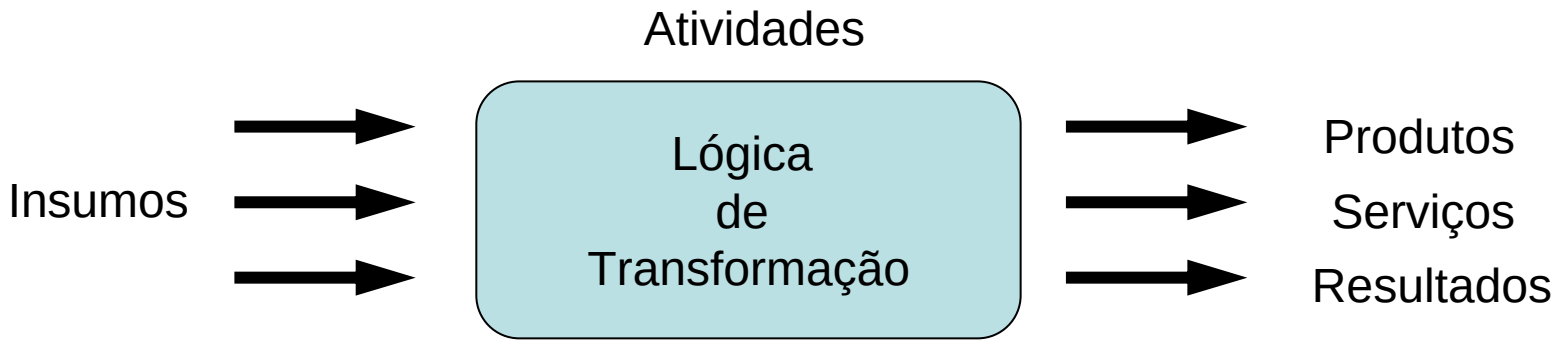
Processos de Software: Conceitos Básicos

Agenda

- *Definição de Processos*
- *Objetivos*
- *Componentes*
- *Conceitos Básicos*
- *Exemplos*

Processo

- *O que é?*
  - Um conjunto de atividades que recebem insumos, transformando-os, de acordo com uma lógica pré-estabelecida e com agregação de valores, em produtos / serviços para responderem às necessidades dos clientes / usuários.



Processo de Desenvolvimento

- *Não confundir com processo de negócio!*

- *Processo de Negócio:*
  - Um grupo de atividades relacionadas de forma lógica que usa os recursos da organização para fornecer resultados definidos em apoio aos objetivos da organização. Foco nos objetivos estratégicos da organização.

- *Processo de Desenvolvimento:*
  - Um conjunto de passos ordenados e executados com o objetivo de alcançar uma meta. No caso de desenvolvimento de software, a meta consiste em criar um software ou desenvolver um já existente.
  - Pode ser visto como um processo de negócio aplicado a objetivos específicos.

Processo de Desenvolvimento

- *Do que é composto?*
  - Um processo de desenvolvimento de software possui 4 etapas básicas:
    - *Especificação*
      - *definição das funcionalidades do software e premissas para sua execução*
    - *Projeto*
      - *construção do software de acordo com a especificação*
    - *Validação*
      - *validação do software para verificar se ele atende as necessidades dos usuários*
    - *Evolução*
      - *evolução do software de modo a atender as modificações das necessidades dos usuários*

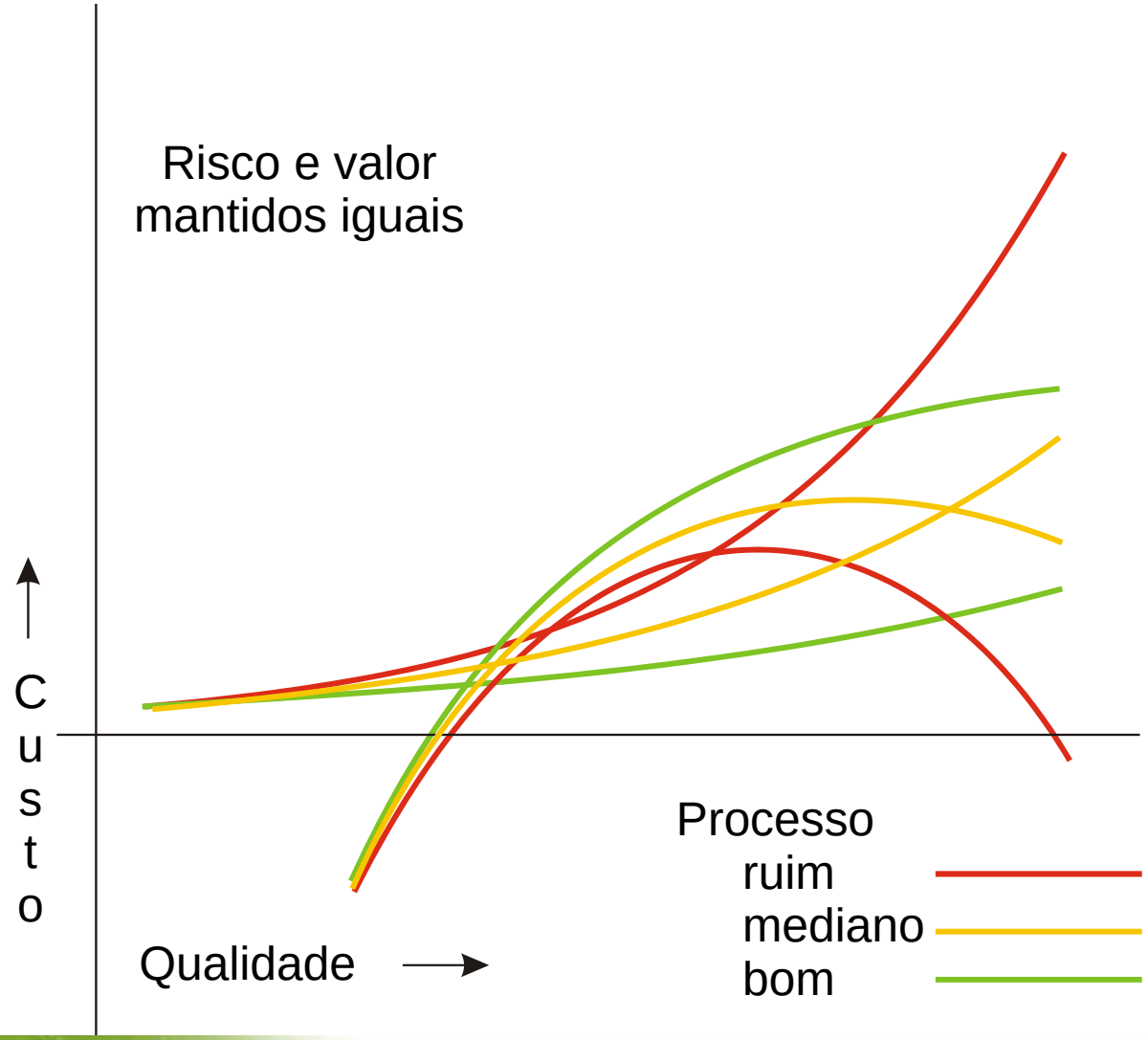
Objetivos

- *Processos de software visam assegurar o desenvolvimento de software:*
  - com prazos e necessidade de recursos definidos
  - com elevada produtividade (de forma econômica)
  - com qualidade assegurada
- *Processos permitem*
  - organizar
  - instrumentar
  - planejar
  - acompanhar projetos
  - treinar equipes

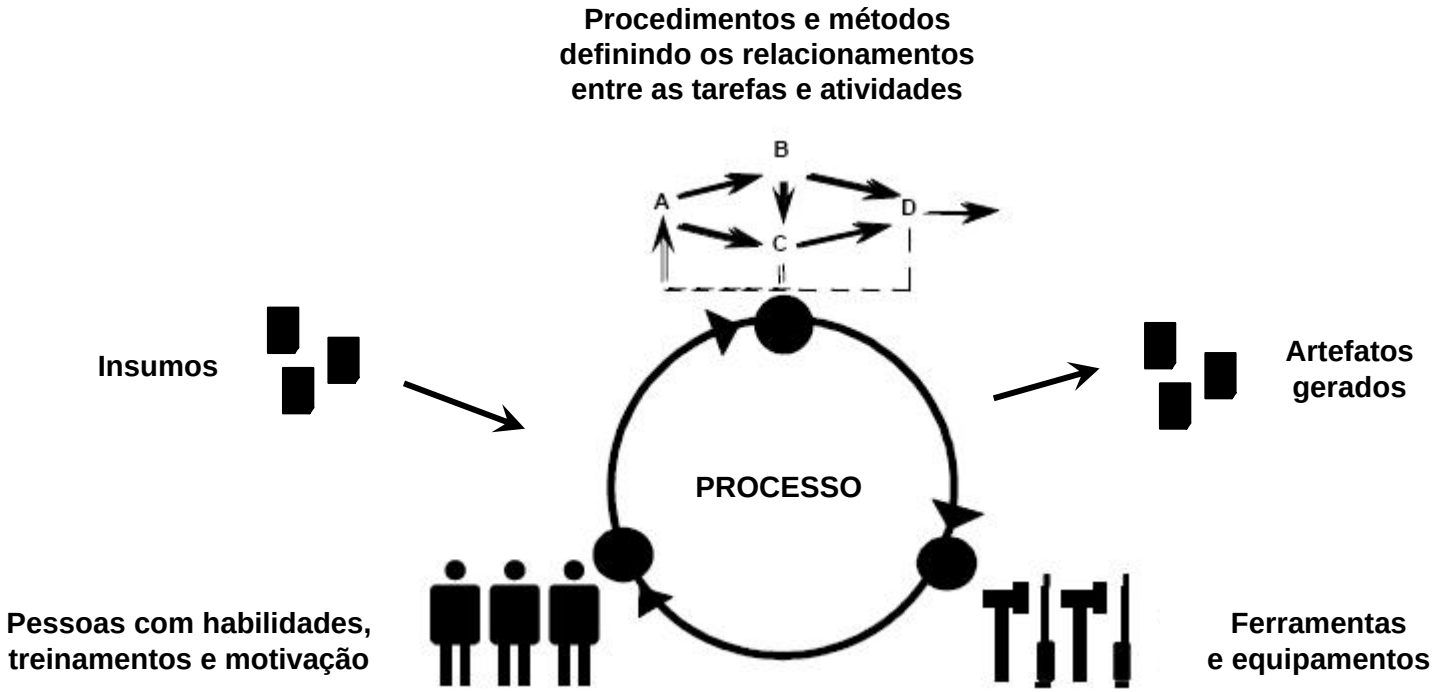
Processo de Desenvolvimento

- *Sem o uso de processos....*
  - Procedimentos existentes na organização não são documentados e usados de forma consistente na prática
  - Erros são cometidos repetidamente
  - Dificuldade de prever cronogramas e orçamentos
  - Alto índice de defeitos, retrabalho e desperdício
  - Dificuldade de implementar boas práticas e lições aprendidas
  - Dificuldade de realizar ações para prevenção de defeitos

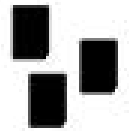
Efeito do Processo sobre o Custo



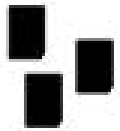
Componentes de Processos



Conceitos Básicos

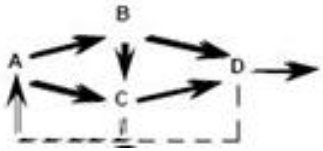


- *Artefato*
  - é um resultado de uma atividade, exemplos
    - *documento revisto e aceito*
    - *módulo implementado, testado e aceito*
    - *construto integrado, testado e aceito*
    - framework *documentado, implementado, testado e aceito*
  - quando entregue ao usuário (cliente) o artefato é um produto



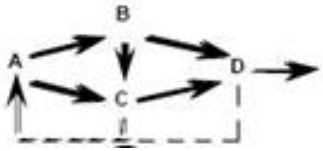
- *Insumo*
  - elemento necessário para a realização de uma tarefa ou atividade
  - pode ser um elemento de saída de outras atividades ou tarefas

Conceitos Básicos



- *Tarefa*
  - é uma ação desempenhada por alguma pessoa visando a realização ou monitoramento do projeto
    - *não representa uma evidência de progresso no desenvolvimento*
      - *ter trabalhado 20 horas não implica ter produzido um artefato de qualidade, mesmo que se tenha estimado serem necessárias 20 horas para o seu desenvolvimento*
    - *consome recursos - consumo real*
      - *esforço (tempo de pessoa)*
      - *equipamento*
      - *financeiro*

Conceitos Básicos



- *Atividade*
  - conjunto de tarefas que levam a um ou mais artefatos de qualidade controlada
    - *representa uma evidência de progresso no desenvolvimento*
      - *os artefatos resultantes existem e podem ser usados*
    - *permite o controle da qualidade do resultado*
      - *num extremo pode ser a mera constatação que o resultado existe*
      - *no outro extremo pode envolver técnicas muito avançadas de controle da qualidade*
    - *o esforço é medido através das tarefas constituintes*

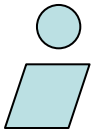
- *Atividades são "mini-projetos"*
  - possuem início e fim definidos
  - consomem um volume finito de recursos
  - produzem artefatos definidos
  - possuem critérios de conclusão estabelecidos

Conceitos Básicos

- *Ferramentas e equipamentos*
  - auxiliam a execução das atividades e tarefas dos processos
  - podem automatizar partes da execução das atividades e tarefas
  - agilizam a execução dos processos

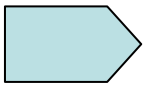
- *Papel*
  - descreve como as pessoas se comportam no processo e quais são as responsabilidades que elas têm
  - requer habilidades específicas necessárias
  - **papéis não são pessoas**
    - *pessoas executam papéis*

Exemplo de Atividades Relacionadas a Papel

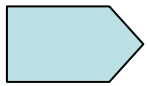


Analista de Testes

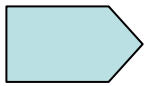
Tarefas



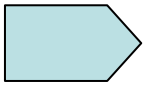
Identificar objetivos do teste



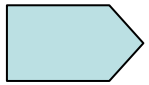
Identificar idéias do teste



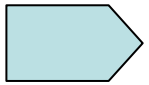
Definir detalhes do teste



Definir necessidades de avaliação e rastreabilidade

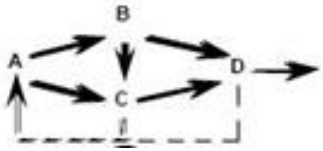


Determinar resultados do teste



Verificar mudanças no build

Exemplo de Atividade



- *Atividade:* ***Realizar estimativas para o projeto***
- *Descrição*
  - A partir do escopo preliminar do projeto, detalhar as atividades necessárias a sua realização, as estimativas de consumo de recursos, os prazos e orçamentos.
- *Papel:*
  - Gerente de Projeto
- *Artefatos de Entrada:*
  - Escopo do projeto; Plano do projeto; Acordo de serviço; Estrutura analítica do projeto (EAP)
- *Artefatos de Saída:*
  - Estimativas de custo da atividades; Estimativa de esforço das atividades, Cronograma e Plano do projeto atualizados
- *Tarefas:*
  - Detalhar atividades e recursos; Detalhar cronograma; Detalhar estimativa de custo das atividades; Detalhar orçamento do projeto

Processos de Software: Conceitos Básicos

Modelagem de Processos de Software

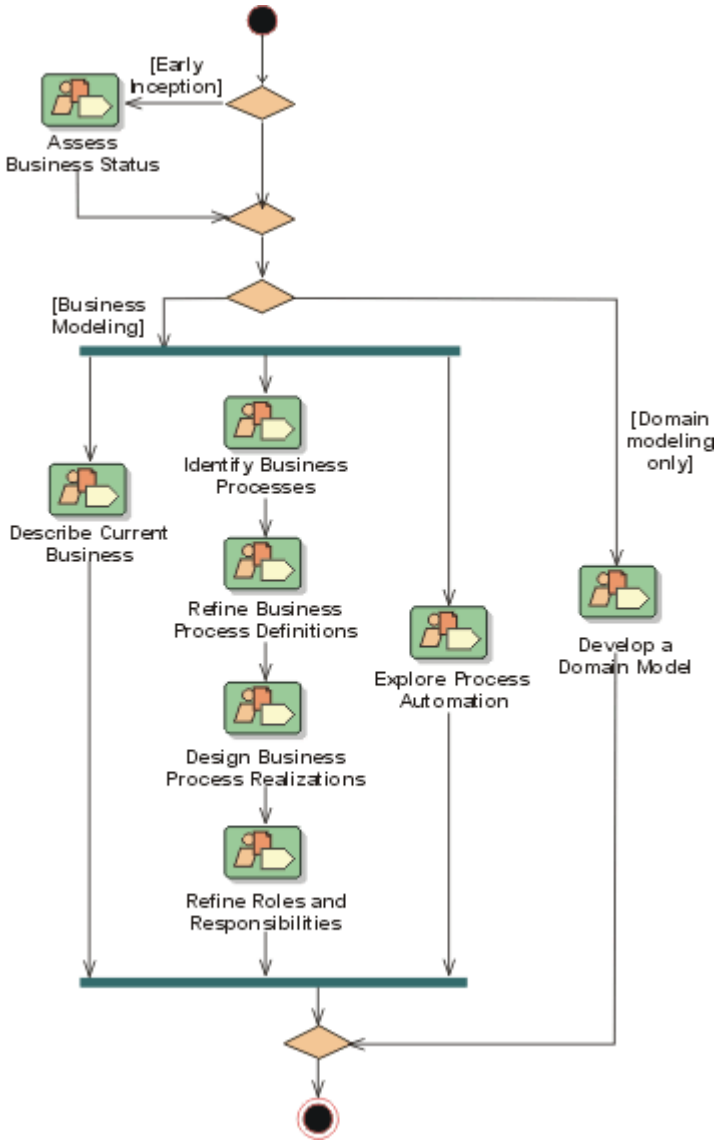
Agenda

- *Objetivos da Modelagem de Processos*
- *Exemplo*

Objetivos

- *Processos de software podem ser melhorados através da padronização dos processos utilizados dentro de uma organização.*
- *A modelagem de processos torna-se essencial para a definição de processos eficientes, capazes de serem replicados.*
- *Modelagem de processos inclui responder as seguintes perguntas:*
  - O quê?
  - Como?
  - Quem?
  - Quando?
  - Por quê?
- *Ferramentas: CFPM (Cross Functional Process Map), diagramas de blocos, fluxogramas, diagrama de atividades,...*
- *Modelos, padrões de Melhoria: ISO, SW-CMM, CMMI, MPS-BR, ...*

Exemplo


[Early Inception]
Assess Business Status
[Business Modeling]
Identify Business Processes
Describe Current Business
Refine Business Process Definitions
Explore Process Automation
[Domain modeling only]
Develop a Domain Model
Design Business Process Realizations
Refine Roles and Responsibilities

Modelagem de Processos de Software

Modelos de Ciclo de Vida

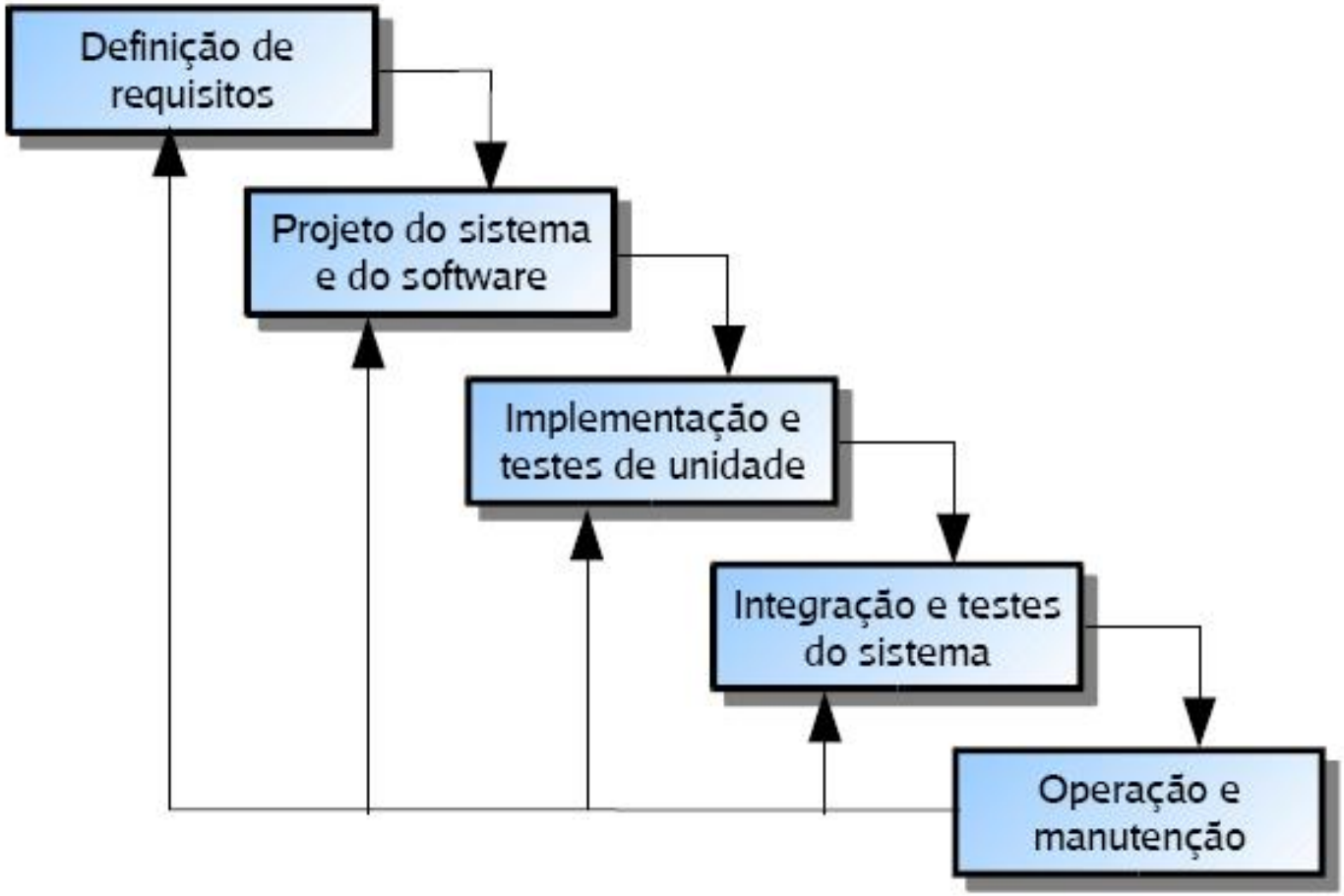
Agenda

- *Introdução*
- *Modelo Cascata*
- *Desenvolvimento Incremental*
- *Desenvolvimento Evolucionário*
- *Modelo Espiral*

Introdução

- *O que é ciclo de vida?*
  - Conjuntos de fases que devem ser executadas para o desenvolvimento de um produto de software. O ciclo de vida determina a ordem e interação entre as fases e atividades.
- *Modelos de ciclo de vida*
  - são representações abstratas de processos
  - descrevem processos a partir de uma perspectiva específica
  - podem ser vistos como frameworks de processos
  - devem ser aplicados e personalizados segundo necessidades específicas
- *Alguns exemplos*
  - Cascata
  - Evolucionário
  - Espiral

Modelo Cascata



Modelo Cascata

- *Fases*
  - Definição e análise de requisitos
  - Projeto do sistema e do software
  - Implementação e testes de unidade
  - Integração e testes do sistema
  - Operação e manutenção

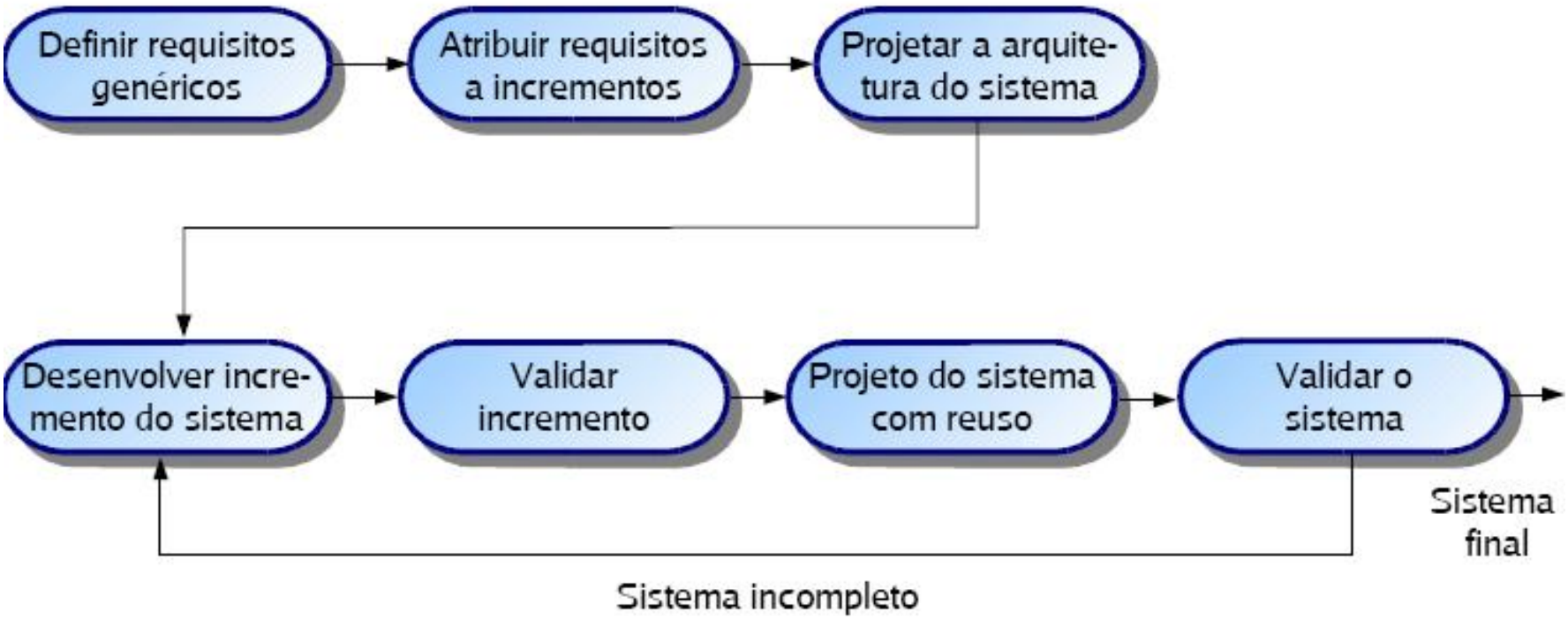
- *Desvantagens*
  - Dificuldade de acomodar as mudanças após o processo ter sido iniciado
  - Particionamento inflexível do projeto em fases distintas
  - Dificuldade de responder a requisitos do usuário que mudam

  - **Portanto, esse modelo mais apropriado quando os requisitos são bem compreendidos**

Desenvolvimento Incremental

- *Também chamado desenvolvimento iterativo.*
- *Ao invés de entregar o sistema como uma única entrega, particiona-se o desenvolvimento e a entrega em incrementos, com cada incremento contendo parte da funcionalidade requerida*
- *Os requisitos do usuário são priorizados e os requisitos de prioridade mais alta são incluídos nos incrementos iniciais*
- *Uma vez que o desenvolvimento de um incremento é iniciado, os requisitos são congelados, ainda que os requisitos para incrementos posteriores continuem a evoluir*

Desenvolvimento Incremental



Desenvolvimento Incremental

- *Vantagens*
  - *Cada incremento pode agregar valor para o cliente, portanto a funcionalidade do sistema está disponível mais cedo*
  - *Incrementos iniciais atuam como um protótipo para ajudar a descobrir requisitos para os incrementos posteriores*
  - *Menor risco de falha do projeto como um todo*
  - *Os serviços de mais alta prioridade do sistema tendem a receber a maior parte dos testes*

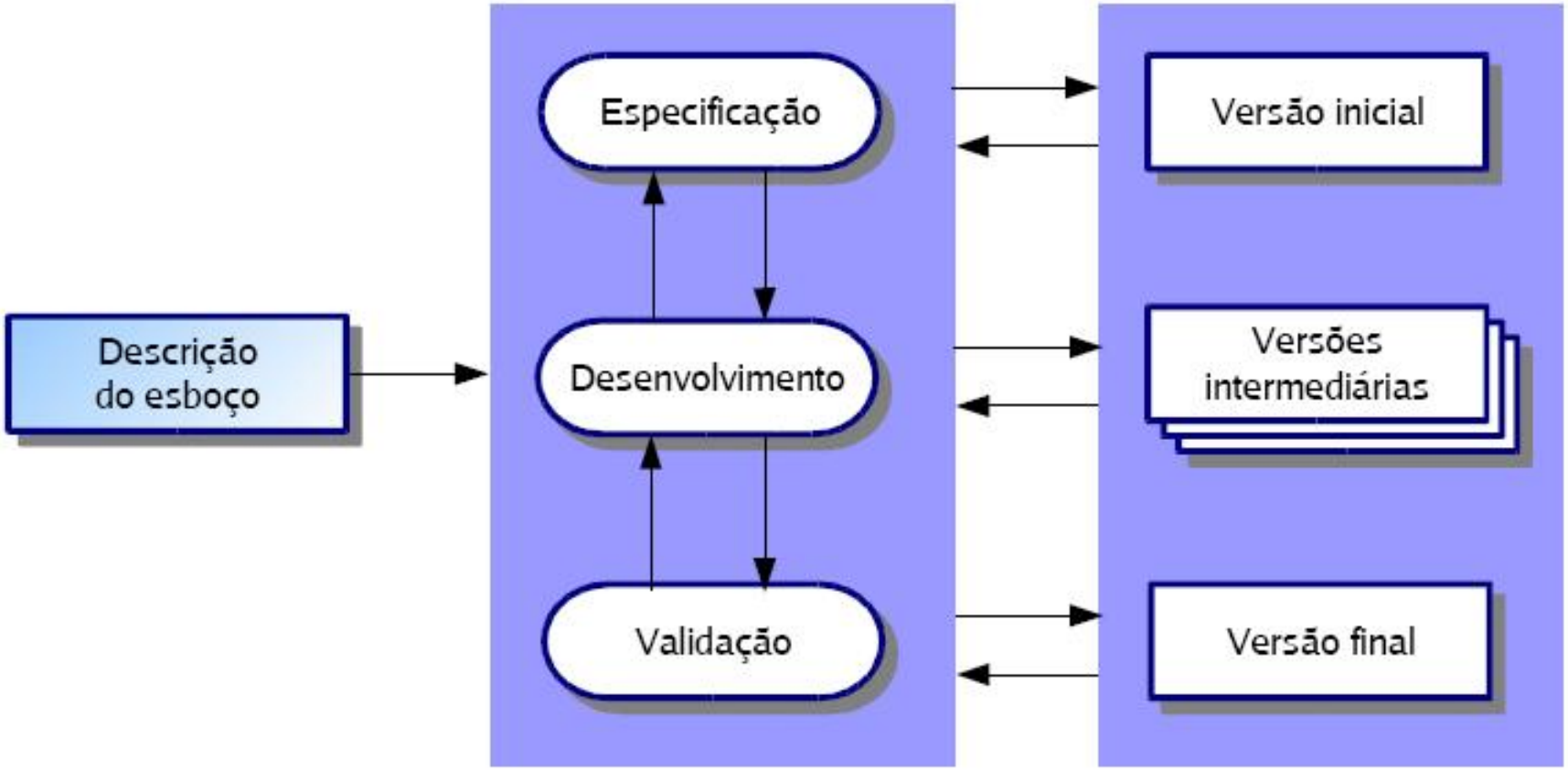
Desenvolvimento Evolucionário

- *Base:*
  - Desenvolver uma implementação inicial
  - Expor resultado ao comentário do usuário
  - Aprimoramento por meio de muitas versões

- *Existem 2 tipos de desenvolvimento evolucionário:*
  - Desenvolvimento exploratório
    - O objetivo é trabalhar com os clientes e evoluir um sistema final a partir de uma especificação genérica inicial. O desenvolvimento se inicia com as partes do sistema que estão compreendidas.

  - Fazer protótipos descartáveis
    - O objetivo é compreender os requisitos do sistema. O protótipo se concentra em fazer experimentos com partes dos requisitos que estejam mal compreendidas.

Desenvolvimento Evolucionário



Desenvolvimento Evolucionário

- Desvantagens
  - *Falta de visibilidade do processo*
  - *Os sistemas frequentemente possuem pouca estrutura*
  - *Podem ser exigidas habilidades especiais (por exemplo, em linguagens para desenvolvimento rápido)*
- Aplicabilidade
  - *Para sistemas interativos pequenos ou de médio porte*
  - *Para partes de sistemas grandes (por exemplo, a interface com o usuário)*
  - *Para sistemas de vida curta*

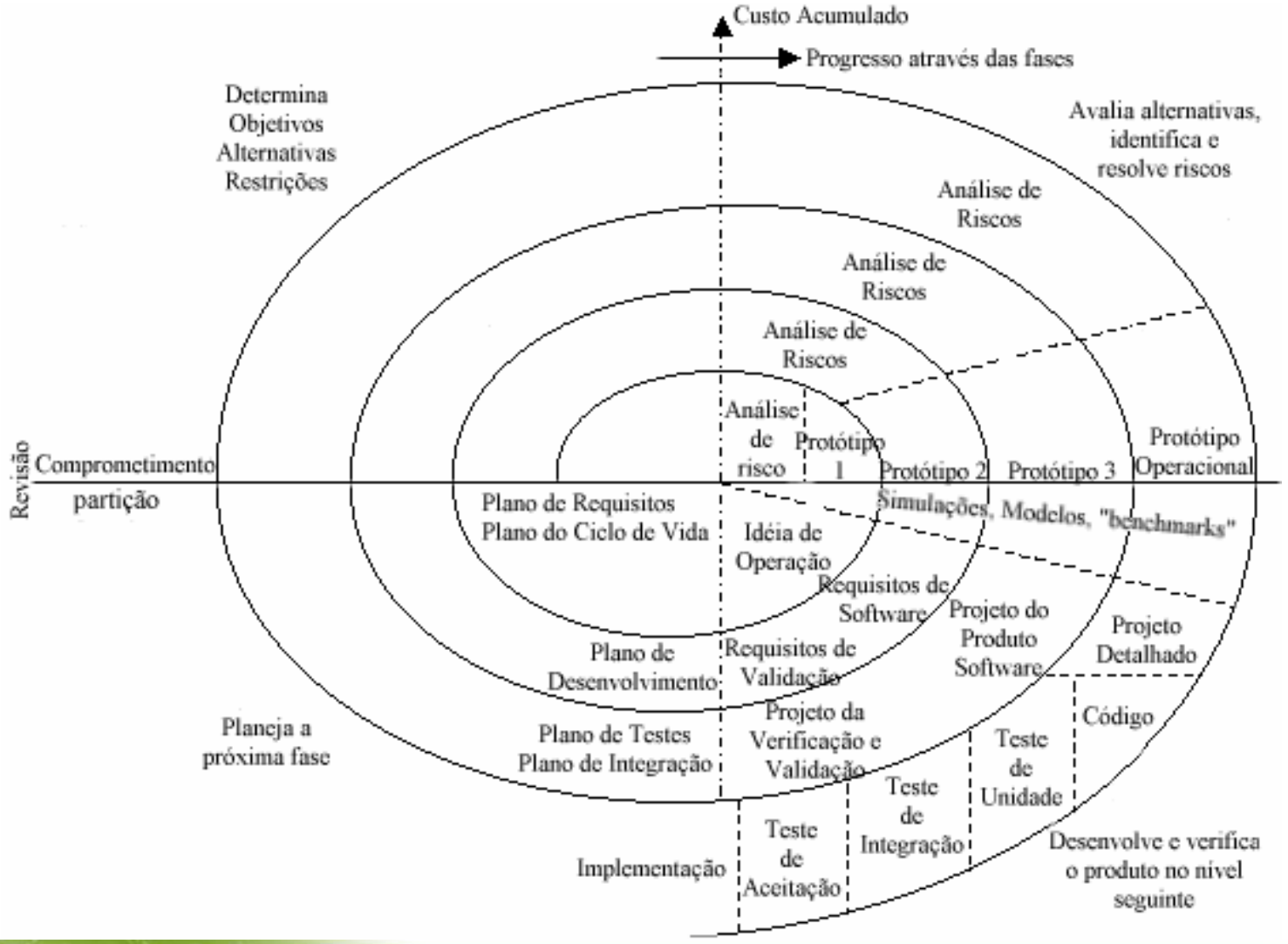
Modelo Espiral

- O processo é representado como uma espiral, em vez de uma seqüência de atividades com caminhos de retorno
- Cada volta na espiral representa uma fase no processo
- Não há fases fixas, tais como especificação ou projeto
  - *As voltas na espiral são escolhidas dependendo do que for exigido*

- *Vantagens*
  - os riscos são explicitamente avaliados e resolvidos durante todo o processo
  - mantém o enfoque sistemático do ciclo clássico

- *Desvantagem*
  - requer boa capacidade para Análise de Riscos

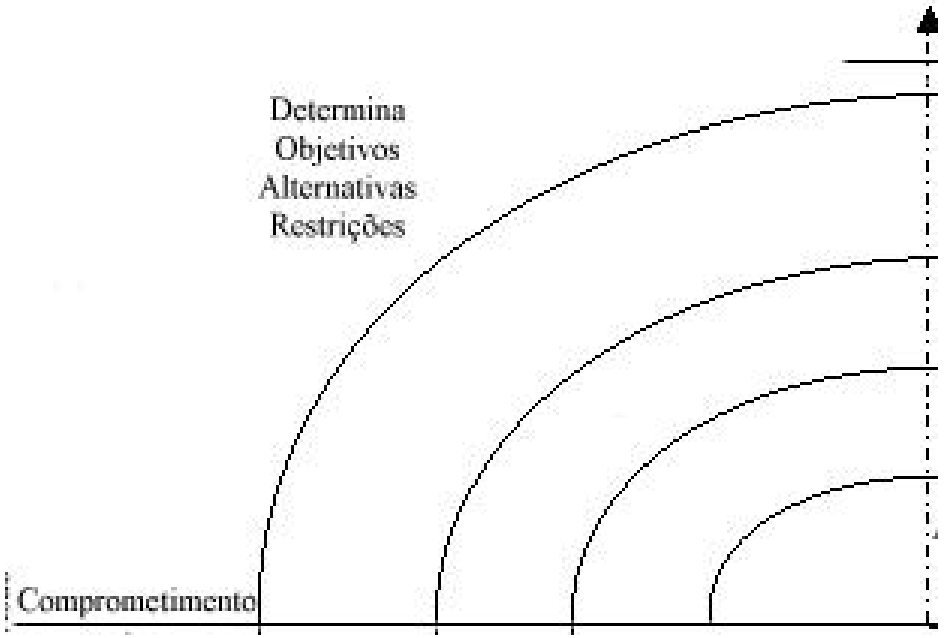
Modelo Espiral



Modelo Espiral – 1º Quadrante

- *Um ciclo se inicia com a tarefa "Determinação de objetivos, alternativas e restrições"*

- *Objetivos principais*
  - comprometimento dos envolvidos
  - estabelecimento de uma estratégia para alcançar os objetivos da fase que se inicia



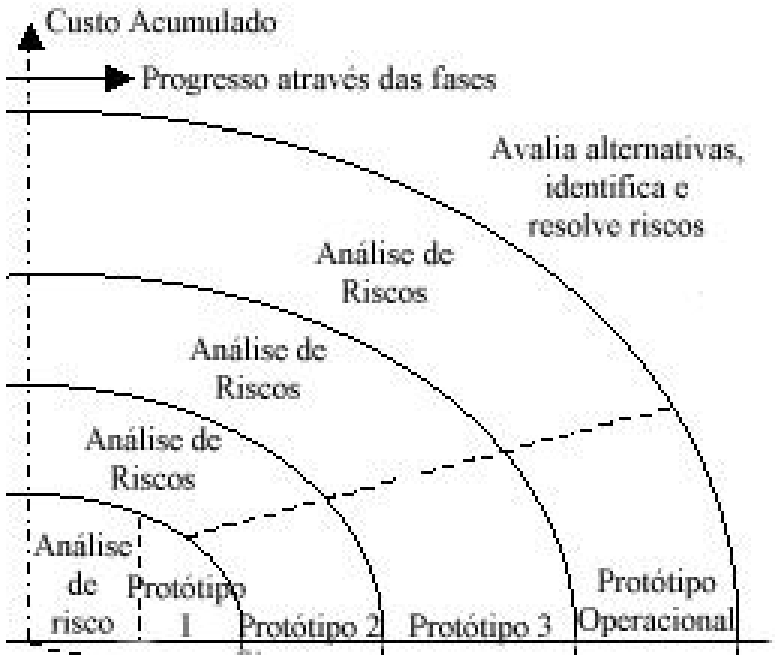
Modelo Espiral – 2º Quadrante

- *Na segunda tarefa, "Avaliação de alternativas, identificação e solução de riscos", executa-se uma análise de risco.*
- *Protótipos são uma forma de avaliar riscos.*

- *Objetivos principais*
  - detectar riscos
  - avaliar soluções que ofereçam menor risco de implementação
  - adotar atividades para reduzir os ricos principais

- *Se o risco for considerado inaceitável, o projeto pode ser encerrado.*

Análise de riscos

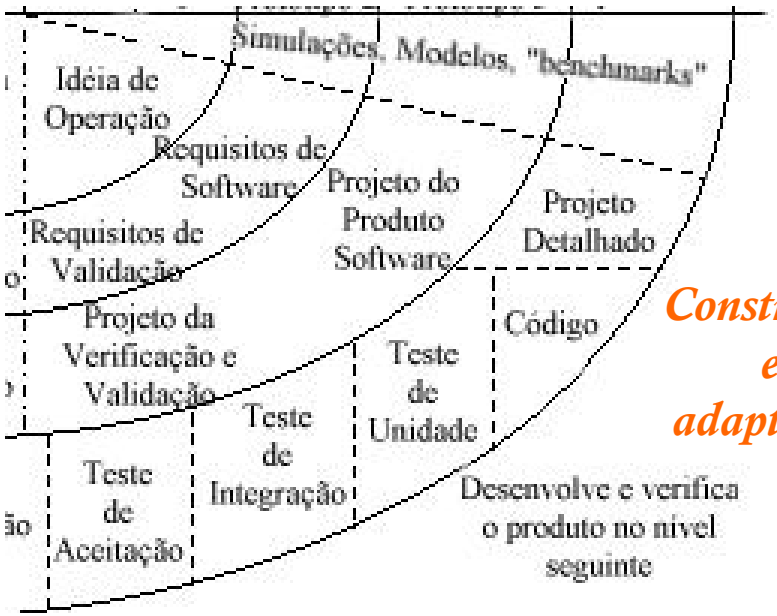


Engenharia

Modelo Espiral – 3º Quadrante

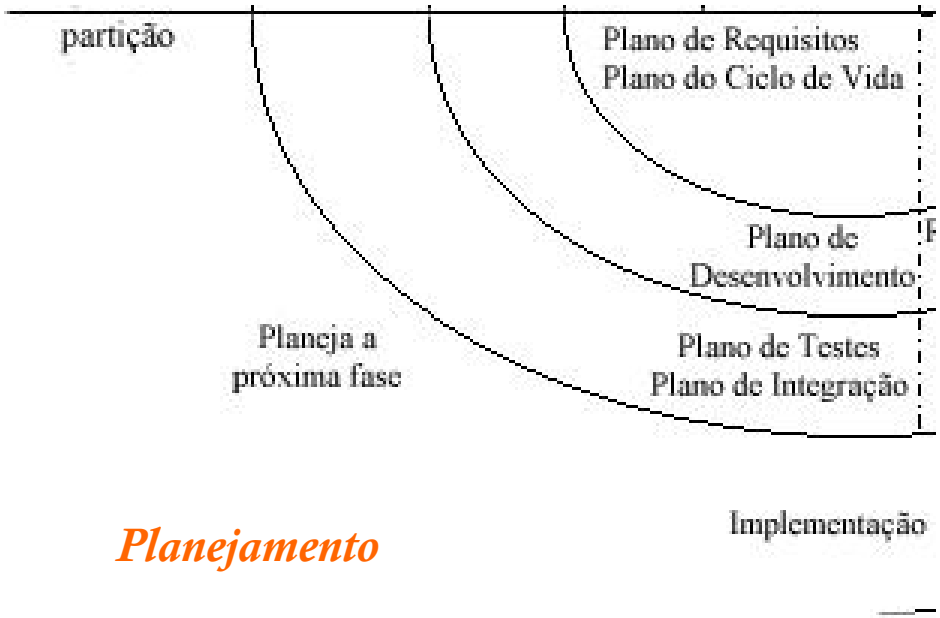
- *Na terceira tarefa ocorre o desenvolvimento do produto.*
- *Deve ser escolhido um modelo de desenvolvimento de software específico*

- *Objetivos principais*
  - definir e validar os requisitos
  - projetar o software
  - projetar a validação e verificação
  - codificar
  - realizar testes
    - *integração*
    - *unidade*
    - *aceitação*



Modelo Espiral – 4º Quadrante

- *Na quarta tarefa o produto é avaliado e se prepara para iniciar um novo ciclo*
- *O projeto é revisado e a próxima fase da espiral é planejada*
- *Objetivos principais*
  - planejar requisitos
  - planejar ciclo de vida
  - planejar desenvolvimento
  - planejar integração e testes



Modelo Espiral

Resumindo...

- Definição do objetivo
  - *Identificam-se os objetivos específicos da fase*
- Avaliação e redução de risco
  - *Os riscos são avaliados e são adotadas as atividades para reduzir os ricos principais*
- Desenvolvimento e avaliação
  - *É escolhido um modelo de desenvolvimento para o sistema, que pode ser qualquer um dos modelos genéricos*
- Planejamento
  - *O projeto é revisado e a próxima fase da espiral é planejada*

Considerações Finais

- *Definir o ciclo de vida adequado às características do projeto é essencial para o seu sucesso*
- *Deve-se analisar os pontos fortes e fracos de cada modelo de ciclo de vida e escolher o que ofereça melhores condições para o desenvolvimento do software*
- *Pontos importantes*
  - variação da especificação dos requisitos ao longo do projeto
  - complexidade do sistema a ser desenvolvido
  - características específicas do projeto

Modelos de Ciclo de Vida

Modelos de Processo

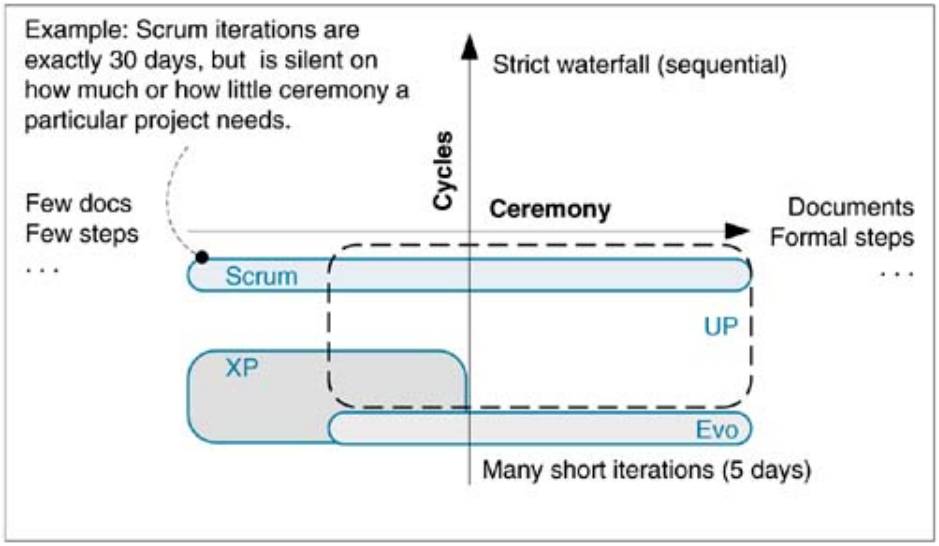
Agenda

- *Introdução*
- *XP*
- *SCRUM*
- *Comparativo entre XP e SCRUM*
- *Rational Unified Process (RUP)*
- *Comparativo entre RUP e Ágeis*

Introdução

- *Modelos de processo são utilizados para auxiliar na definição da sequência de atividades a serem realizadas durante o desenvolvimento de software*
- *Constituem formas específicas de organizar o processo de software para obter vantagens de qualidade e produtividade*

- *Modelos de processo podem ser classificados*
  - de acordo com o grau de cerimônia
    - *peso do método em termos de documentação, etapas formais, revisão, etc*
  - de acordo com o número de ciclos
    - *número e tamanho das iterações*



Ágeis *versus* Rigorosos

- *Métodos rigorosos requerem um alto grau de cerimônia*
- *Métodos ágeis requerem o grau de cerimônia* ***estritamente*** *necessário para a realização da atividade*
  - grau pode ser baixo ou alto
- *Métodos ágeis tendem a ter iterações menores do que métodos rigorosos*
- *Alguns exemplos*
  - Ágeis
    - *XP*
    - *Scrum*
    - *Crystal*
  - Rigorosos
    - *RUP*

Manifesto Ágil

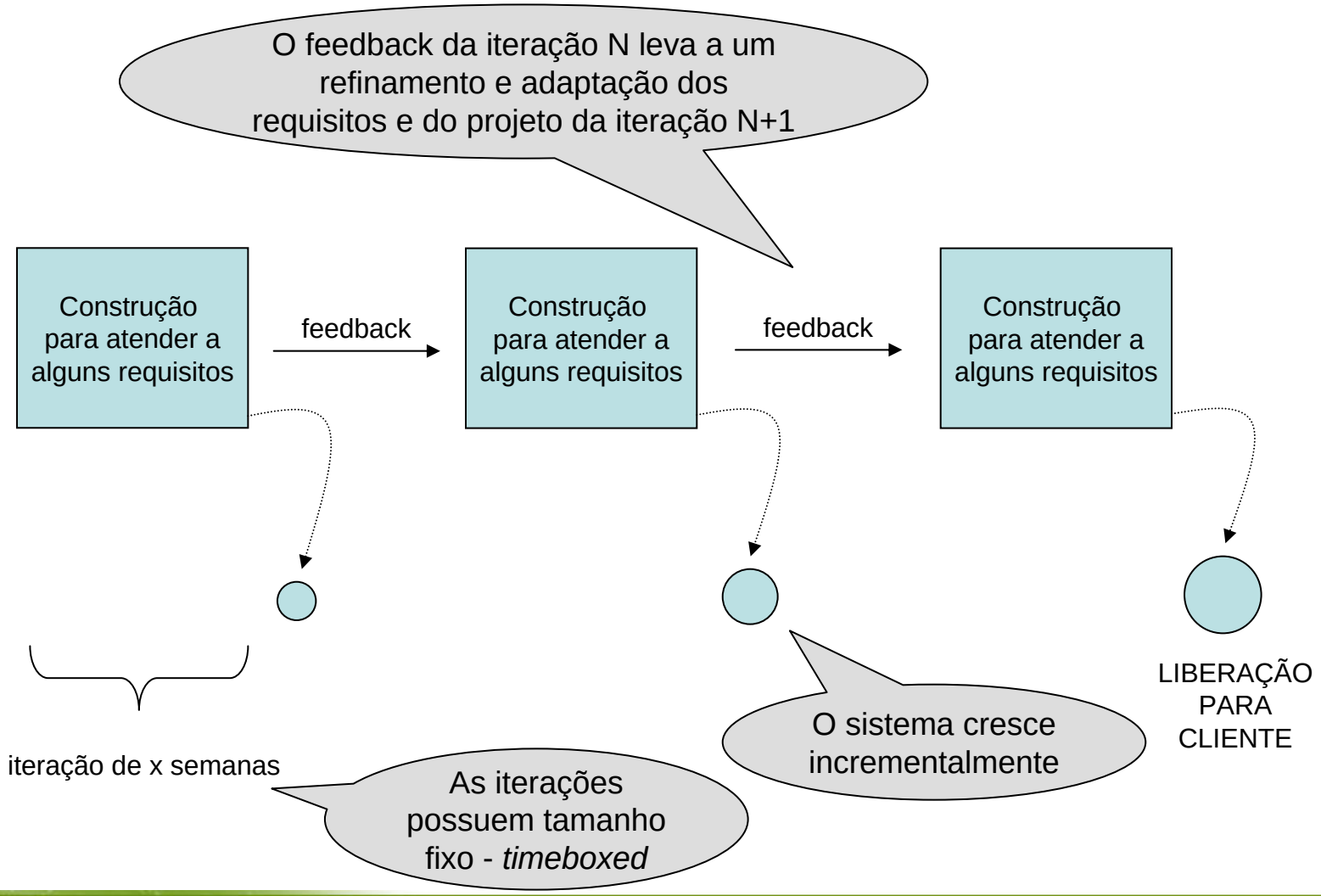
Indivíduos e interações	***sobre***	*Processos e ferramentas*
Software funcionando	***sobre***	*Documentação abrangente*
Colaboração com o cliente	***sobre***	*Negociação de contrato*
Responder a mudanças	***sobre***	*Seguir um plano*

Métodos Ágeis

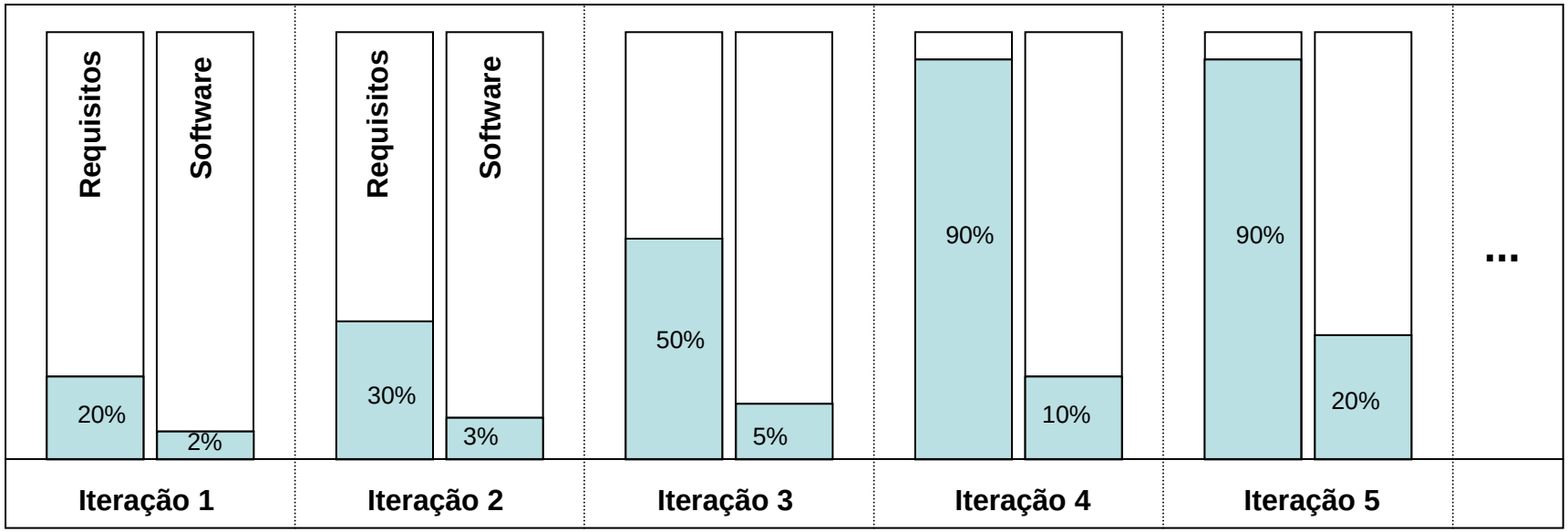
- *Quando usar?*
  - construção de software complexo
  - desenvolvimento de novo produto com grandes índices de mudanças
  - desenvolvimento imprevisível
  - necessidade ou desejo de vantagens competitivas

- *Práticas comuns*
  - desenvolvimento iterativo
  - orientado a riscos e a clientes
  - timeboxing
  - desenvolvimento evolucionário e adaptativo
  - requisitos evolucionários
  - planejamento evolucionário e adaptativo

Desenvolvimento Iterativo e Incremental



Requisitos Evolucionários



Extreme Programming

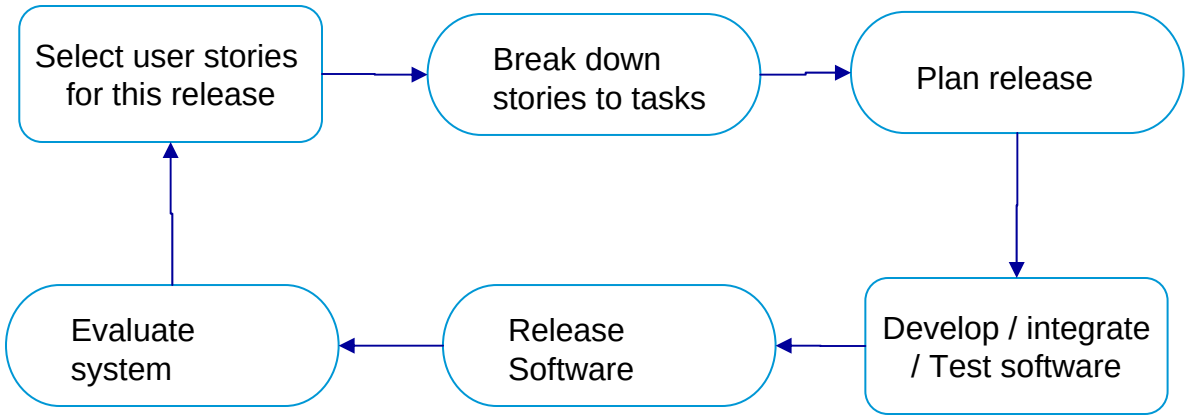
- *Foco*
  - colaboração
  - criação de software rápida e antecipadamente
  - práticas de desenvolvimento voltadas para habilidades
  - buscar o máximo de valor a cada dia de trabalho da equipe para o cliente

- *Baseado em 5 valores*
  - comunicação
    - *cliente sempre disponível*
  - simplicidade
    - *desenvolvedores devem implementar da forma mais simples possível o que o cliente deseja*
  - feedback
    - *do desenvolvedor: aponta riscos, estimativas e alternativas de design*
    - *do cliente: conduz o desenvolvimento do seu produto, estabelece prioridades e informa aquilo que é realmente importante*

Extreme Programming

- *Baseado em 5 valores (continuação)*
  - coragem
    - *para mudar de acordo com novas solicitações*
    - *para enfrentar riscos associados a mudanças*
  - respeito
    - *valor fundamental para que os demais valores sejam atingidos*
    - *saber ouvir, saber compreender e respeitar o ponto de vista do outro é essencial para que um projeto de software seja bem sucedido*

Extreme Programming

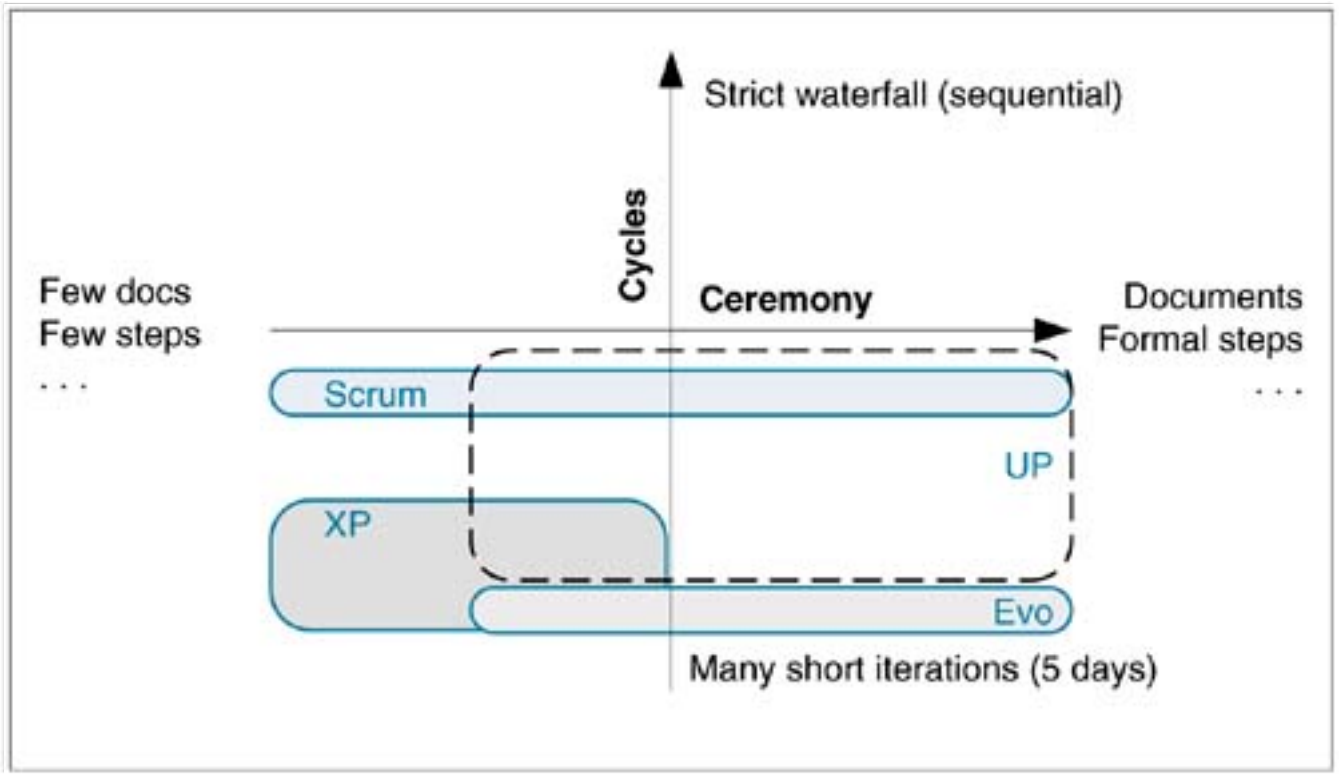


Extreme Programming

- *Práticas*
  - Histórias
    - *requisitos dos clientes são escritos em pequenos cartões*
    - *histórias servem como guias para a equipe*
  - Ciclo Semanal
    - *uma vez por semana os desenvolvedores se reúnem com o cliente*
    - *cliente solicita as funcionalidades desejadas através de estórias*
  - Equipe Integral
    - *equipe deve ser formada por desenvolvedores, cliente e por qualquer pessoa que possa contribuir para o projeto*
  - Programação em pares
    - *todo e qualquer código implementado no projeto deve ser efetuado em dupla*
  - Integração contínua
    - *se possível deve ser efetuada diversas vezes ao dia*
    - *a equipe deve ter conhecimento do código recém desenvolvido*
  - Reuniões em pé (Stand-up meetings)

Extreme Programming

- *Classificação*
  - ciclos pequenos – para projetos médios, cerca de 1 a 3 semanas
  - pouco grau de cerimônia – o menor possível



Extreme Programming

- *Papéis*
  - Analistas de Teste
  - Arquitetos
  - Designers de Interação
  - Executivos
  - Gerentes de Projeto
  - Gerentes de Produto
  - Programadores
  - Recursos Humanos
  - Redatores Técnicos
  - Usuários

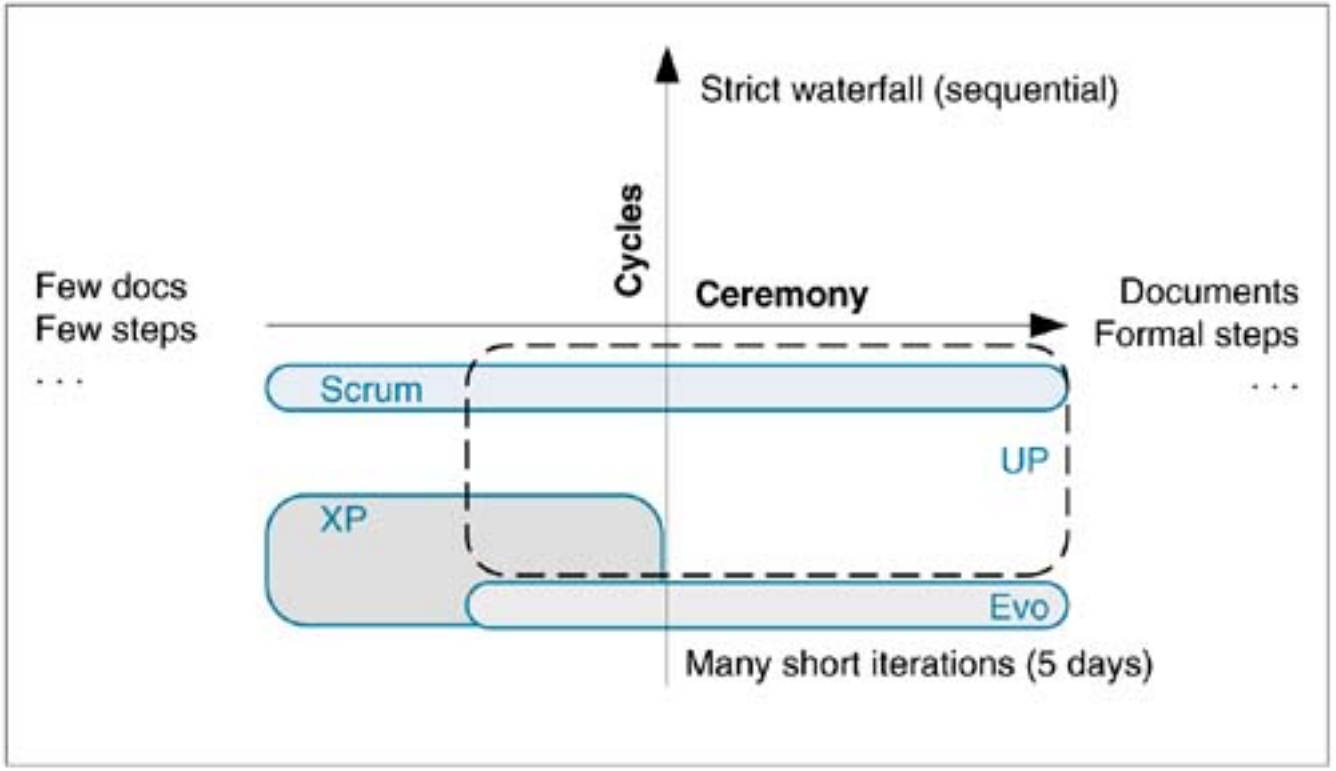
Scrum

- *Valores*
  - compromisso
  - foco
  - abertura
  - respeito
  - coragem

- *Pontos chave*
  - equipes que se auto organizam e gerenciam
  - iterações de cerca de 30 dias – chamadas sprints
  - reuniões diárias com a equipe (stand-up meetings ou Scrum meetings) abordando questões especiais
  - entregas para o cliente ao final de cada iteração
  - para cada iteração, planejamento adaptativo orientado ao cliente

Scrum

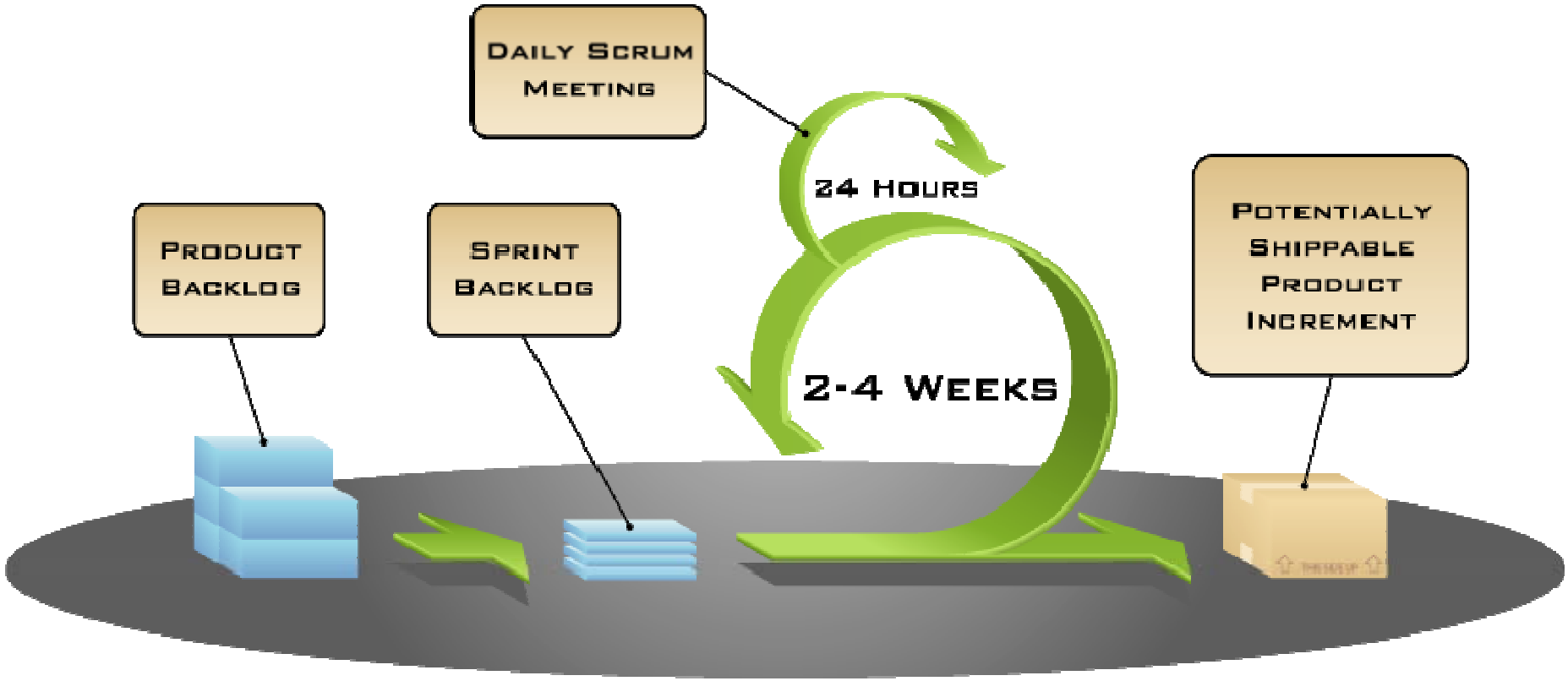
- *Classificação*
  - rígido em termos de tamanho das iterações – sempre iterações curtas
  - flexível em termos de grau de cerimônia – depende da complexidade do projeto



Scrum

- *Enfatiza um conjunto de valores e práticas de gerenciamento de projeto, ao invés de requisitos, implementação, etc*
- *Ênfase em processos empíricos ao invés de processos definidos*
- *O produto evolui em uma série de iterações mensais (sprints)*
- *O produto é projetado, codificado e testado durante o sprint*
- *Os requisitos são listados em um 'Product Backlog'*
  - a cada sprint a prioridade da fila de requisitos pode ser modificada pelo cliente, sem custo adicional para o projeto
  - antes do início de cada sprint é feito um planejamento rápido das atividades necessárias para atingir o objetivo do sprint
    - *Sprint Backlog*
- *Durante um sprint, alguns parâmetros são mantidos fixos*
  - esforço
  - prazo
  - objetivo do sprint

Scrum


Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
Copyright © 2005, Mountain Goat Software

Scrum

- *Papéis*
  - Scrum Master
    - *gerencia e codifica*
    - *conhece e reforça a visão e os objetivos do projeto e da iteração*
    - *assegura o seguimento das práticas e valores do Scrum*
    - *mediador entre a gerência e a equipe*
    - *remove impedimentos ao progresso do projeto*
    - *conduz as reuniões diárias*
    - *conduz as revisões ao final de cada sprint*
  - Dono do Produto
    - *1 pessoa responsável pela criação e priorização do Product Backlog*
    - *escolhe os objetivos para o próximo sprint*
    - *junto com outros stakeholders, revisa o sistema no final de cada sprint*
  - Equipe
    - *trabalha no Sprint Backlog*
    - *não há hierarquia*

Scrum

- *Vantagens*
  - práticas simples e produtos de trabalho gerenciáveis
  - resolução de problemas e auto-gerência realizadas pela equipe e individualmente
  - requisitos e desenvolvimento evolucionários e incrementais, com comportamento adaptativo
  - participação do cliente, que também guia a execução das atividades
  - foco
  - abertura e visibilidade
  - facilmente combinado com outros métodos
  - comunicação, aprendizagem e agregação de valor em equipe
  - construção da equipe através das reuniões diárias

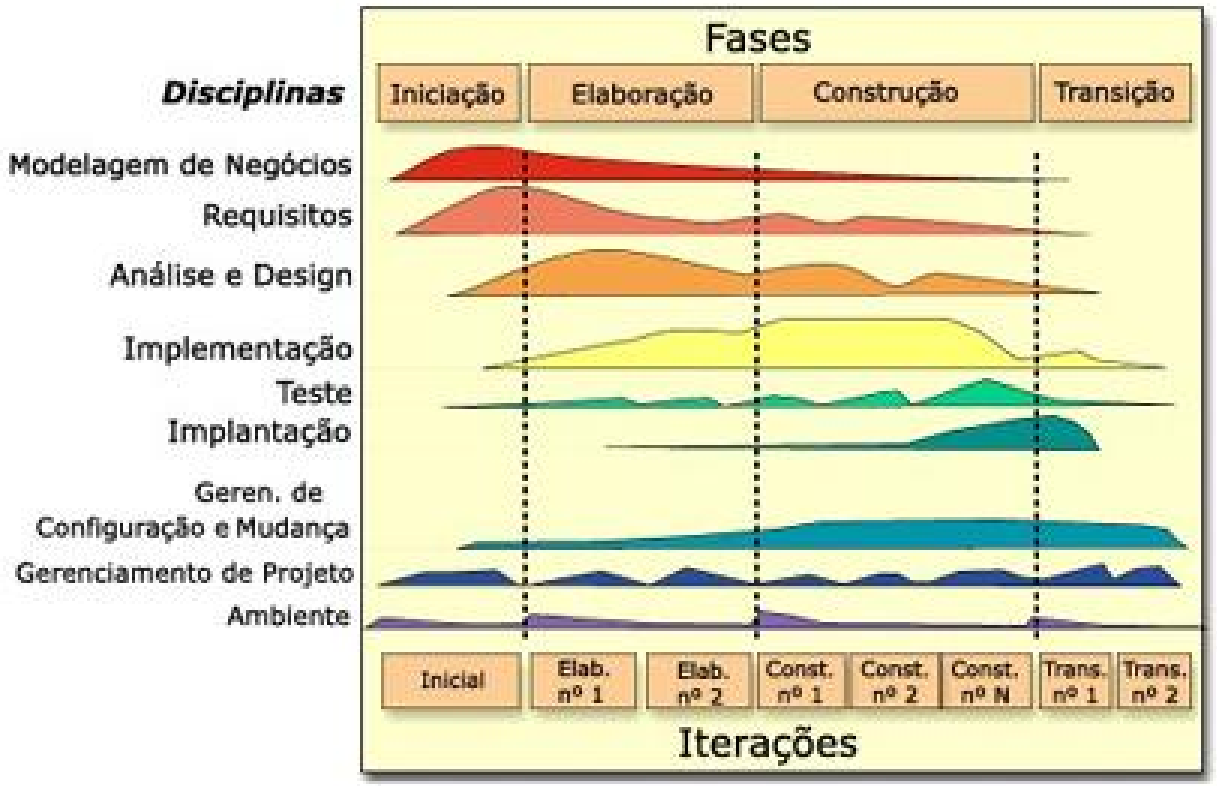
- *Desvantagens*
  - oferece apoio mínimo sobre as demais disciplinas, com exceção de gerenciamento de projetos (ex. requisitos, programação, técnicas de engenharia de software, etc)
  - não define padrões para documentação do projeto

Rational Unified Process

- *Conforme [Kroll e Kruchten 2003], podemos definir o RUP como uma maneira de desenvolvimento de software que é iterativa, centrada à arquitetura e guiada por casos de uso.*
- *É classificado como um processo de Engenharia de Software organizado em disciplinas e fases.*
- *O RUP foi criado baseando-se nas melhores práticas de engenharia de software.*
- *Contém todos os elementos básicos de um processo de desenvolvimento (papéis, tarefas, atividades, artefatos, fluxo de trabalho).*
- *Possui uma vasta biblioteca conceitual sobre elementos relacionados à engenharia de software.*
- *Atualmente na versão 7.0*
- *Para conhecer com mais detalhes o RUP , acesse: http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf*

Rational Unified Process

- *Fases*



Rational Unified Process

- *Princípios*
  - Adaptar o Processo para diferentes necessidades
  - Alinhar prioridades dos Stakeholders com objetivos de negócio
  - Otimizar a comunicação entre as equipes
  - Trabalhar em conjunto com equipes
  - Demonstrar o Valor Iterativamente
  - Elevar o Nível de Abstração
  - Focar Continuamente na Qualidade

Comparação

- *Pontos fortes – Abordagem "Rigorosa"*
  - Considera características mais abrangentes da gerência
    - *Riscos*
    - *Gerenciamento de Contratos com Fornecedores*
  - Planejamento de recursos necessários ao projeto
    - *Infraestrutura*
    - *Recursos Humanos*
  - Gerencia os compromissos com elementos externos ao projeto que não são clientes, mas que afetam o seu resultado
  - Provê uma visão abrangente do planejamento do projeto

Comparação

- *Pontos fortes – Abordagem Ágil*
  - Permite balancear a flexibilidade e a estabilidade
  - A priorização e o planejamento são focados na satisfação do cliente
  - Facilidade de tratar as mudanças no escopo e nas prioridades do projeto
  - Respostas mais imediatas aos desvios devido à micro-gerencia
  - Valoriza a colaboração, motivação e compartilhamento do conhecimento.

Rigorosos X Ágeis

Fatores que afetam a Gerência de Projetos - Comparação	
Processos "Rigorosos"	***Métodos Ágeis***
Previsibilidade	*Adaptabilidade*
Conhecimento Explícito (documentação)	*Conhecimento Tácito (comunicação)*
Profissionais Especialistas	*Profissionais Generalistas*
Cliente externo ao projeto	*Cliente na equipe*
Prioridade execução - fatores técnicos (WBS, estimativas, riscos)	*Prioridade execução – maior valor ao Cliente / Negócio*
Entregas dependentes do plano	*Entregas frequentes e periódicas*
Escopo Definido e Fixo	*Escopo Indefinido e Variável*
Gerência de Mudanças	*Mudanças incorporadas naturalmente*
Foco no planejamento global	*Foco no micro planejamento*
Contratos Formais	*Contratos "Colaborativos"*
Progresso – Artefato entregue	*Progresso - Software funcionando*

Bibliografia

- Livros:
  - *Beck, K.;* Extreme Programming Explained*; Second Edition; Addison Wesley; 2005*
  - *Schwaber, K.;* Agile Project Management with SCRUM*; Microsoft Press; 2004*
  - *Chrissis, M.B.; Konrad, M; Shrum, S;* CMMI Guidelines for Process Integration and Product Improvement*; Addison Wesley; 2007*
  - *Boehm, B.;Turner, R.;* Balancing Agility and Discipline; *Addison Wesley; 2004*
  - *Larman, C.;* Agile & Iterative Development; *Addison Wesley; 2004*
  - *Fiorini, S.; Staa, A.v.; Baptista, R.M.;* Engenharia de Software com CMM*; Brasport; 1998*
  - *Jacobson, I.; Booch, G.; Rumbaugh, J.;* The Unified Software Development Process*; Addison Wesley; 1999*
  - *Sommerville, Ian; Software Engineering 2004*

Modelos de Processo