PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

Processo de Desenvolvimento de Software

Prof. Carlos Eduardo de B. Paes

carlosp@pucsp.br

Aula 01

Agenda

- Apresentação da Disciplina
- Referências Bibliográficas
- Contextualização: Falhas famosas de software e crise do software
- Processo de Desenvolvimento de Software
- Definições: processo, método e metodologia
- O Processo RUP
- RUP – Melhores Práticas
- As Filosofias do RUP

Apresentação da Disciplina
Objetivos

- Objetivos da disciplina:
  - Conduzir o estudante a dominar os conceitos relacionados a processo de desenvolvimento de software
  - Apresentar as melhor práticas de engenharia de software
  - Proporcionar reflexões sobre Métodos Ágeis X RUP
  - Apresentar a estrutura do RUP, as suas linhas gerais, as fases e os workflows descritos no processo

Apresentação da Disciplina
Objetivos

- Objetivos da disciplina:
  - Discutir alguns estudos de caso usando o RUP
  - Customização e implantação do RUP
  - Engenharia de Processo de Software
    - OMG SPEM
    - RUP 2003 x RUP 2007 (UMA)

Apresentação da Disciplina
Avaliação

- Avaliação:
  - Prova final (última aula)
  - Atividades em sala de aula
  - Aulas práticas em laboratório (RUP): exploração do processo; customização do RUP; ferramentas de apoio ao RUP
  - **Leitura recomendada de artigos, livros, revistas e etc.**

Referências Bibliográficas

- **Kruchten,P.,The Rational Unified Process: An Introduction. Wokingham, UK: Addison-Wesley, 1998**
- Jacobson,I., Booch, G., Rumbaugh, J., **The Unified Software Development Process**, Wokingham, UK: Addison-Wesley, 1999
- AMBLER, S. W.; NALBONE, J.; VIZDOS, V. **The enterprise unified process: extending the rational unified process**. Englewood Cliffs: Prentice Hall PTR, 2005. ISBN: 0-13-191451-0
- **KROLL, P.; KRUCHTEN, P. The rational unified process made easy: a practitioner's guide to the RUP. Wokingham: Addison Wesley, 2003**
- Humphrey, W. S., Managing the Software Process , SEI series in Software Engineering, 1989.
- Craig Larman ,**Agile & Iterative development: A Manager´s Guide**,, 2004

Crise do software

- Conferência da OTAN sobre Engenharia de Software (NATO Software Engineering Conference) em Garmisch, Alemanha (F. L. Bauer)
- Objetivo: Resolver a "Crise do Software"
- O estabelecimento e uso dos princípios da Engenharia a fim de obter software de baixo custo que seja confiável e trabalhe com eficiência em máquinas reais (Fratz Bauer, 1969)

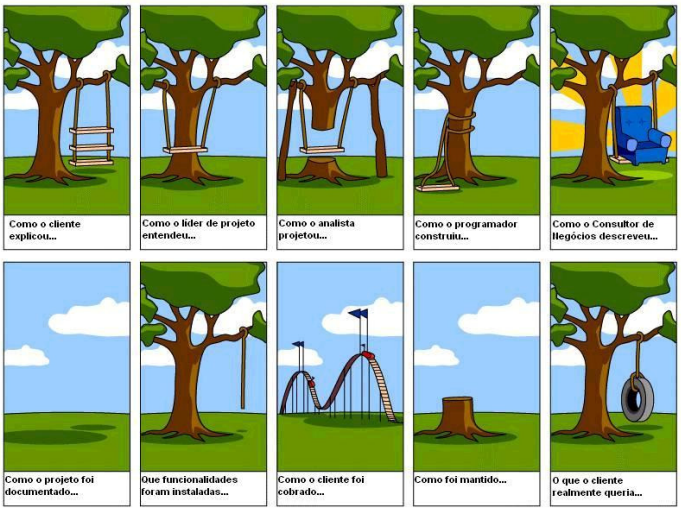
Falhas Famosas de Software

- 1979 – Comando Estratégico do Ar
  - Iniciou contra-ataque à URSS
  - Simulação interpretada como ataque real
- 1991 – Guerra do Golfo
  - 28 mortos, 98 feridos
  - Sincronia dos mísseis Patriot
- 1985-87 Therac – 25 (Sistema Médico)
  - Pelo menos 2 pacientes mortos por radiação
  - Software de controle de dose

Falhas Famosas de Software

- 1992 - London Ambulance System - despacho de ambulâncias em Londres.
  - Morte de pessoas que não foram socorridas em tempo.
- 1993 – Aeroporto internacional de Denver (EUA)
  - Sistema de triagem/controle de bagagem do aeroporto
  - Atrasou a inauguração do aeroporto. Custo do sistema: US$ 193 milhões
  - Inauguração estava prevista para Out/1993.
  - Em Junho/1994 o sistema ainda não estava funcionando e causava prejuízos de US$ 1,1 milhão/dia
- 1996 - Ariane 5
  - O foguete explodiu 40 segundos após a sua primeira decolagem
  - Prejuízo de U$ 500 milhões

A velha figura.....



Projetos de Software

- 1995 - The Standish Group publicou um estudo analisando as estatísticas sobre sucesso e fracasso dos projetos de desenvolvimento de software: o Chaos Report
  - 84% dos projetos de software são mal-sucedidos, sejam sendo cancelados ou apresentando falhas críticas (dentre elas conclusão fora da janela de tempo prevista, fora do orçamento previsto ou com menos funcionalidades do que o planejado)
  - Considerando apenas os projetos mal-sucedidos, o custo real foi 189% maior que o estimado, e o tempo de conclusão 222% maior
- 2006 - 35% dos projetos de software iniciados em 2006 terem obtido sucesso, ainda é assustador saber que dois terços de todos eles fracassam

Projetos de Software

CHAOS Report 2009, Standish Group

http://www.standishgroup.com/

	1994	1996	1998	2000	2002	2004	2006	2009
Successful	16%	27%	26%	28%	34%	29%	**35%**	**32%**
Challenged	53%	33%	46%	49%	51%	53%	46%	44%
Failed	31%	40%	28%	23%	15%	18%	19%	24%

http://www.projectsmart.co.uk/the-curious-case-of-the-chaos-report-2009.html

challenged (late, over budget and/or with less than the required features and functions)

failed (cancelled prior to completion or delivered and never used)

Problemas no Desenvolvimento de Software
Sintomas e Causas Raízes

- Quais são alguns dos principais sintomas de problemas no desenvolvimento de software?
- Quais causas raízes geram esses sintomas?

Sintomas dos Problemas em Desenvolvimento de Software

- Necessidades de negócio e de usuários não são atendidos
- Requisitos confusos
- Módulos não se integram
- Difícil de manter
- Descoberta tardia de falhas
- Qualidade ruim ou falta de experiência dos usuários final
- Problemas de performance
- Sem coordenação nos esforços do time
- Problemas na construção e entrega

Boas Práticas (RUP)

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

Sintomas X Causas Raízes X Boas Práticas

Sintomas	Causas Raízes	Boas Práticas
Necessidades não fecham	Requisitos insuficientes	Desenvolvimento iterativo
Requisitos confusos	**Comunicação ambígua**	Gerenciamento de requisitos
Módulos não se integram	Arquitetura frágil	Arquitetura componentizada
Difícil de manter	Complexidade subjugada	**Modelagem visual (UML)**
Descobertas tardes	**Inconsistências**	**Verificação contínua**
Qualidade ruim	**não detectadas**	**da qualidade**
Performance ruim	Testes pobres	Gerência de mudanças
Confusão no time	Avaliação subjetiva	
Construção e entrega	Desenvolvimento em cascata	
	Mudanças sem controle	
	Automação insuficientes	

Prática 1:
Desenvolvimento Iterativo

Boas Práticas

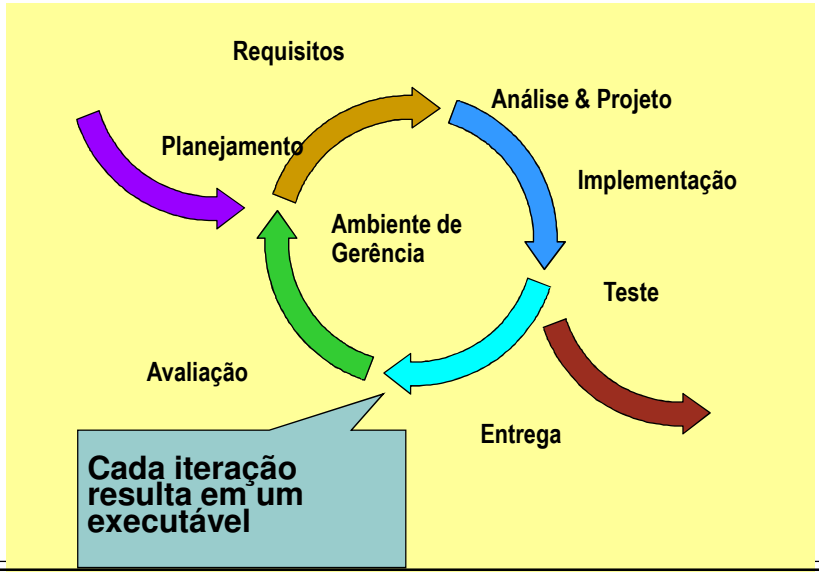
Desenvolvimento Iterativo
Gerência de Requisitos
Arquitetura Componentizada
Modelagem Visual (UML)
Verificação Contínua da Qualidade
Gerência de Mudanças

Desenvolvimento em Cascata

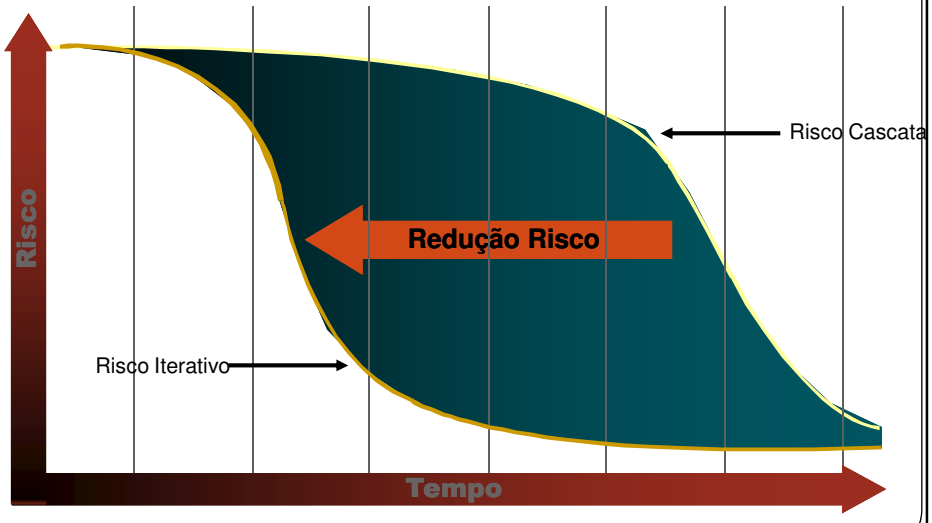
Processo Cascata

- 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 de agregado e testando
- 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

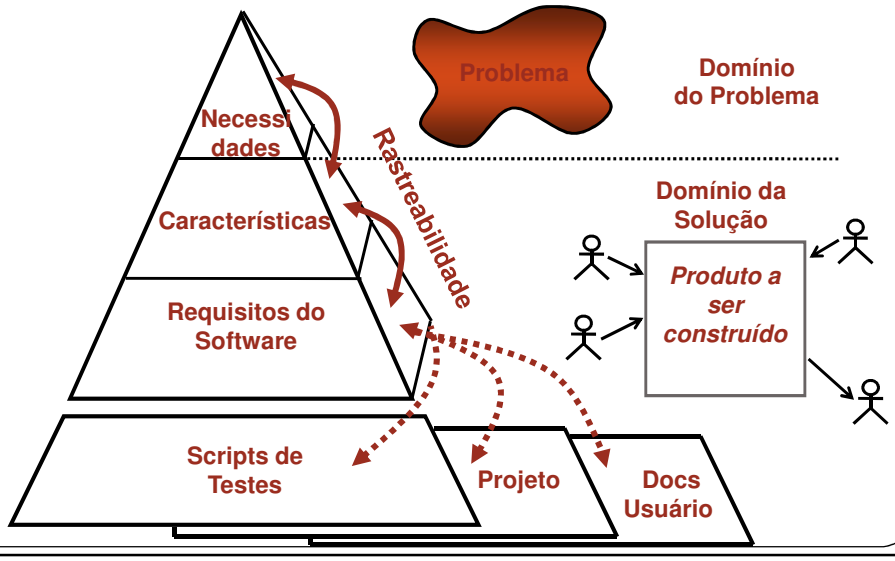
Gerência de Requisitos Significa...

- 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



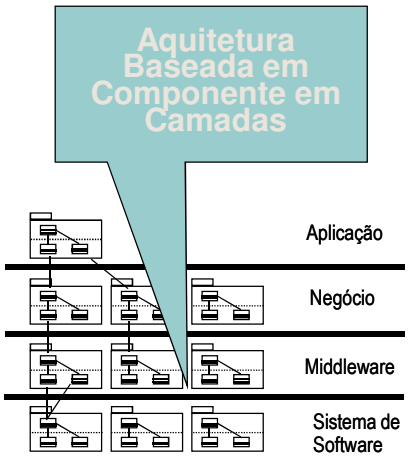
Prática 3:
Arquitetura Componentizada

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

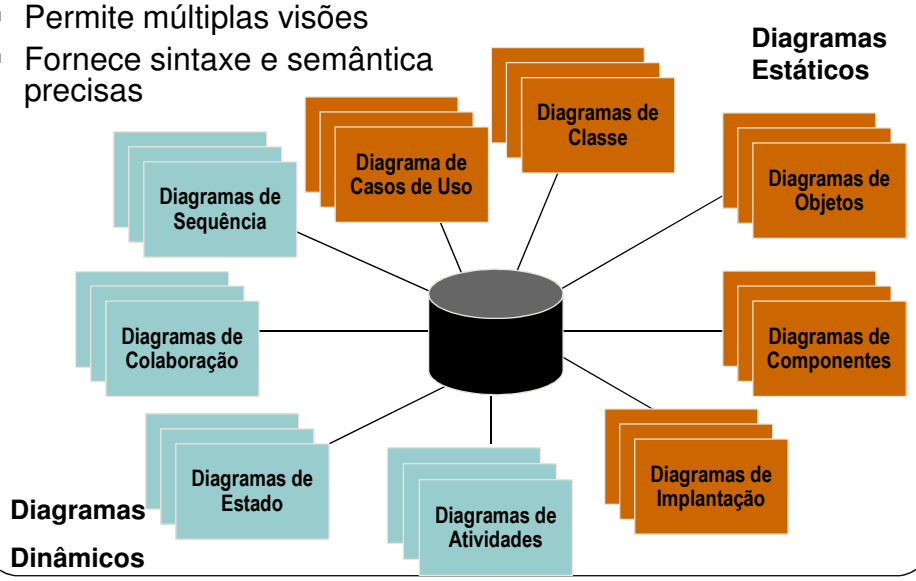
Desenvolvimento Iterativo
Gerência de Requisitos
Arquitetura Componentizada
Modelagem Visual (UML)
Verificação Contínua da Qualidade
Gerência de Mudanças

Porque Modelar Visualmente?

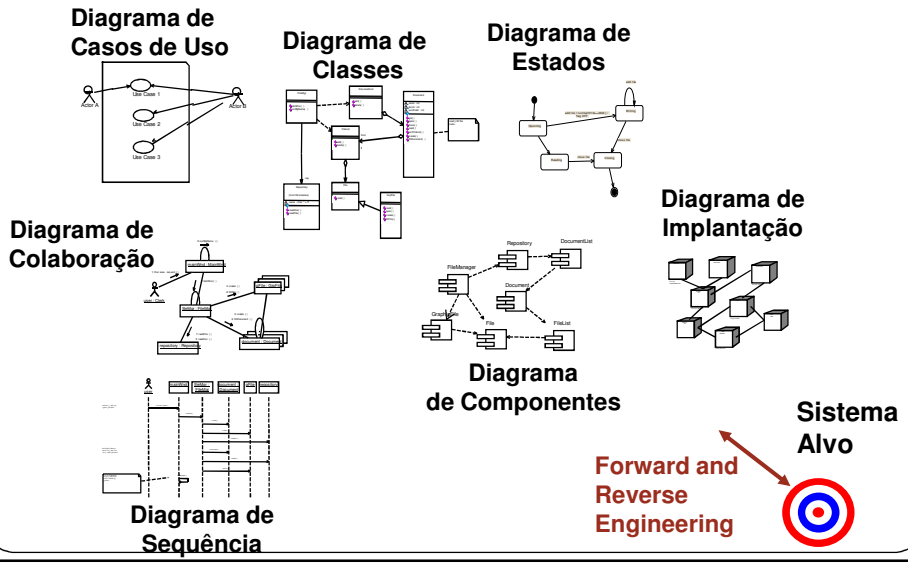
- Para:
  - Capturar a estrutura e comportamento do sistema
  - Mostrar como os elementos do sistema se encaixam
  - Manter o projeto e implementação consistentes
  - Esconde e expor detalhes de forma apropriada
  - Promover comunicação sem ambigüidade
    - UML fornece um linguagem padrão para todos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

Modelagem Visual com UML

- Permite múltiplas visões
- Fornece sintaxe e semântica precisas



Modelagem Visual com UML



Prática 5:
Verificação Contínua da Qualidade

Boas Práticas

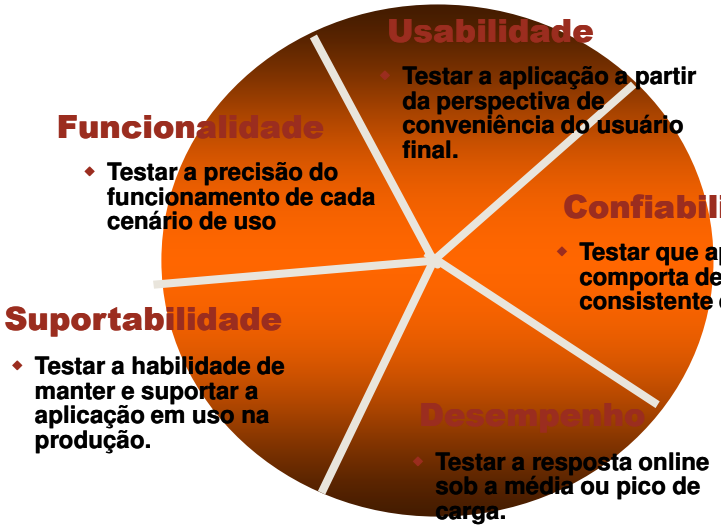
Desenvolvimento Iterativo
Gerência de Requisitos
Arquitetura Componentizada
Modelagem Visual (UML)
Verificação Contínua da Qualidade
Gerência de Mudanças

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.

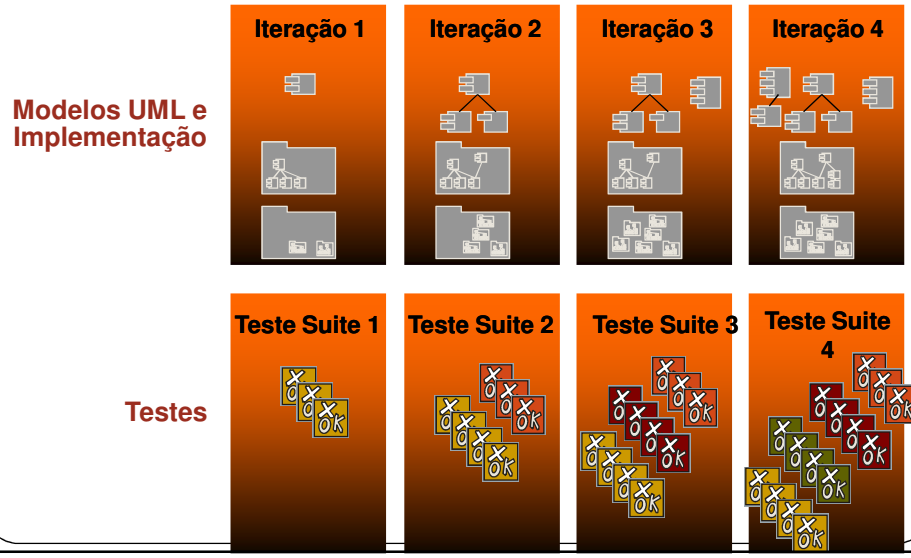
Funcionalidade
- Testar a precisão do funcionamento de cada cenário de uso

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



Prática 6: Gerência de Mudanças

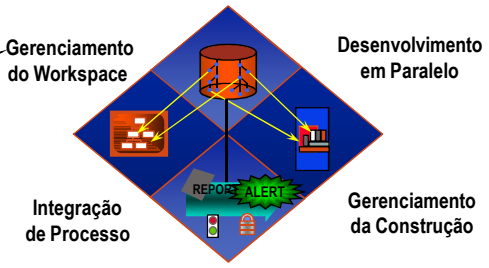
Boas Práticas

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

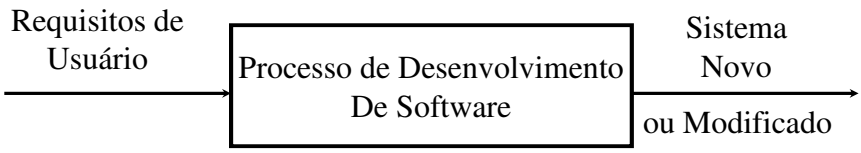
- Gerencia de Requisições de Mudanças (CRM)
- Relatório do Status da Configuração
- Gerenciamento de Configuração
- Rastreamento de Mudanças
- Seleção de Versão
- Manufatura do Software

Boas Práticas Suporta Uma a Outra

Processo de Desenvolvimento de Software

- Bom sistema [Jacobson92] :
  - Do ponto de vista do usuário: deve ser correto, rápido, confiável, fácil de ser usado, eficiente, etc.
  - Do ponto de vista dos desenvolvedores: deve ser fácil de ser entendido, modificado, reutilizável, compatível com outros sistemas, portável, etc.
  - Varia em função da aplicação

Processo de Desenvolvimento de Software



"Um processo é o conjunto total de atividades de engenharia necessárias para transformar requisitos do usuário em software"

"Managing the Process", Humphrey, 1989

Processo de Desenvolvimento de Software
Algumas Definições

- "Um **processo de software** é um método para desenvolver ou produzir software"
- "Define quem faz o que, quando e como"
- "Conjunto coerente de atividades para especificar, projetar, implementar e testar sistemas de software"

Processo de Desenvolvimento de Software
Características de um Processo

- Atividades principais
- Recursos
- Produtos intermediários e finais
- Subprocessos, com hierarquia ou organizados de algum modo
- Critérios de entrada e saída para cada atividade
- Seqüência de atividades, de modo que a ordem de execução de uma para outra seja clara
- Conjunto de diretrizes que explicam os objetivos de cada atividade
- Restrições e controles para cada atividade, recurso ou produto

Processo de Desenvolvimento de Software
Características de um Processo

- Quando um processo envolve a construção de um produto → nos referimos a um processo como um ciclo de vida (modelo de ciclo de vida)
- Processo de desenvolvimento de software → pode ser chamado de ciclo de vida do software
- **Um processo deve ser adequado ao domínio da aplicação e ao projeto específico. Deve ser considerado a tecnologia, a organização e o grupo de desenvolvimento**

Processo de Desenvolvimento de Software
Modelo X Processo

- Um modelo de processo de software é uma representação abstrata de um processo de software.
- O processo deve determinar ações práticas a serem realizadas pela equipe como prazos definidos e métricas para se avaliar como elas estão sendo realizadas

Modelo + Planejamento = Processo

Processo de Desenvolvimento de Software
Modelos de Processo

- Um modelo de processo ou método define um conjunto de atividades específicas.
- Principais modelos:
  - Cascata (Waterfall)
  - Espiral (Spiral)
  - Evolutivo e Incremental
  - Desenvolvimento formal de sistemas
  - Desenvolvimento orientado a reuso
  - Iterativo e Incremental
  - …

Hoje...

- UP (Unified Process)
- RUP (Rational Unified Process)
- XP (eXtreme Programming)
- Scrum
- FDD (Feature Driven Development)
- DSDM (Dynamic Systems Development Method)
- ICONIX
- OPEN/UP
- …

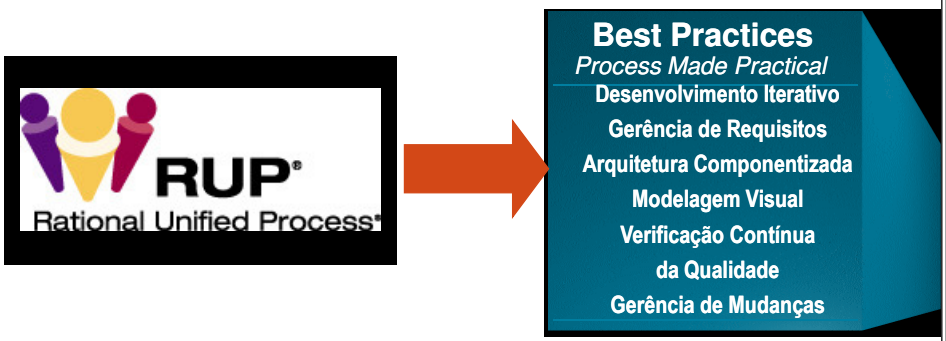
Processo, Método e Metodologia

- Fonte: Wikipédia
  (http://pt.wikipedia.org/wiki/Metodologia_%28engenharia_de_software%29)
- Uma metodologia é um conjunto estruturado de práticas que pode ser repetível durante o processo de produção de software
- Metodologias de Engenharia de Software abrangem muitas disciplinas, incluindo Gerenciamento de Projetos, e as suas fases como: análise, projeto, codificação, teste, e mesmo a Garantia da Qualidade
- Método X metodologia → ainda existe muita discussão !!

Processo, Método e Metodologia

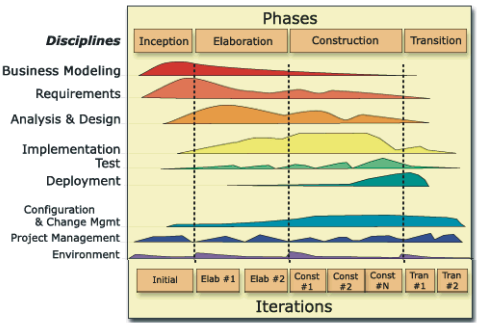
- Uns argumentam que método é um processo com uma série de passos, para construir um software, enquanto que uma metodologia é a codificação de um conjunto de práticas recomendadas
- Método de Engenharia de Software → pode ser considerado como parte da metodologia. Também, alguns autores acreditam que uma metodologia exista com base em uma abordagem filosófica do problema. Utilizando-se dessas definições, pode-se afirmar que a Engenharia de Software é rica em métodos, mas com poucas metodologias

RUP Implementa as Boas Práticas



Alcançando as Boas Práticas com o RUP

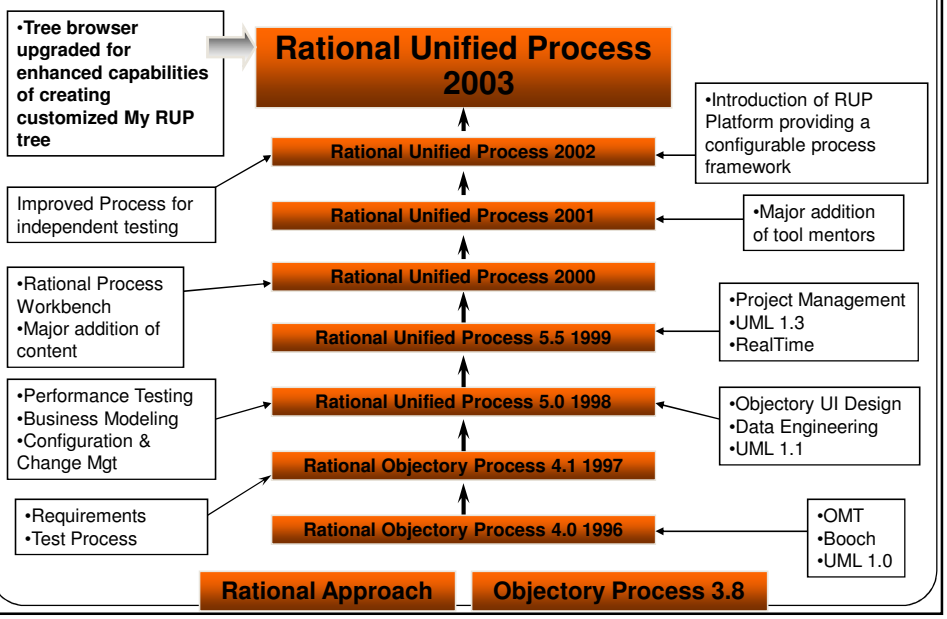
- Abordagem iterativa
- Guiado por atividade e artefatos
- Processo focado na arquitetura
- Use cases "dirige" o projeto e implementação
- Modelos abstraem o sistema



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



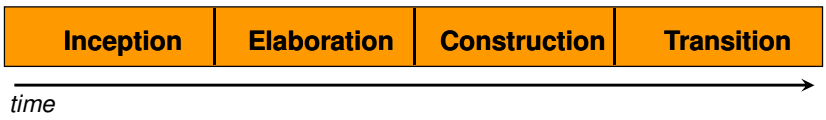
UML e RUP

- 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

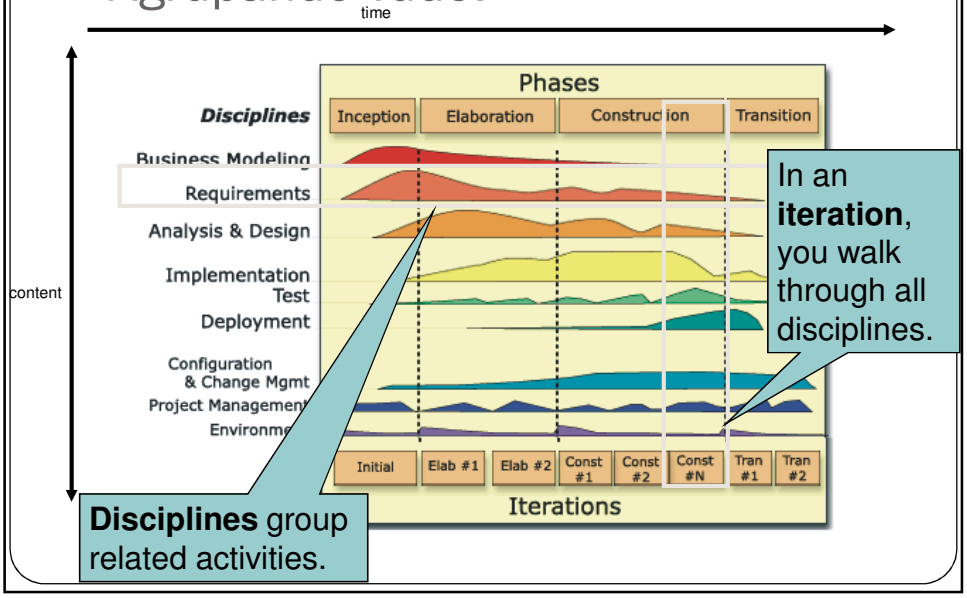


- RUP possui quatro fases:
  - **Concepção** (*Inception*): define o escopo do projeto
  - **Elaboração** (*Elaboration*): Plano do projeto, características específicas, linha-base da arquitetura
  - **Construção** (*Construction*): Construção do produto
  - **Transição** (*Transition*): transição do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