PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
Ciência da Computação

Engenharia de Software - Processo

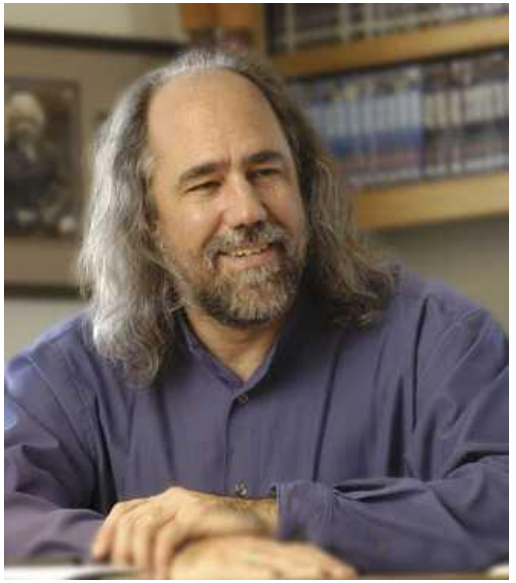
Rational Unified Process

Prof. Carlos Eduardo de B. Paes
Departamento de Computação
Pontifícia Universidade Católica de São Paulo
carlosp@pucsp.br

“Three Amigos”

Ivar Jacobson
(OOSE)

James Rumbaugh
(OMT)

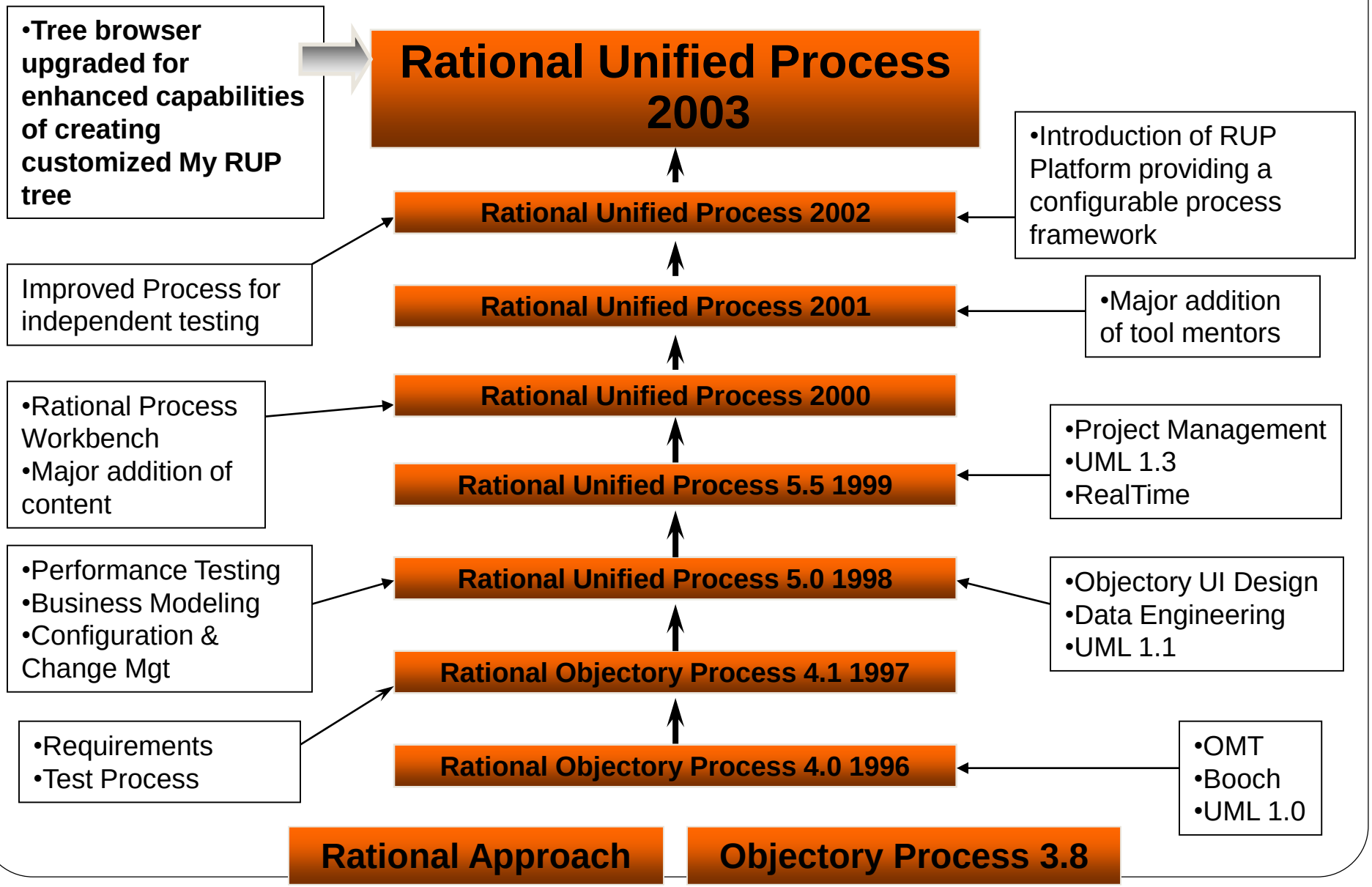


Grady Booch
(Booch 93)

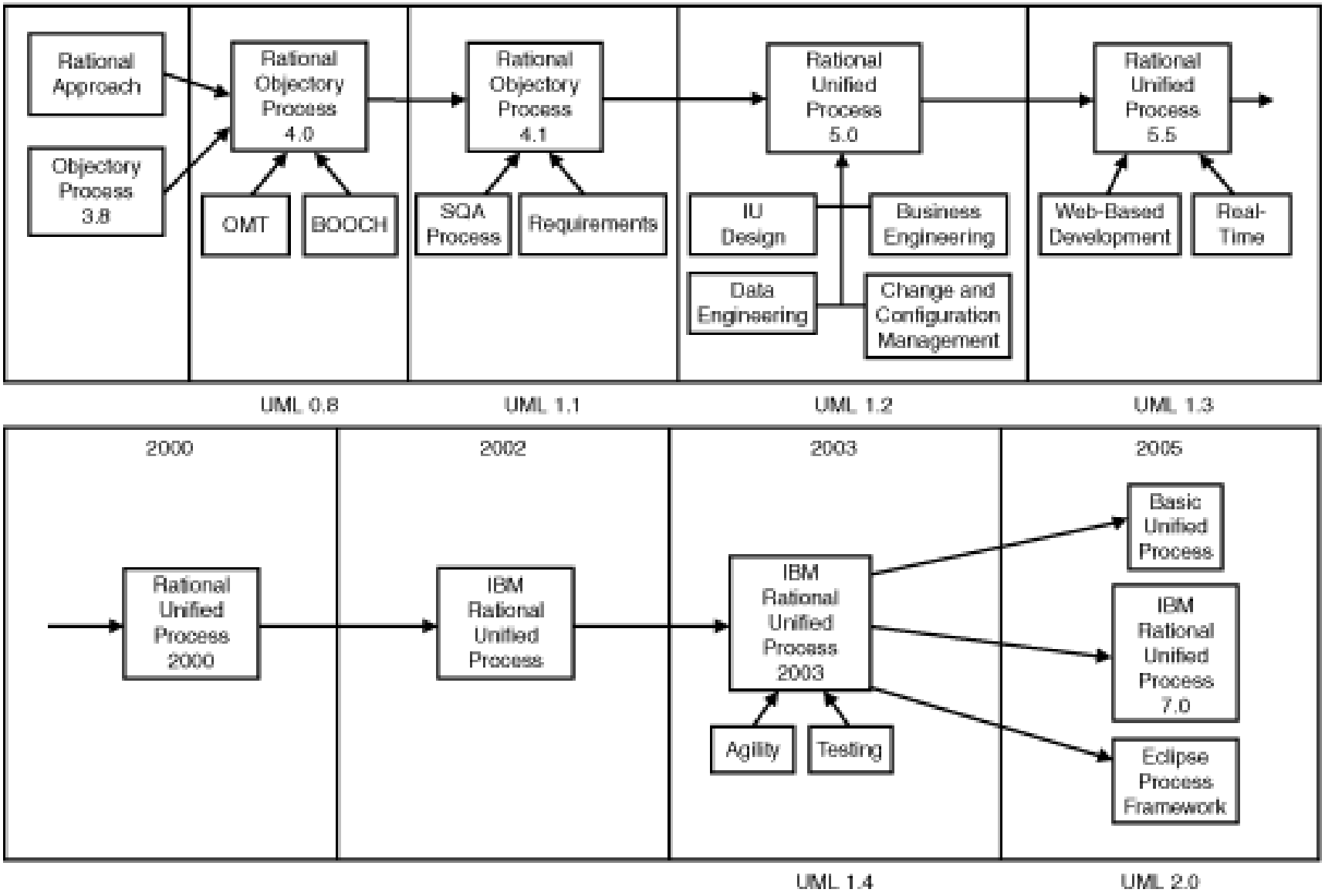
Rational Unified Process

- Processo Produto
  - IBM Rational
  - Rational Method Composer (RUP 2007)
- É um framework de processo (ou metamodelo e processo)
- Captura e apresenta as boas práticas de desenvolvimento de software
- Está sempre em evolução
- Contém templates, orientações e Help on-line
- Promove visão e cultura comum entre membros da equipe
- Suporta diversas ferramentas (Rational)

Rational Unified Process
Evolução Histórica



Rational Unified Process
Evolução Histórica



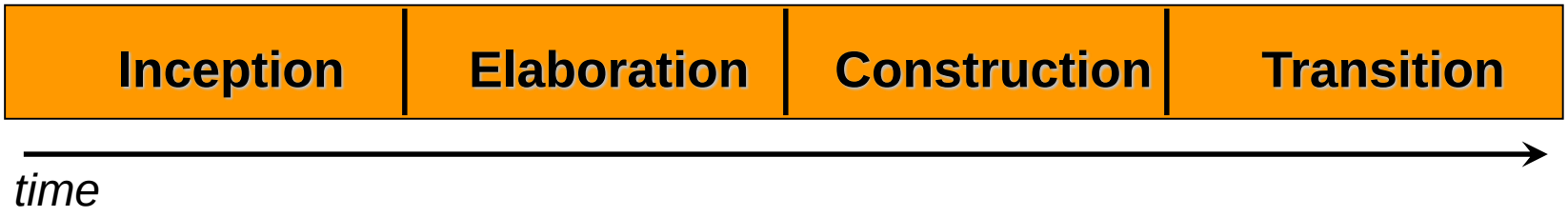
UML e RUP

- Andam de mãos dadas
- Muitos dos artefatos do RUP tem uma representação UML
- RUP também inclui orientações para conceitos da UML

Organização do RUP

- RUP é organizado:
  - Pelo tempo
    - **Fases e Iterações**
  - Pelo Conteúdo
    - **Disciplinas**

RUP Organização pelo Tempo

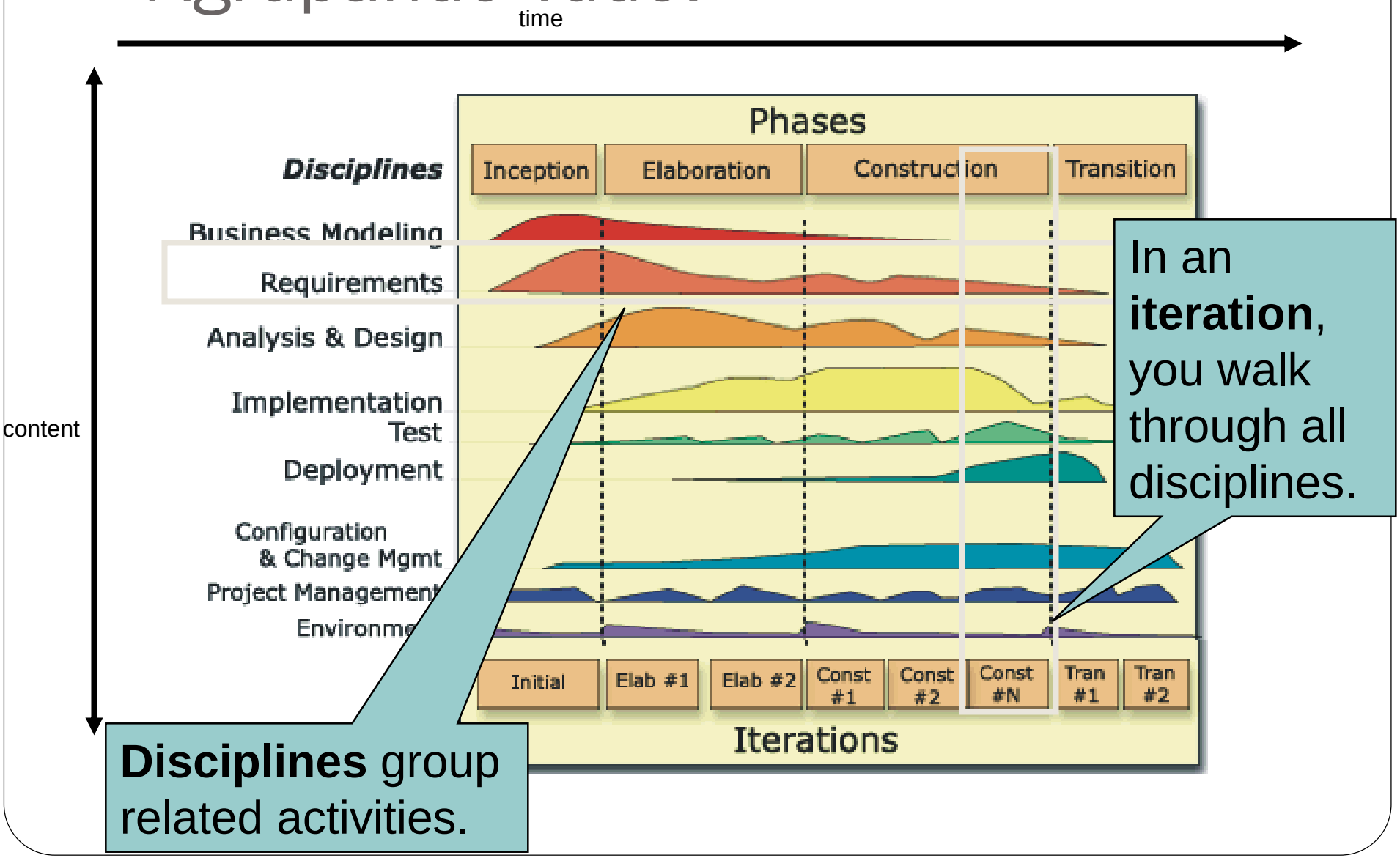


- RUP possui quatro fases:
  - **Concepção** (*Inception*): define o escopo do projeto
  - **Elaboração** (*Elaboration*): Plano do projeto, características específicas, linha-base da arquitetura
  - **Construção** (*Construction*): Construção do produto
  - **Transição** (*Transition*): transição do produto para o usuário final

RUP Organização pelo Conteúdo

- O conteúdo do RUP é organizado em disciplinas
- Uma disciplina é uma coleção de atividade que são todas relacionadas com as principais áreas de interesse
- As disciplinas do RUP são:
  - Modelagem de Negócios
  - Requisitos
  - Análise & Projeto
  - Implementação
  - Testes
  - Implantação
  - Gerenciamento de Configuração e Mudanças
  - Gerenciamento de Projetos
  - Ambiente

Agrupando Tudo!



Rational Unified Process
As Filosofias

- Atacar os maiores riscos o mais cedo possível e continuamente, ou eles irão atacar você
- Assegurar que está sendo entregue algo de valor para seu cliente
- Ficar focado em um software executável
- Acomodar mudanças o mais cedo possível no projeto
- Definir uma linha base de uma arquitetura executável o mais cedo possível

Rational Unified Process
As Filosofias

- Construir o seu sistema com componentes
- Trabalhar em grupo como um time
- Fazer a qualidade como um estilo de vida, não um preocupação postergada

Boas Práticas Propostas pelo RUP

- Desenvolvimento iterativo
- Gerenciamento de requisitos
- Arquitetura componentizada
- Modelagem visual (UML)
- Verificação contínua da qualidade
- Gerência de mudanças

Prática 1:
Desenvolvimento Iterativo

Boas Práticas

Desenvolvimento Iterativo

Gerência de Requisitos

Arquitetura Componentizada

Modelagem Visual (UML)

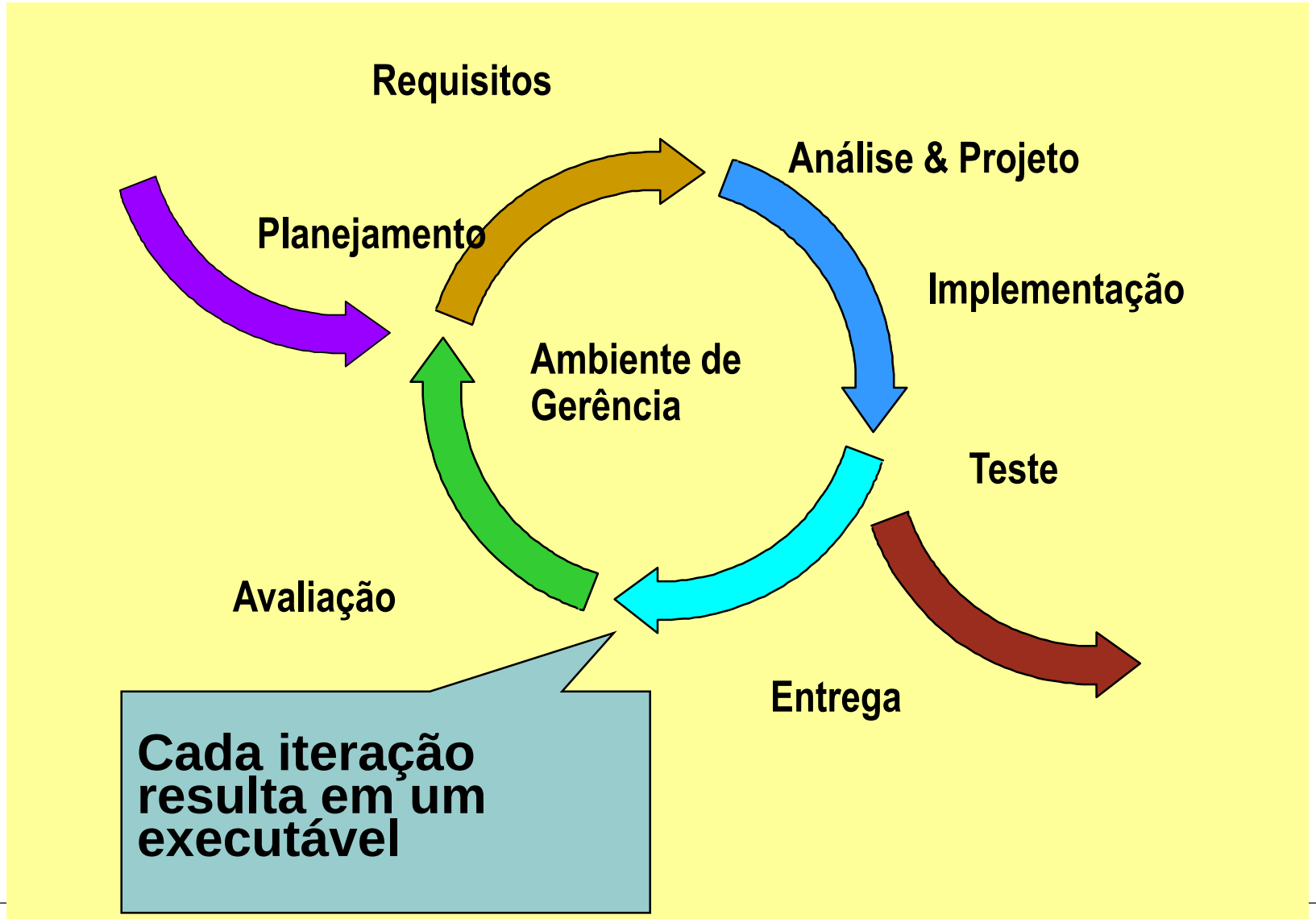
Verificação Contínua da Qualidade

Gerência de Mudanças

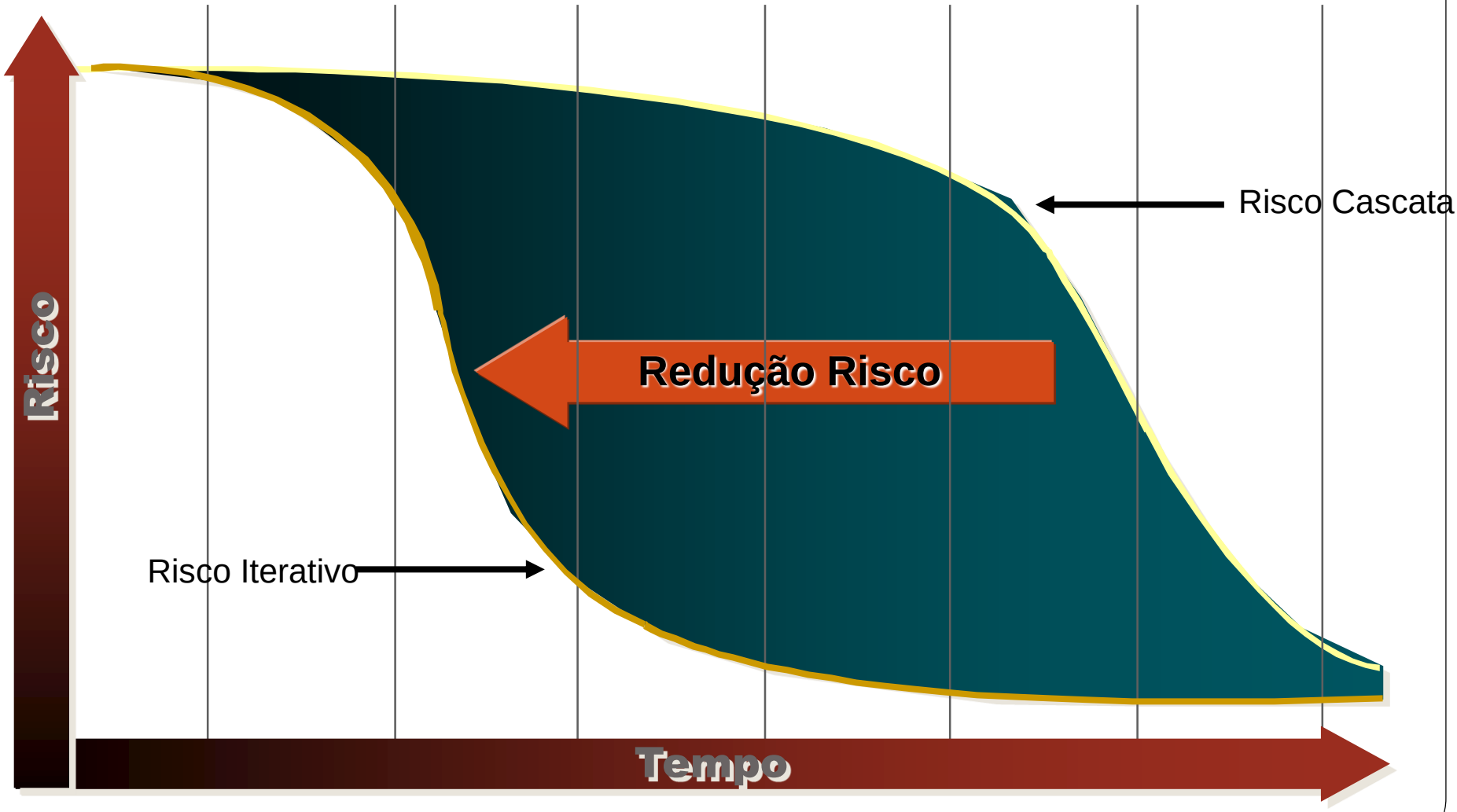
Desenvolvimento em Cascata

- Demora na confirmação de resolução dos riscos críticos
- Medidas progridem avaliando-se os produtos de trabalho que são fontes pobres de previsão de tempo
- Atrasos na integração e testes
- Impede desenvolvimento prematuro
- Freqüentemente resulta em grandes iterações não planejadas

Desenvolvimento Iterativo Produz Executáveis



Perfil dos Riscos



Prática 2:
Gerência de Requisitos

Boas Práticas

Desenvolvimento Iterativo

Gerência de Requisitos

Arquitetura Componentizada

Modelagem Visual (UML)

Verificação Contínua da Qualidade

Gerência de Mudanças

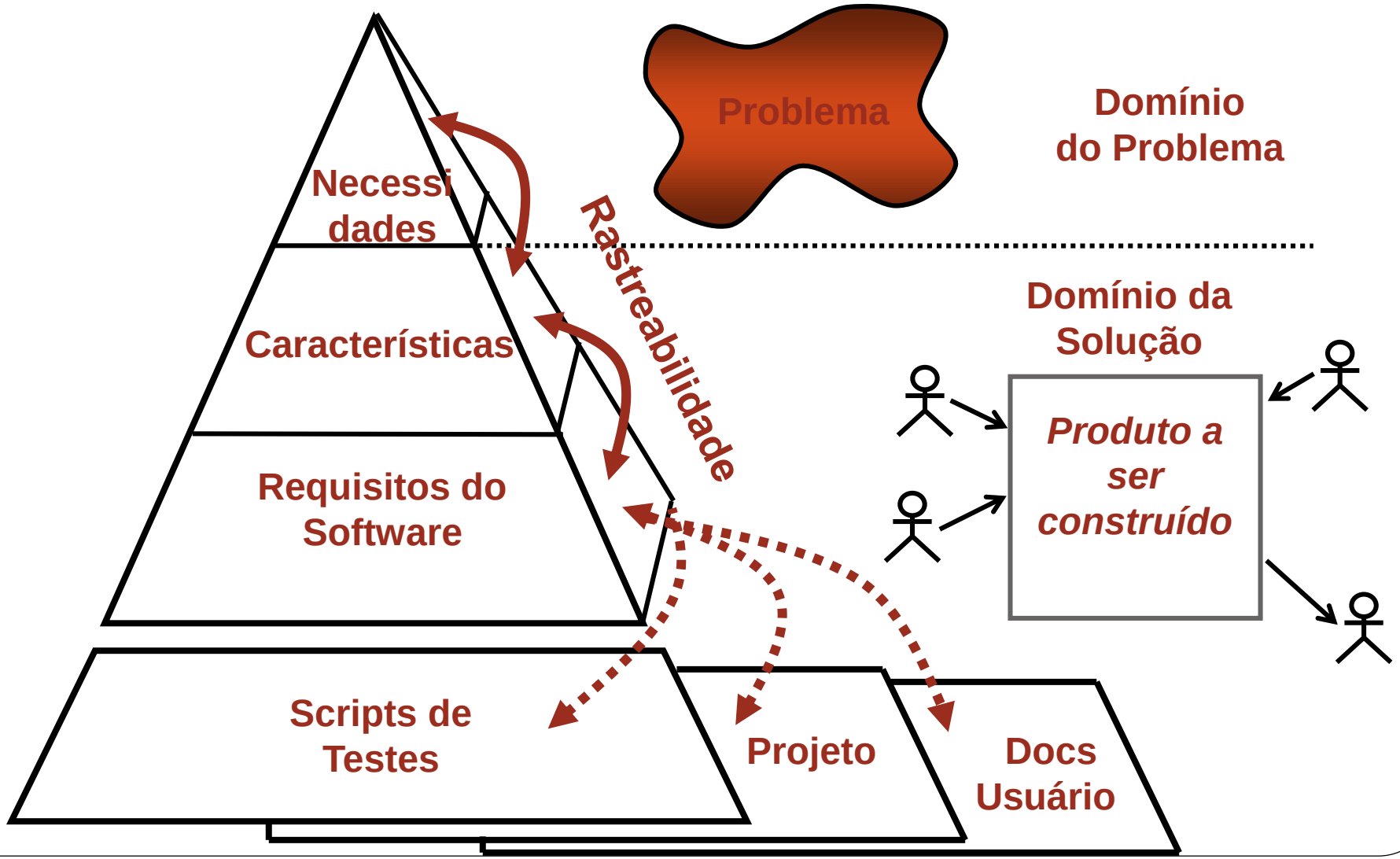
Gerência de Requisitos Significa...

- Ter certeza que você:
  - Resolve o problema certo
  - Constrói o sistema certo
- Através de uma abordagem sistemática para:
  - capturar
  - organizar
  - documentar
  - gerenciar
- As mudanças nos requisitos de um software

Aspectos de Gerência de Requisitos

- Analisar o problema
- Entender as necessidades dos usuários
- Definir o sistema
- Gerenciar o escopo
- Refinar a definição do sistema
- Gerenciar mudanças nos requisitos

Mapa do Território



Prática 3:
Arquitetura Componentizada

Boas Práticas

Desenvolvimento Iterativo

Gerência de Requisitos

Arquitetura Componentizada

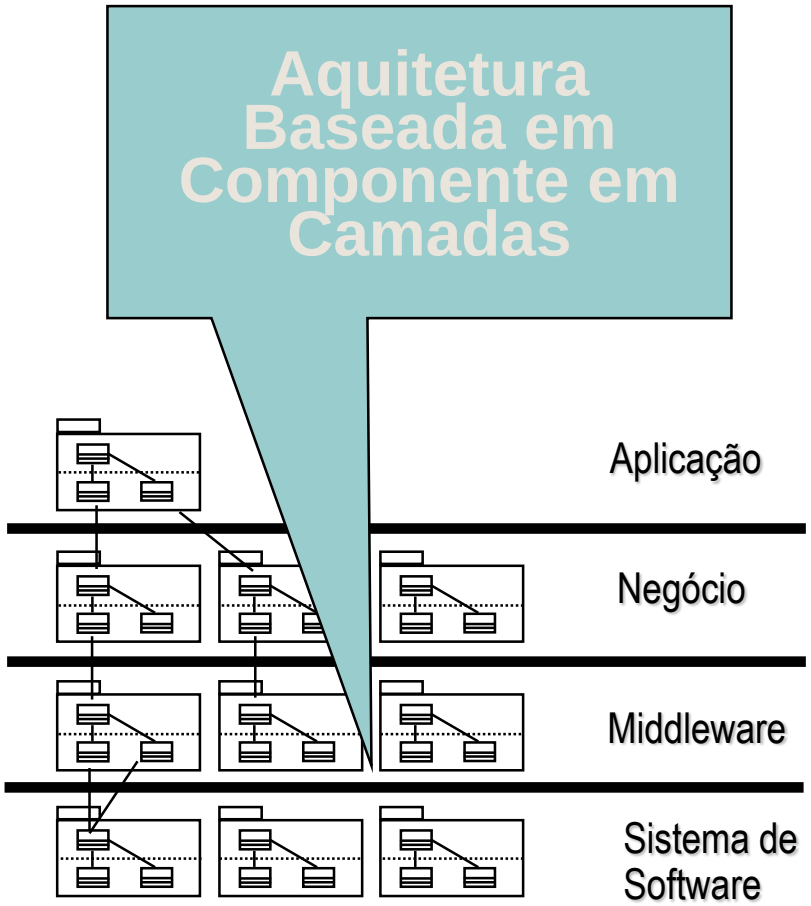
Modelagem Visual (UML)

Verificação Contínua da Qualidade

Gerência de Mudanças

Proposta de uma Arquitetura Baseada em Componentes

- Base para reuso:
  - Reuso de componente
  - Reuso de arquitetura
- Base para gerência de projeto:
  - Planejamento
  - Pessoal
  - Entrega
- Controle Intelectual
  - Gerência da complexidade
  - Manter a integridade



Resiliência e Arquitetura Baseada em Componentes

- Resiliência
  - Atende requisitos atuais e futuros
  - Melhora a extensibilidade do sistema
  - Permite o reuso
  - Encapsula as dependências do sistema
- Baseado em Componentes
  - Componentes reusáveis ou customizáveis
  - Componentes disponíveis comercialmente
  - Envolver software existente incrementalmente

Prática 4:
Modelagem Visual (UML)

Boas Práticas

Desenvolvimento Iterativo

Gerência de Requisitos

Arquitetura Componentizada

Modelagem Visual (UML)

Verificação Contínua da Qualidade

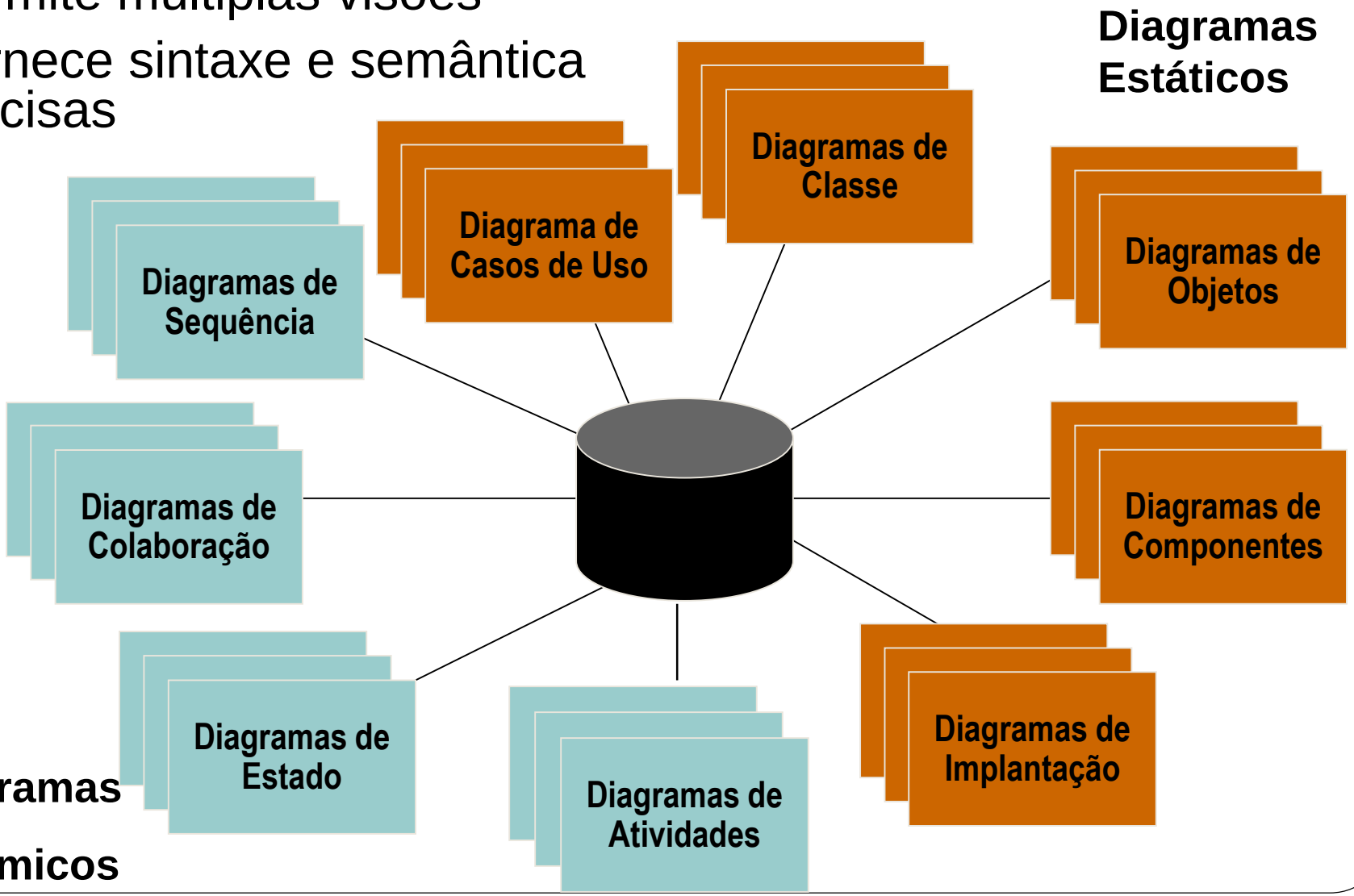
Gerência de Mudanças

Porque Modelar Visualmente?

- Para:
  - Capturar a estrutura e comportamento do sistema
  - Mostrar como os elementos do sistema se encaixam
  - Manter o projeto e implementação consistentes
  - Esconde e expor detalhes de forma apropriada
  - Promover comunicação sem ambigüidade
    - UML fornece um linguagem padrão para todos os envolvidos

Modelagem Visual com UML

- Permite múltiplas visões
- Fornece sintaxe e semântica precisas



Modelagem Visual com UML

Diagrama de Casos de Uso

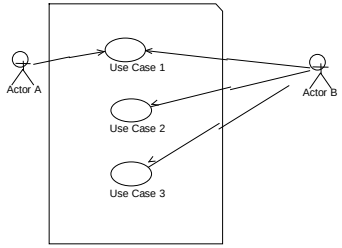


Diagrama de Classes

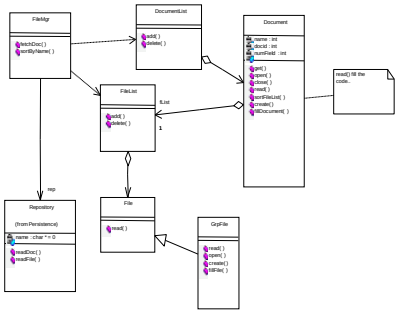


Diagrama de Estados

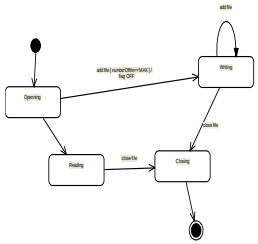


Diagrama de Implantação

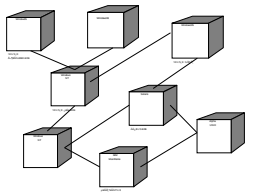
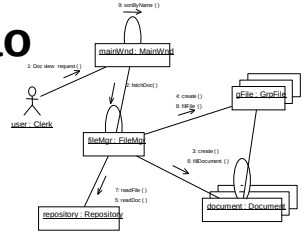


Diagrama de Colaboração



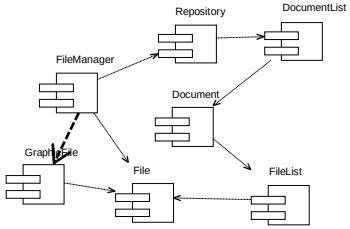


Diagrama de Componentes

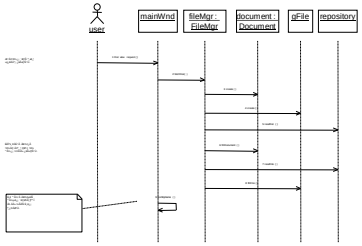
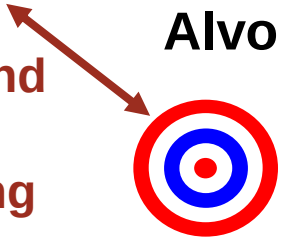


Diagrama de Sequência

Sistema Alvo

Forward and Reverse Engineering



Prática 5:
Verificação Contínua da Qualidade

Boas Práticas

Desenvolvimento Iterativo

Gerência de Requisitos

Arquitetura Componentizada

Modelagem Visual (UML)

Verificação Contínua da Qualidade

Gerência de Mudanças

Verificação Contínua da Qualidade

Problemas com software são de 100 a 1000 vezes mais caro para encontrar e reparar depois do desenvolvimento

- Custo para reparar o software
- Custo de perda de oportunidades
- Custo de perda de clientes

Dimensões de Teste de Qualidade

Usabilidade

- Testar a aplicação a partir da perspectiva de conveniência do usuário final.

Funcionalidade

- Testar a precisão do funcionamento de cada cenário de uso

Confiabilidade

- Testar que aplicação se comporta de forma consistente e previsível.

Suportabilidade

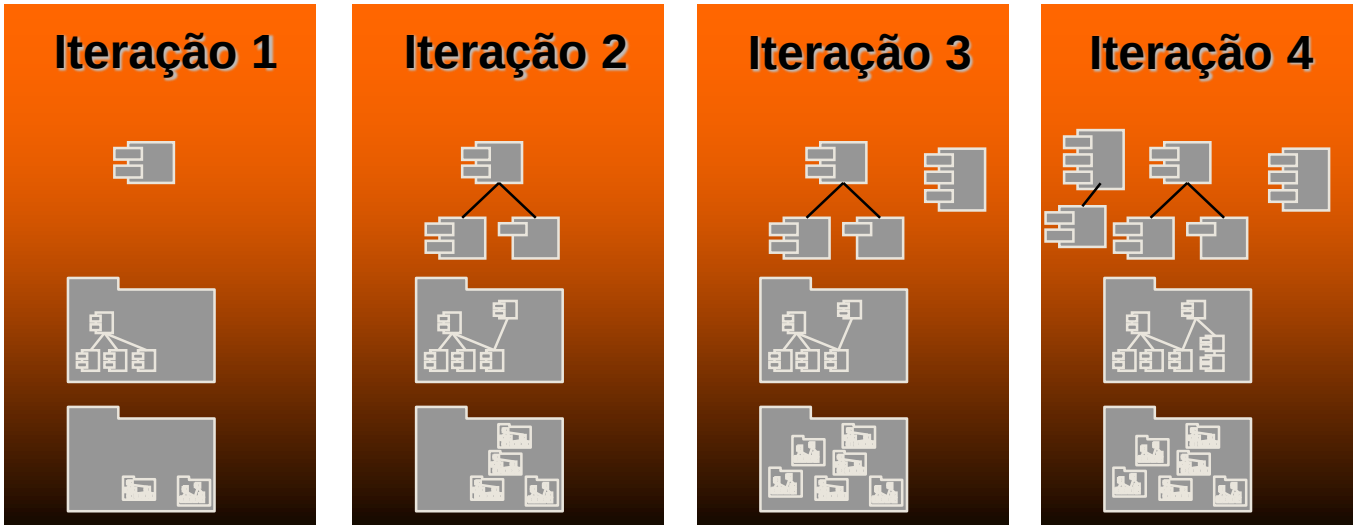
- Testar a habilidade de manter e suportar a aplicação em uso na produção.

Desempenho

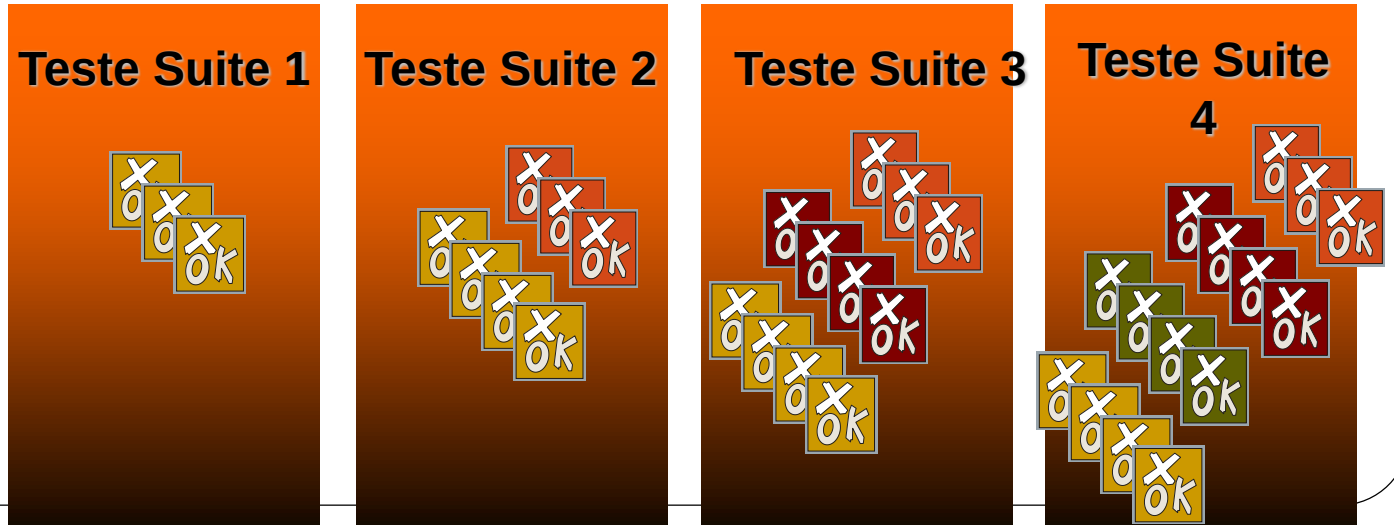
- Testar a resposta online sob a média ou pico de carga.

Teste em Cada Iteração

Modelos UML e Implementação



Testes



Prática 6:
Gerência de Mudanças

Boas Práticas

Desenvolvimento Iterativo

Gerência de Requisitos

Arquitetura Componentizada

Modelagem Visual (UML)

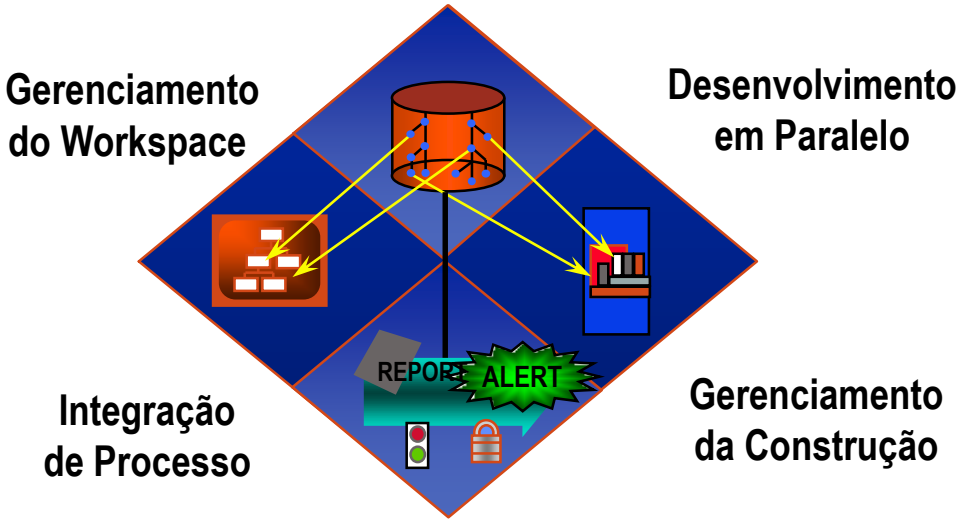
Verificação Contínua da Qualidade

Gerência de Mudanças

O Que Você Deseja Controlar?

- Workspaces seguros para cada desenvolvedor
- Gerenciamento automáticos de integração e construção
- Desenvolvimento em paralelo

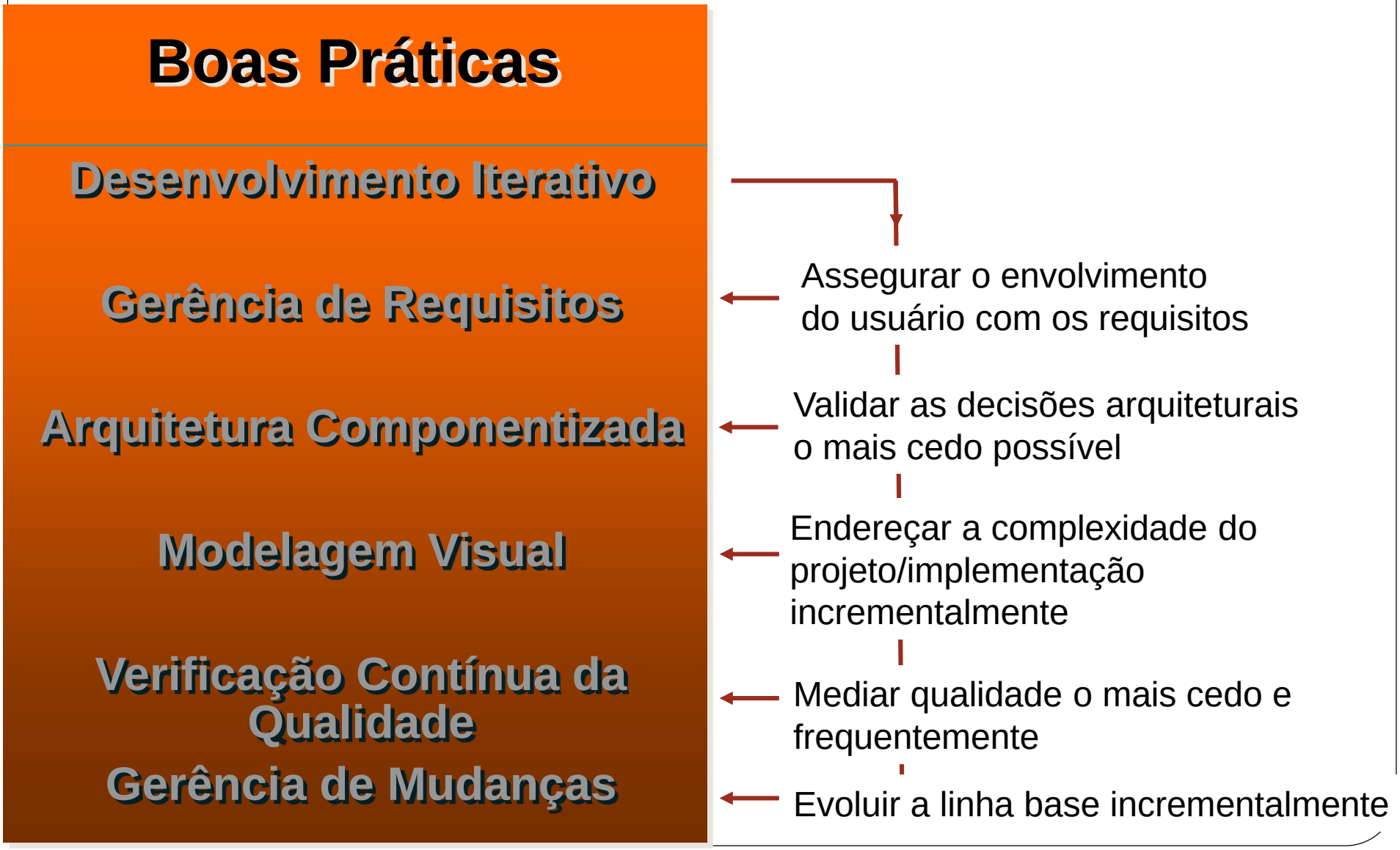
Gerência de Configuração é mais do que apenas *check-in* e *check-out*.



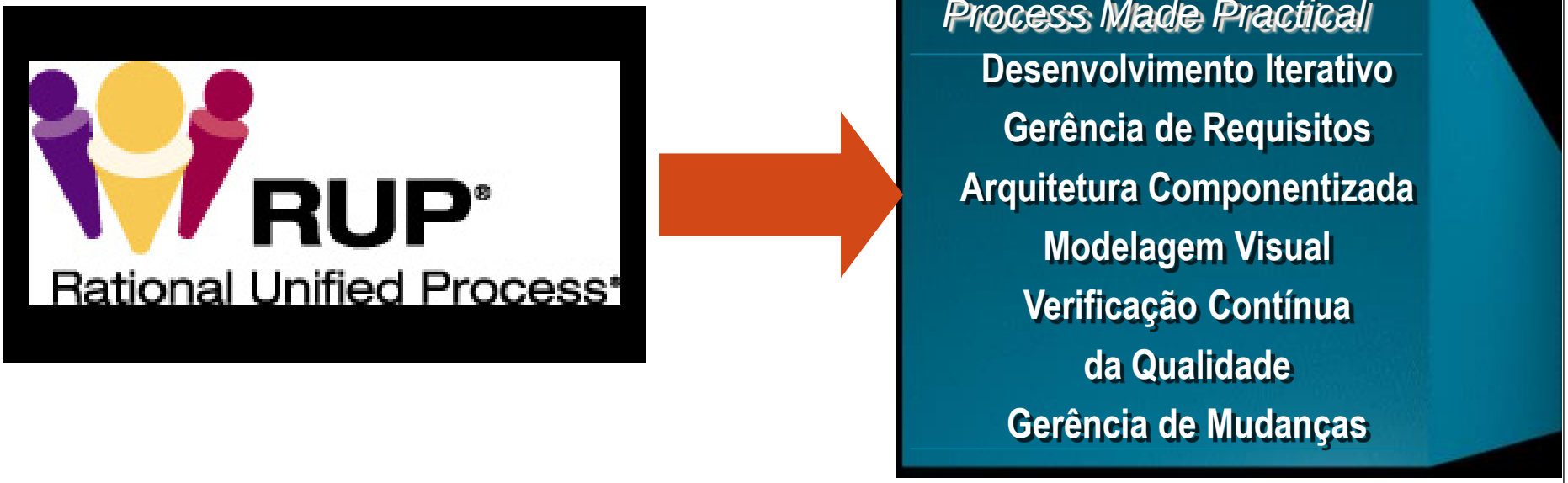
Aspectos da Gerência de Mudanças

- Gerencia de Requisições de Mudanças (CRM)
- Relatório do Status da Configuração
- Gerenciamento de Configuração
- Rastreamento de Mudanças
- Seleção de Versão
- Manufatura do Software

Boas Práticas Suporta Uma a Outra

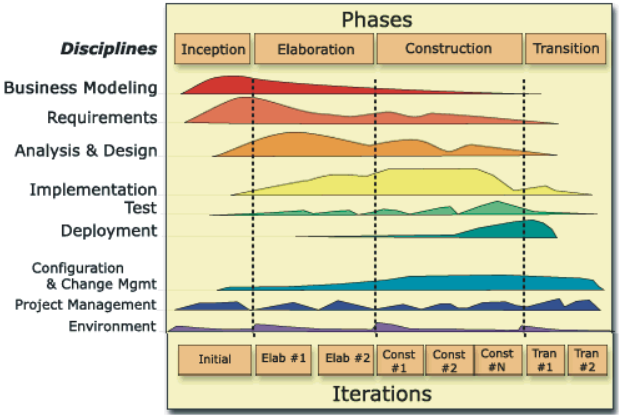


RUP Implementa as Boas Práticas



Alcançando as Boas Práticas com o RUP

- Abordagem iterativa
- Guiado por atividade e artefatos
- Processo focado na arquitetura
- Use cases "dirige" o projeto e implementação
- Modelos abstraem o sistema



Estrutura do RUP