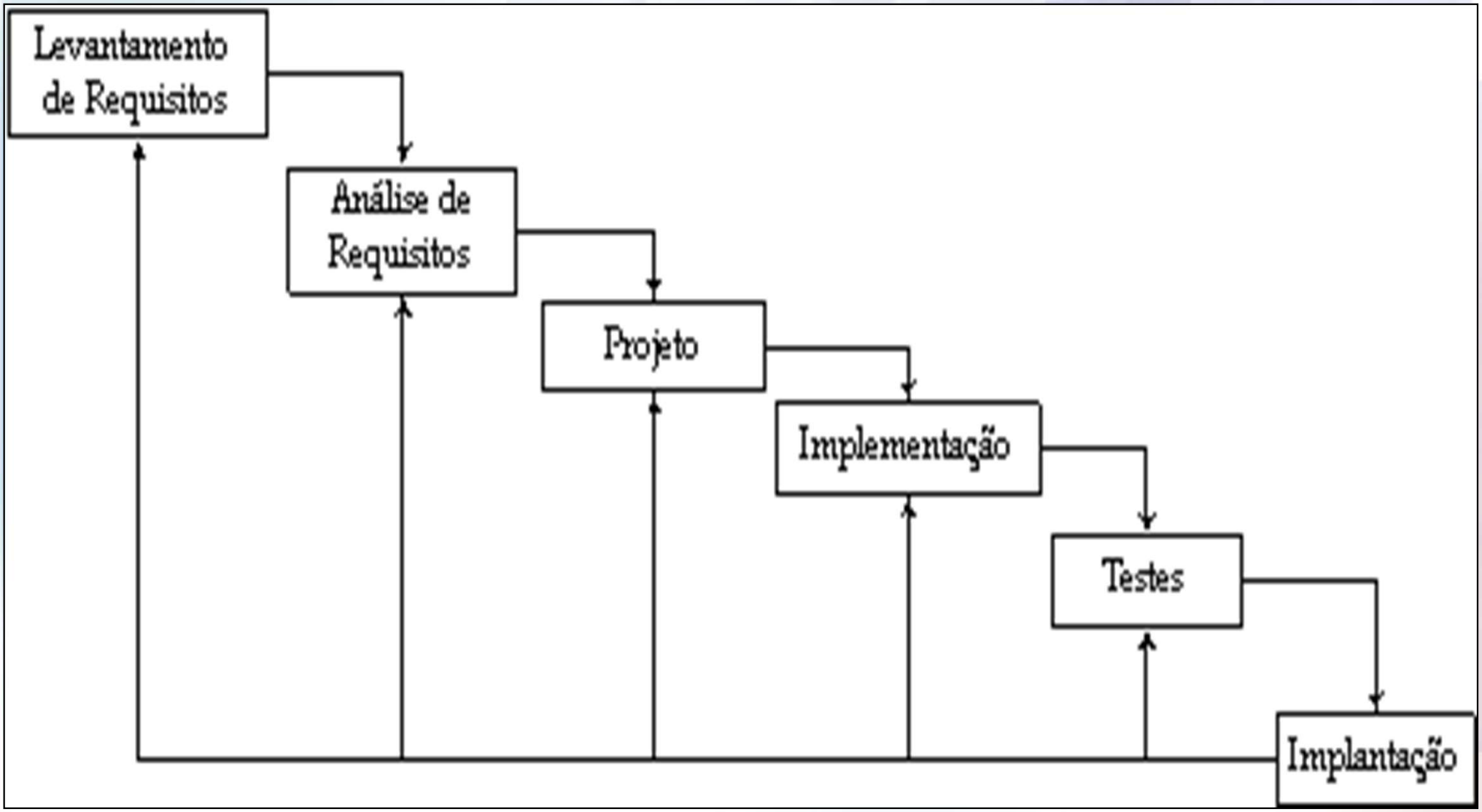
Modelo em Cascata / *Waterfall* (2)



Modelo em Cascata / *Waterfall* (3)

- **Estrutura rigorosa**
  - Uma etapa só começa quando a anterior estiver totalmente concluída.
  - Inflexível
    - Não se adapta bem a mudanças de requisitos
- **Quando** usar:
  - **Requisitos** bem **conhecidos**
    - **Mas.....** É difícil estabelecer requisitos completos antes de começar a codificar
  - É **adequado** para **equipes tecnicamente** fracas ou **inexperientes** → **Estrutura sólida** ao projeto, direcionando todos os esforços
- **Cliente** só possui o **produto** ao **final**
  - Não produz resultados tangíveis até a fase de implementação (na perspectiva do cliente)
- **O que pode ser feito pra melhorar este modelo?**

Modelo Cascata com Realimentação



Modelos Prescritivos / Processos Tradicionais

Modelos Evolutivos

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

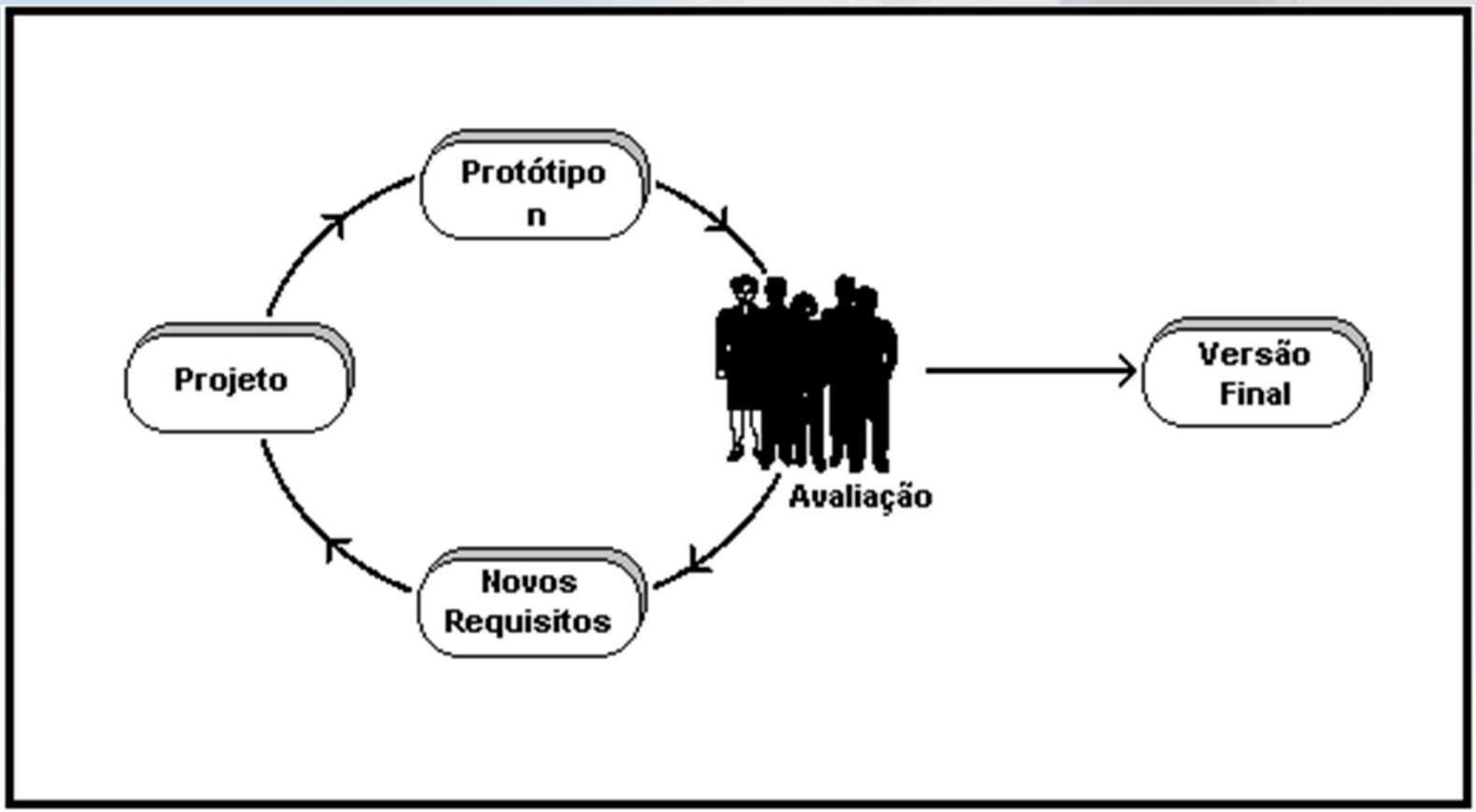
Modelo Prescritivos / Tradicionais: Evolutivos

- Existem situações em que a Engenharia de Software **necessita** de um **modelo** de **processo** que possa **acomodar** um **produto** que **evolui com** o **tempo**
  - Modelos evolutivos são **iterativos**
  - Possibilitam o desenvolvimento de **versões cada vez mais completas** do software

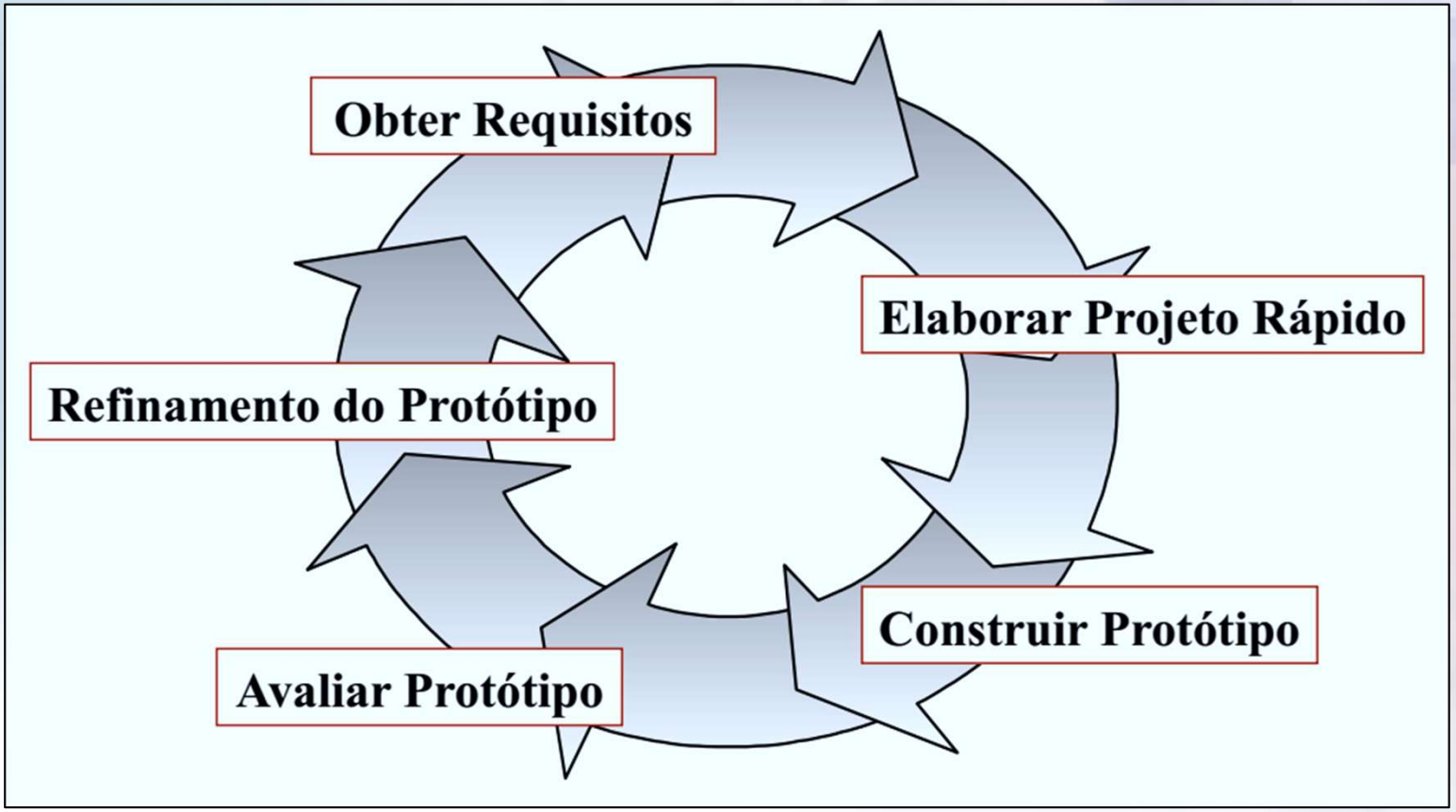
- **Quando utilizar:**
  - Requisitos de produto e de negócio evoluem conforme o desenvolvimento procede
  - Data de entrega apertada (mercado): impossível a conclusão de um produto completo
  - Conjunto de requisitos importantes é bem conhecido, porém os detalhes ainda devem ser definidos

Prototipagem Evolutiva (1)

- **Objetivo: entender os requisitos** do usuário → **obter uma melhor definição dos requisitos do sistema**
  - Possibilita que o desenvolvedor crie um modelo (protótipo) do software que deve ser construído
  - Apropriado para quando o cliente não definiu detalhadamente os requisitos



Prototipagem Evolutiva (2)



Prototipagem Evolutiva (2)

- Bom para o desenvolvedor
  - Codificação rápida
- Bom para o cliente
  - Versão desenvolvida rapidamente
- Ruim porquê...
  - Cliente não vê “remendos”
  - Não fica claro a diferença na execução das atividades de desenvolvimento e testes

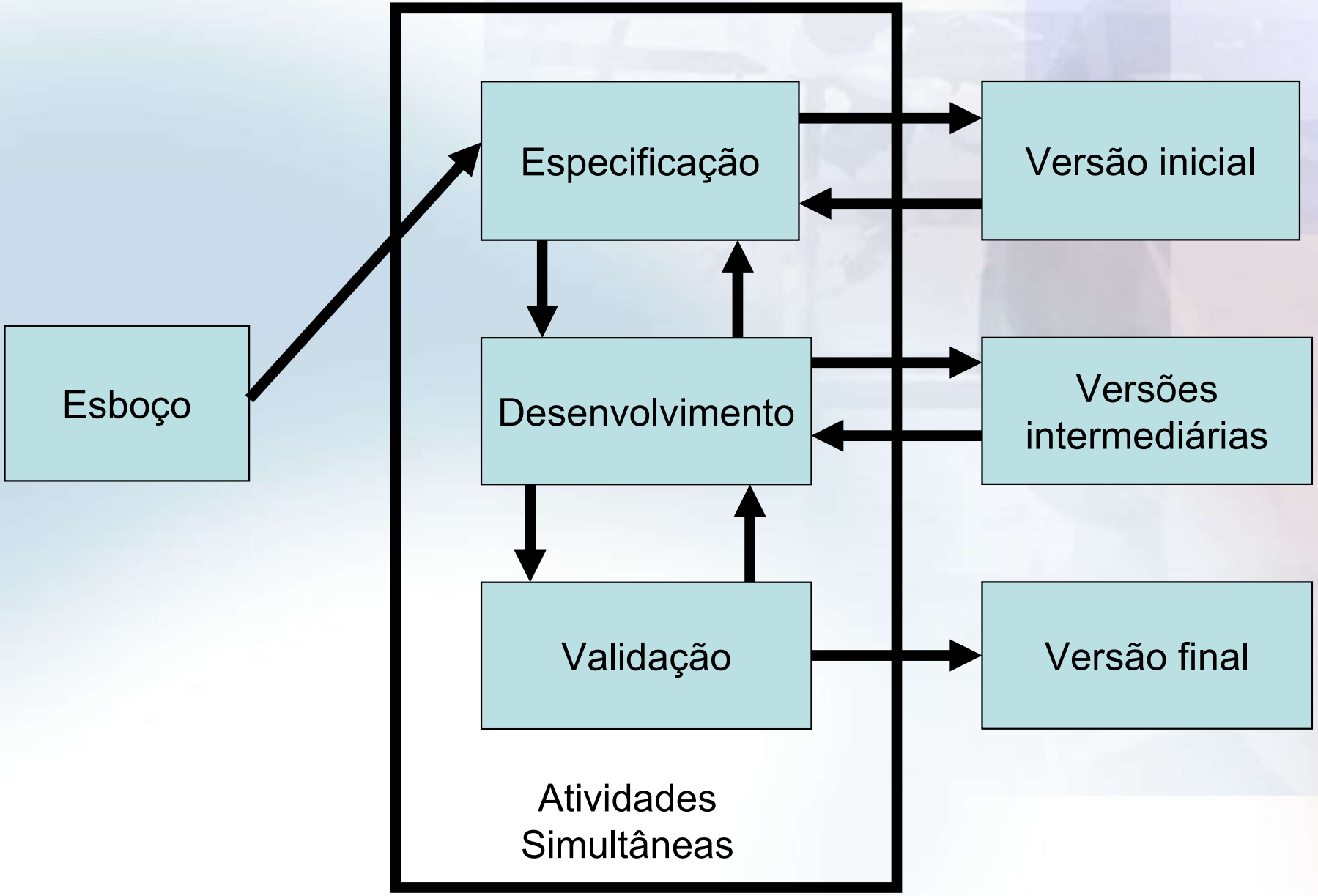
Têm que existir um contrato financeiro bem definido com os clientes. Por quê?

Modelo Incremental / Evolucionário (1)

- **Proposta**: **combina** elementos do modelo **cascata** (aplicado repetidamente) com a filosofia **iterativa** da **prototipação**

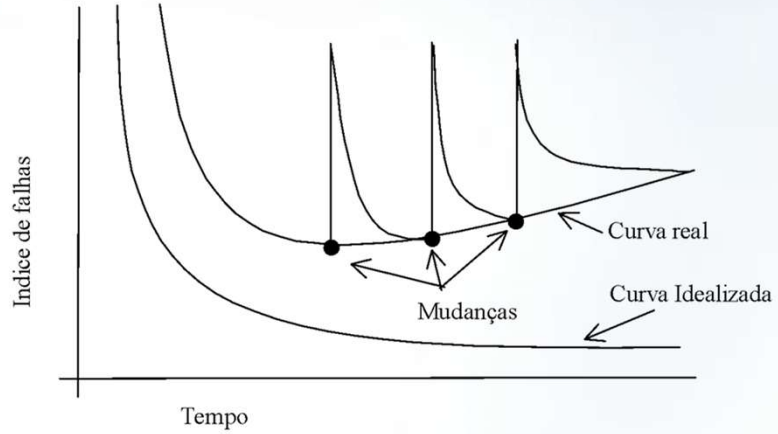
- **Ideia: trabalhar** junto do **cliente** para **descobrir** seus **requisitos**, de maneira **incremental**, até que o produto final seja obtido

Modelo Incremental / Evolucionário (2)



Modelo Incremental / Evolucionário (3)

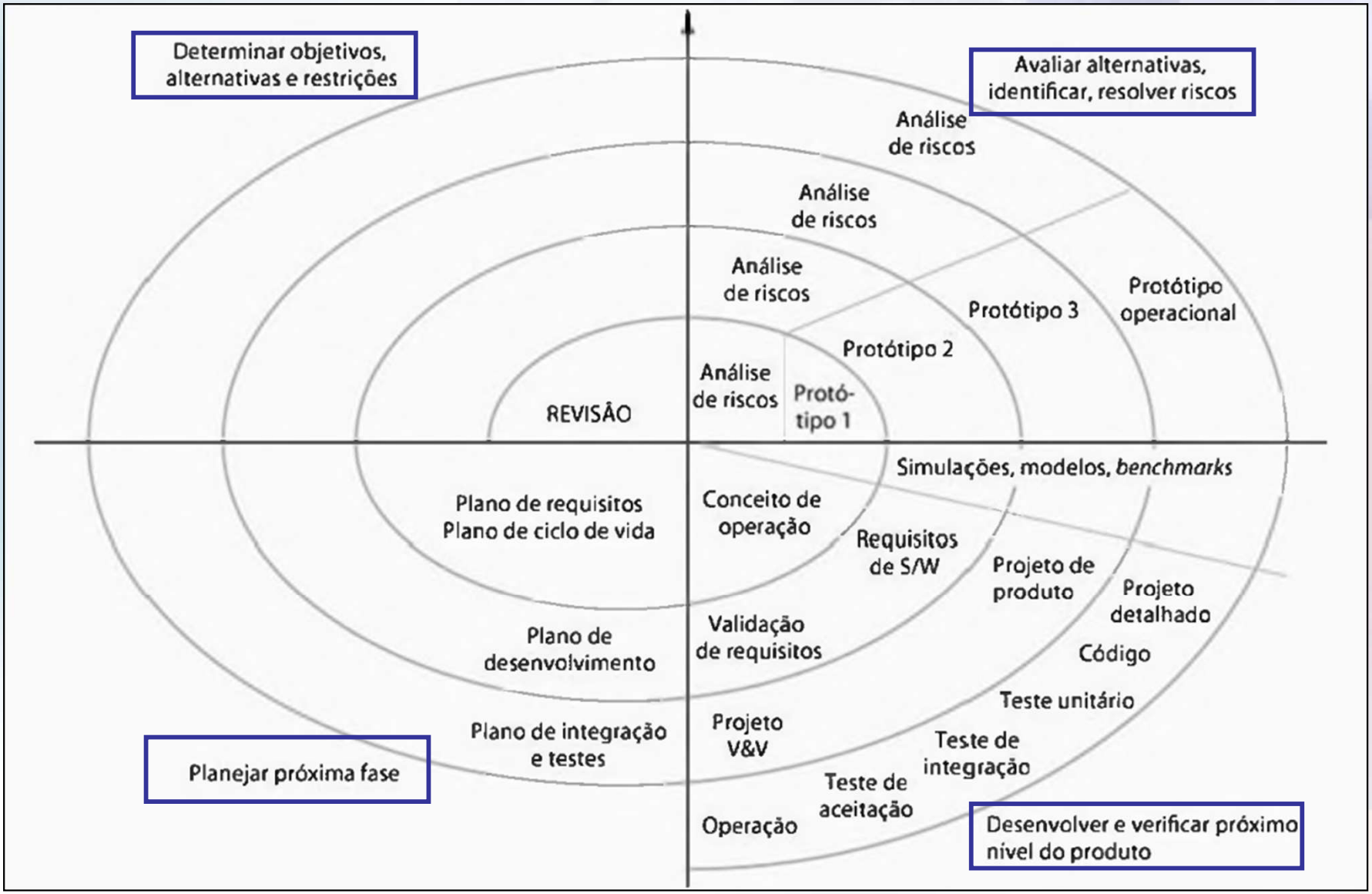
- Especificação **incremental**
- Modelo importante quando é **difícil estabelecer**, a priori, uma **especificação detalhada** dos **requisitos**
- Ideal para sistemas médios e pequenos
  - menos de 500.000 LOC
- **Problemas:**
  - Difícil gerenciar pequenos incrementos
  - Sistemas mal estruturados
    - Muitas mudanças tendem a corromper a estrutura do software



Modelo Espiral (1)

- Proposto por Boehm (1986)
- Orientado a **redução** de **riscos**
- Baseado na realização de **ciclos iterativos**
- **Proposta**:
  - Iniciar com miniprojetos, abordando os principais riscos
  - Expandir o projeto através da construção de protótipos, testes e replanejamento → Abarcar os riscos identificados
  - Após a equipe ter adquirido um conhecimento mais completo dos potenciais problemas com o sistema, passará a desenvolver um ciclo final semelhante ao do modelo cascata

Modelo Espiral (2)



Modelo Espiral (3)

- **Passos** do Modelo:
  1. Determinar inicialmente os objetivos, alternativas e restrições relacionadas à iteração que vai se iniciar
  2. Identificar e resolver riscos relacionados à iteração em andamento
  3. Avaliar as alternativas disponíveis
     - Podem ser utilizados protótipos para verificar a viabilidade de diferentes alternativas
  4. Desenvolver os artefatos relacionados a essa iteração e certificar-se de que estão corretos
  5. Planejar a próxima iteração
  6. Obter concordância em relação à abordagem para a próxima iteração, caso se resolva realizar uma

Modelo Espiral (4)

- **Similar** a outros **processos**
  - Ex: Processo evolucionário
- **Diferença**
  - Análise e resolução de riscos
    - Ex.
      - Uso de nova linguagem: Ferramentas não disponíveis ou com problemas.
- **Problema**
  - Difícil gerenciar todos os incrementos produzidos
  - Não fornece indicações suficientes sobre quantidade de trabalho esperada em cada ciclo
    - O tempo de desenvolvimento (prazo) se torna imprevisível

Outros modelos...

- **Processo Unificado**
- **Desenvolvimento Baseado em Modelos *(Model Driven Development* - MDD)**
- Métodos formais
- Engenharia de software baseada em componentes
- **Desenvolvimento Baseado em Testes (*Test Driven Development* - TDD)**
- **Metodologias Ágeis**

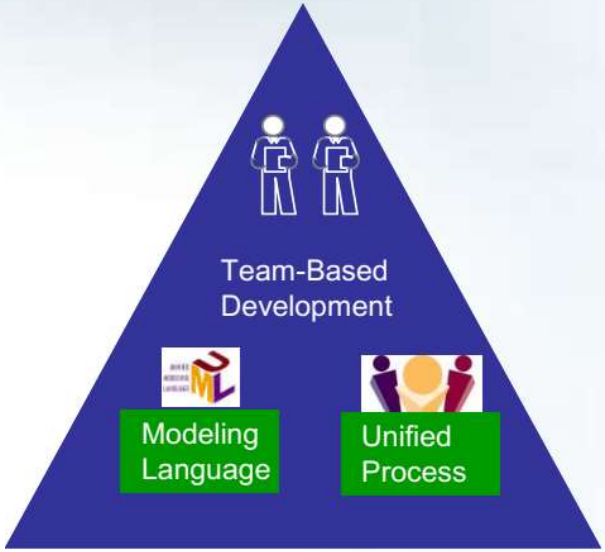
Processo Unificado da Rational (RUP)

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Contexto para Aplicação do RUP

- Não é suficiente apenas a presença de desenvolvedores altamente treinados:

  Necessita-se de uma linguagem para a equipe poder se comunicar entre si e com os clientes (UML), além disso necessitamos de um guia organizacional: um processo (RUP)



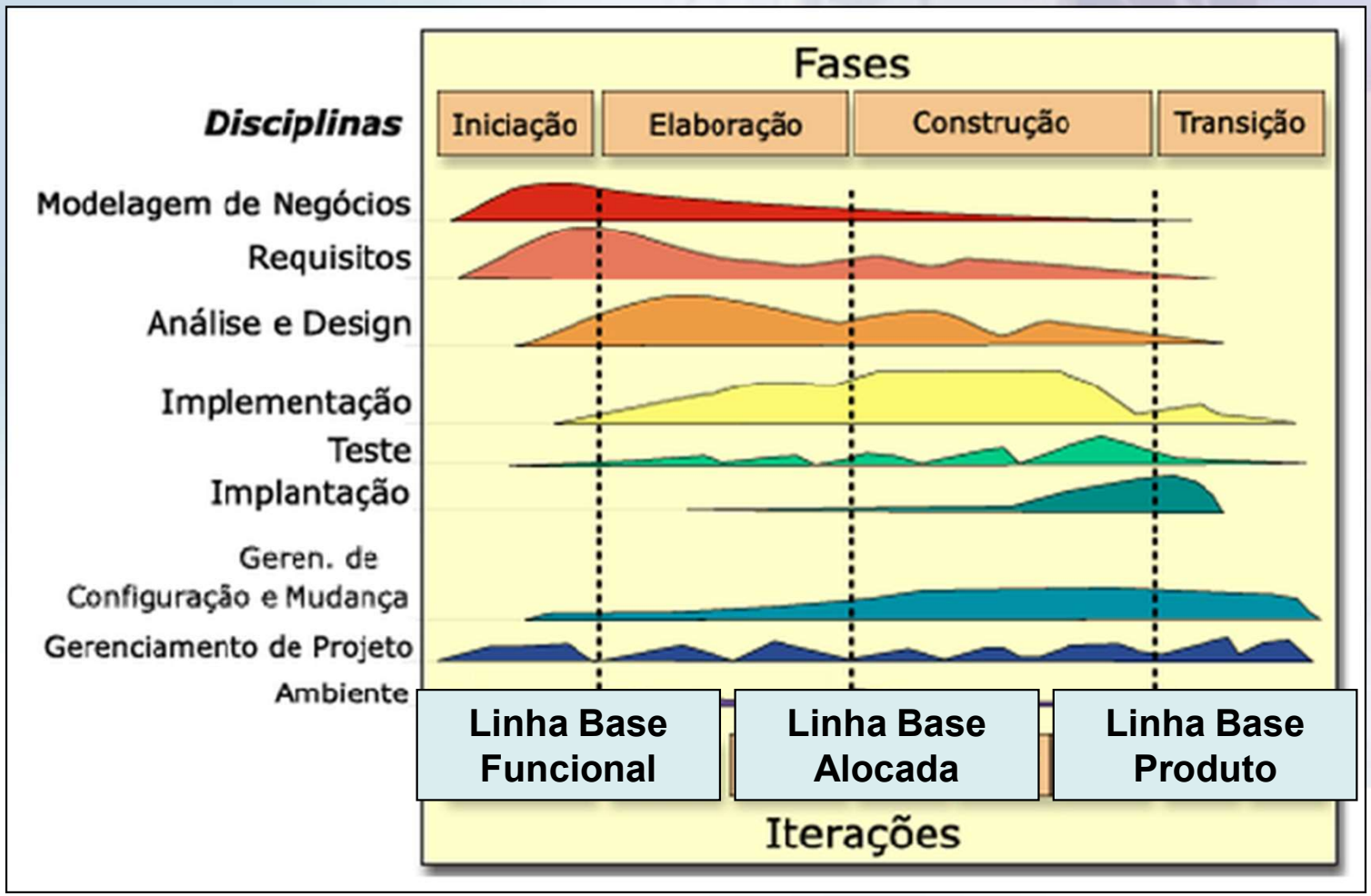
O que é UML (*Unified Modeling Language*) ?

- A UML (*Unified Modeling Language*) é o sucessor de um conjunto de métodos de análise e projeto orientados a objeto (OOA&D)
- A UML é um modelo de linguagem, não um método
  - Um método pressupõe um modelo de linguagem e um processo
  - O modelo de linguagem **é a notação que o método usa para descrever o projeto**
  - O processo são os passos que devem ser seguidos para se construir o projeto
- O modelo de linguagem corresponde ao ponto principal da comunicação
  - Se uma pessoa quer conversar sobre o projeto, como outra pessoa, é através do modelo de linguagem que elas se entendem
- A UML é uma linguagem-padrão para a elaboração da estrutura de projetos de software
- É empregada para a visualização, especificação, construção e documentação de artefatos que façam uso de sistemas complexos de software

“UML é uma linguagem de modelagem, não uma metodologia”

O que é RUP (Rational Unified Process) ?

- É um processo configurável de Engenharia de Software
- O RUP é um guia para como usar efetivamente a UML



RUP e CMM (*Capability Maturity Model*)

- O objetivo do RUP é assegurar uma produção de alta **qualidade de software**, que realiza a necessidade do usuário seguindo prazos e o orçamento

- Com o advento do CMMi, as organizações focalizam a qualidade em primeiro plano e o RUP pode ser bastante útil quando se quer atingir níveis maiores

Engenharia de Software I

Aula 04: Processos de Desenvolvimento de Software Tradicionais

Dúvidas?

Breno Lisi Romano

http://sites.google.com/site/blromano

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Bacharelado em Ciência da Computação – BCC (ENSC5)

Tecnologia em Sistemas para Internet – TSI (ESWI5)

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista