Modelos de Ciclo de Vida e Processos de Software

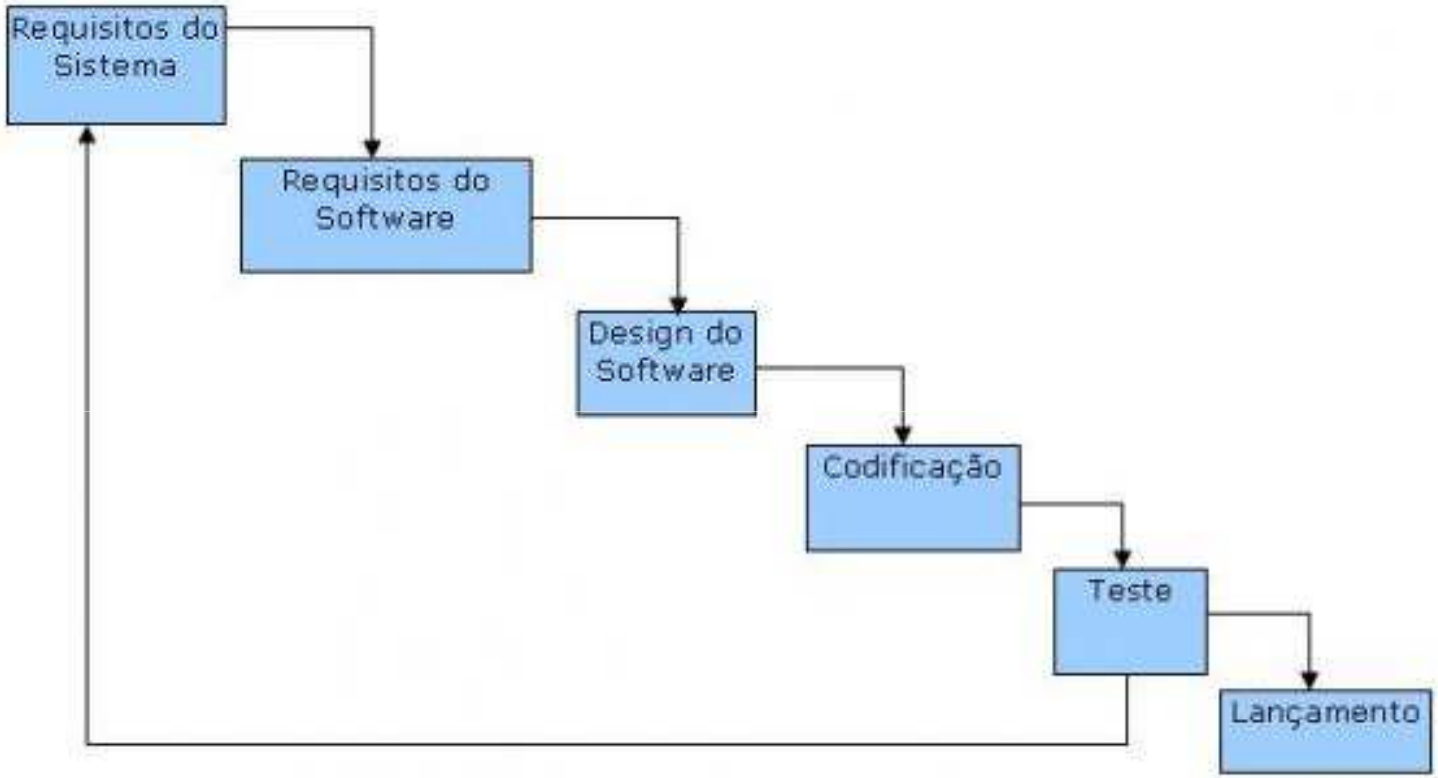
Cascata ou Waterfall

É um dos modelos de ciclo de vida mais simples e mais conhecidos das organizações de desenvolvimento de software.

Nele, as fases do projeto são executadas em uma **seqüência linear** e uma próxima fase só tem início quando a fase anterior está completamente pronta.

Ao final de cada fase, é feita uma revisão para avaliar se realmente pode-se avançar para a próxima.

Caso a revisão aponte que o projeto não está apto a entrar na fase seguinte, ele permanece na fase corrente até que ele seja aprovado.


Requisitos do Sistema
Requisitos do Software
Design do Software
Codificação
Teste
Lançamento


Viabilidade
Análise
Projeto
Implementação
Testes
Implantação
Operação e
Manutenção

Vantagens

- Fácil de gerenciar: Etapas bem definidas e sem sobreposição;
- Eficiente em casos nos quais o domínio de aplicação é bem entendido;
- Eficiente no desenvolvimento de projetos em quais vários sistemas similares foram construí-dos anteriormente.

Desvantagens

- Em função da dificuldade de se obter todos os requisitos do sistema no início do projeto, geralmente esse processo resulta em um atraso para o início da fase de projeto, cumulativa ao prazo final;
- Raramente as fases de execução seguem um fluxo tão seqüencial e sem interações, portanto o planejamento não é de qualidade total;
- Obtenção do produto final apenas no final do projeto, deixando margens de correção menores.

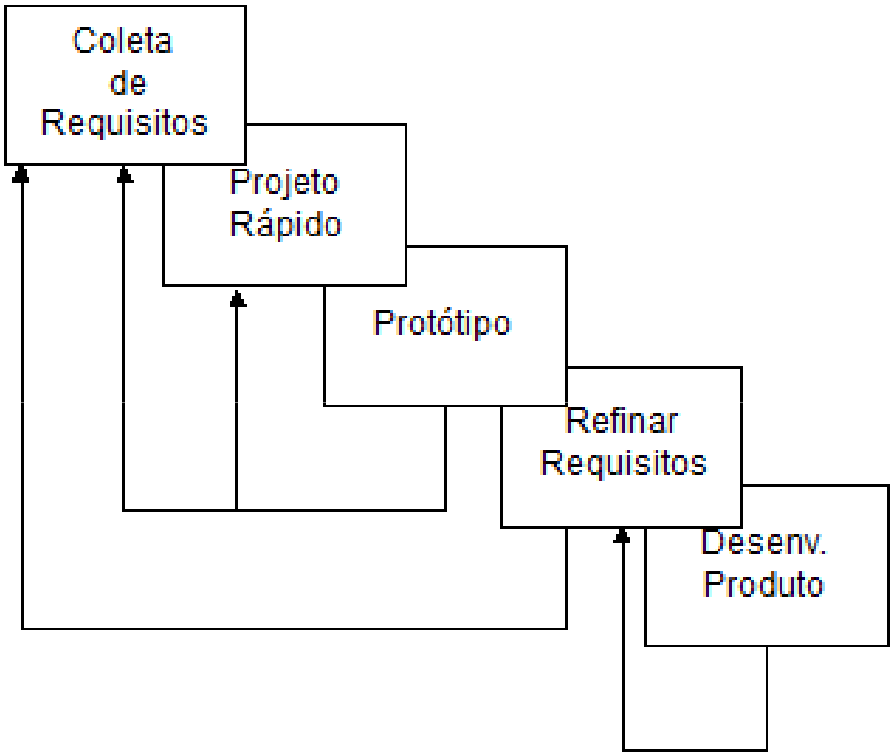
Prototipação

Neste modelo, os desenvolvedores iniciam o projeto e a implementação das partes essenciais do sistema em um **protótipo** e refinam o mesmo, adicionando novas funcionalidades e melhorias até ele se tornar o produto de software final que eventualmente se tornará o produto a ser entregue.

A diferença desse modelo de ciclo de vida com o anterior é que, naquele, fazia-se primeiramente um protótipo apenas para auxiliar na elaboração dos requisitos finais do sistema, o qual depois era descartado.

Na prototipação evolutiva, o próprio protótipo será o produto entregue ao cliente.


início
fim
obtenção dos requisitos
projeto rápido
construção protótipo
avaliação protótipo
refinamento protótipo
construção produto


Coleta
de
Requisitos
Projeto
Rápido
Protótipo
Refinar
Requisitos
Desenv.
Produto

Vantagens

- Apropriado para sistemas nos quais os requisitos mudam rapidamente;
- Provê um feedback constante do cliente a respeito do produto;
- Apropriado para sistemas nos quais o cliente está apressado para que você entregue um conjunto de requisitos;
- Aumenta a visibilidade do produto que está sendo desenvolvido;
- Apropriado para sistemas nos quais nem você nem seu cliente conhecem bem o domínio da aplicação.

Desvantagens

- Dificuldade em se prever o tempo que será necessário para produzir um produto aceitável;

. Projeto pobre: Tendência a se desenvolver no modo *code-and-fix*, de esquecimento das fases de análise, projeto e testes;

. Manutenibilidade pobre em razão do projeto;

. Maior complexidade de gerenciar o projeto, os riscos e a qualidade do produto final.

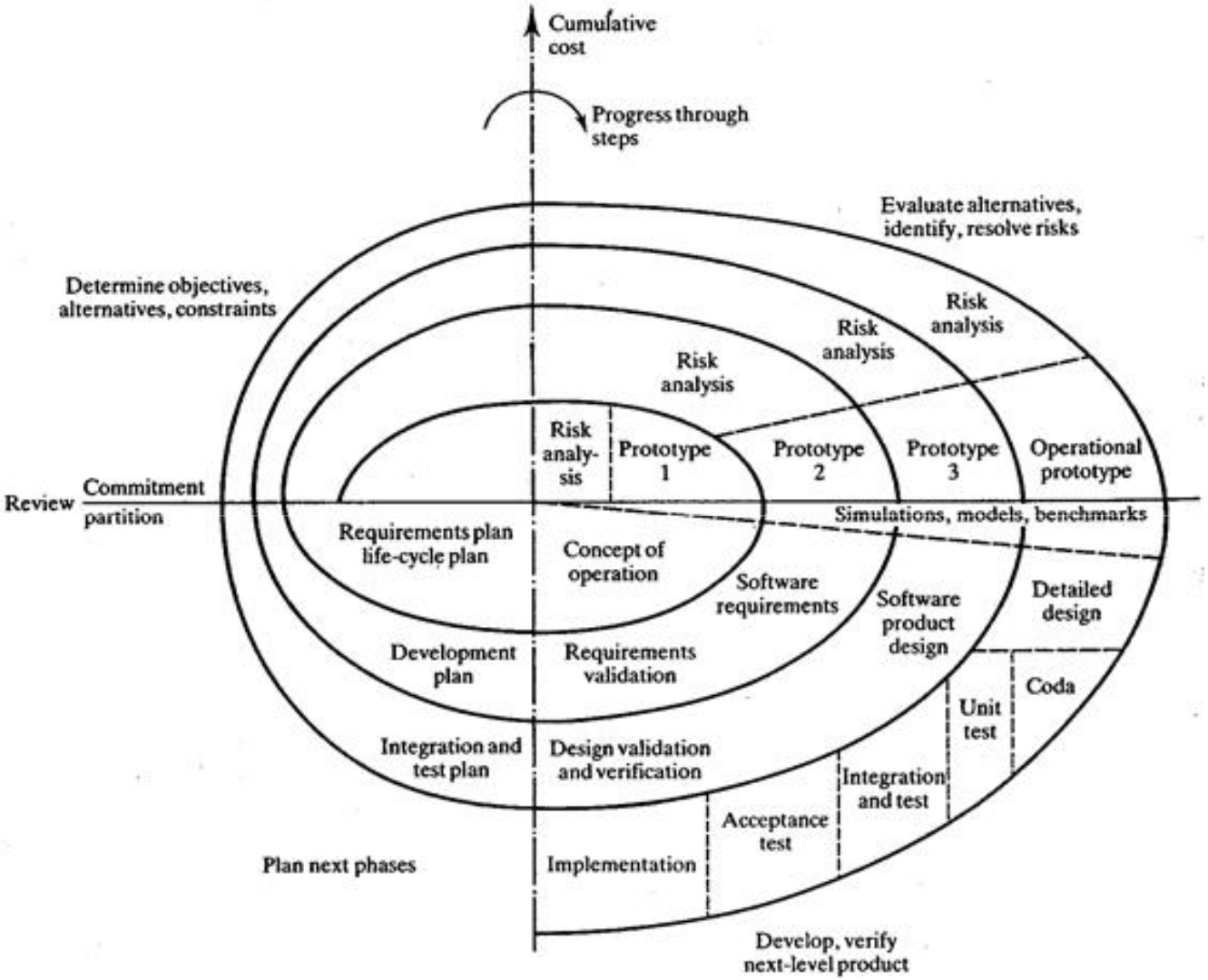
Espiral

- É um modelo de ciclo de vida orientado a riscos, que reparte o projeto em mini-projetos.
- O **conceito de risco** pode referenciar desde uma compreensão pobre dos requisitos ou da arquitetura até problemas de performance ou no conhecimento da tecnologia a ser aplicada, dentre outros.
- Após a resolução dos riscos maiores, o modelo espiral pode ser terminado com um outro modelo de ciclo de vida qualquer.

Passos do Ciclo Espiral

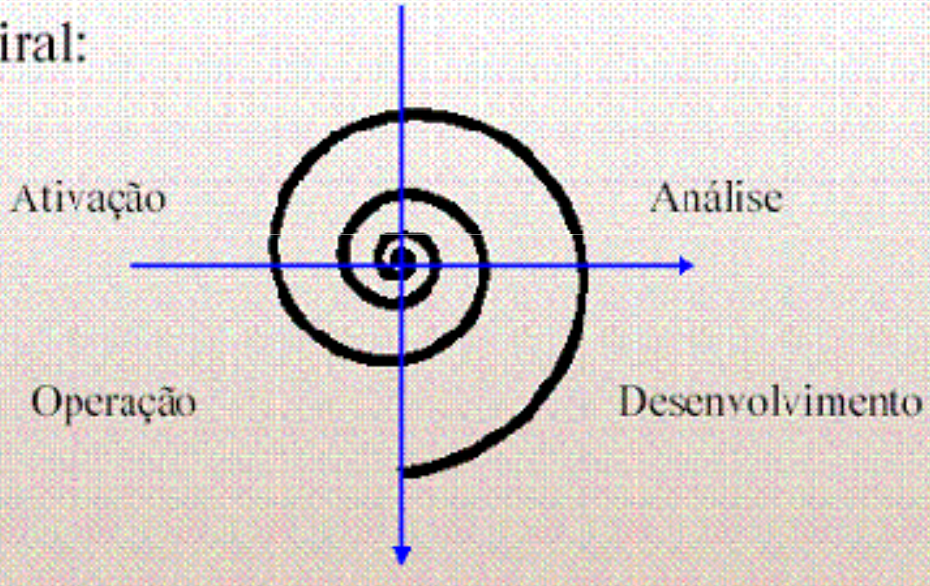
- Cada ciclo da espiral envolve alguns passos para que o mesmo seja concluído, dentre os quais:

  . Determinar objetivos, alternativas e incertezas;
  . Identificar e resolver riscos;
  . Avaliar as alternativas;
  . Desenvolver os produtos de entrega para aquela interação e verificar seu grau de correção;
  . Planejar a próxima interação.


Cumulative cost
Progress through steps
Evaluate alternatives, identify, resolve risks
Determine objectives, alternatives, constraints
Risk analysis
Risk analysis
Risk analysis
Risk analysis
Prototype 1
Prototype 2
Prototype 3
Operational prototype
Review
Commitment partition
Simulations, models, benchmarks
Requirements plan life-cycle plan
Concept of operation
Software requirements
Software product design
Detailed design
Development plan
Requirements validation
Coda
Unit test
Integration and test plan
Design validation and verification
Integration and test
Acceptance test
Plan next phases
Implementation
Develop, verify next-level product

Processos de software

- A espiral:



Vantagens

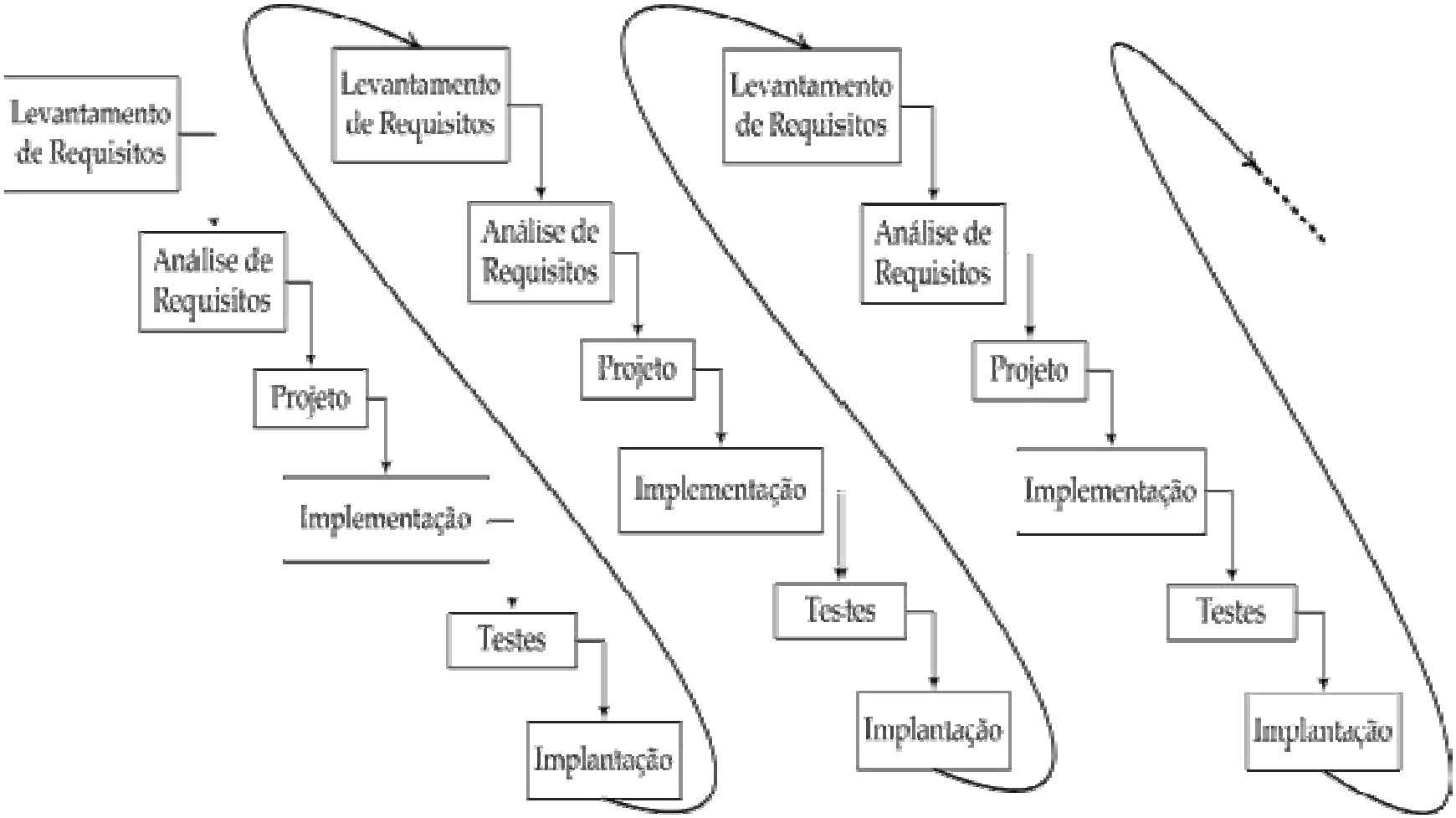
- Possibilidade de combinar o modelo espiral com outros modelos de ciclo de vida;
- Ajuda a aumentar a qualidade pelo planejamento e análise dos riscos em cada fase;
- Maior visibilidade para a gerência, sobretudo na **gerência de riscos**.

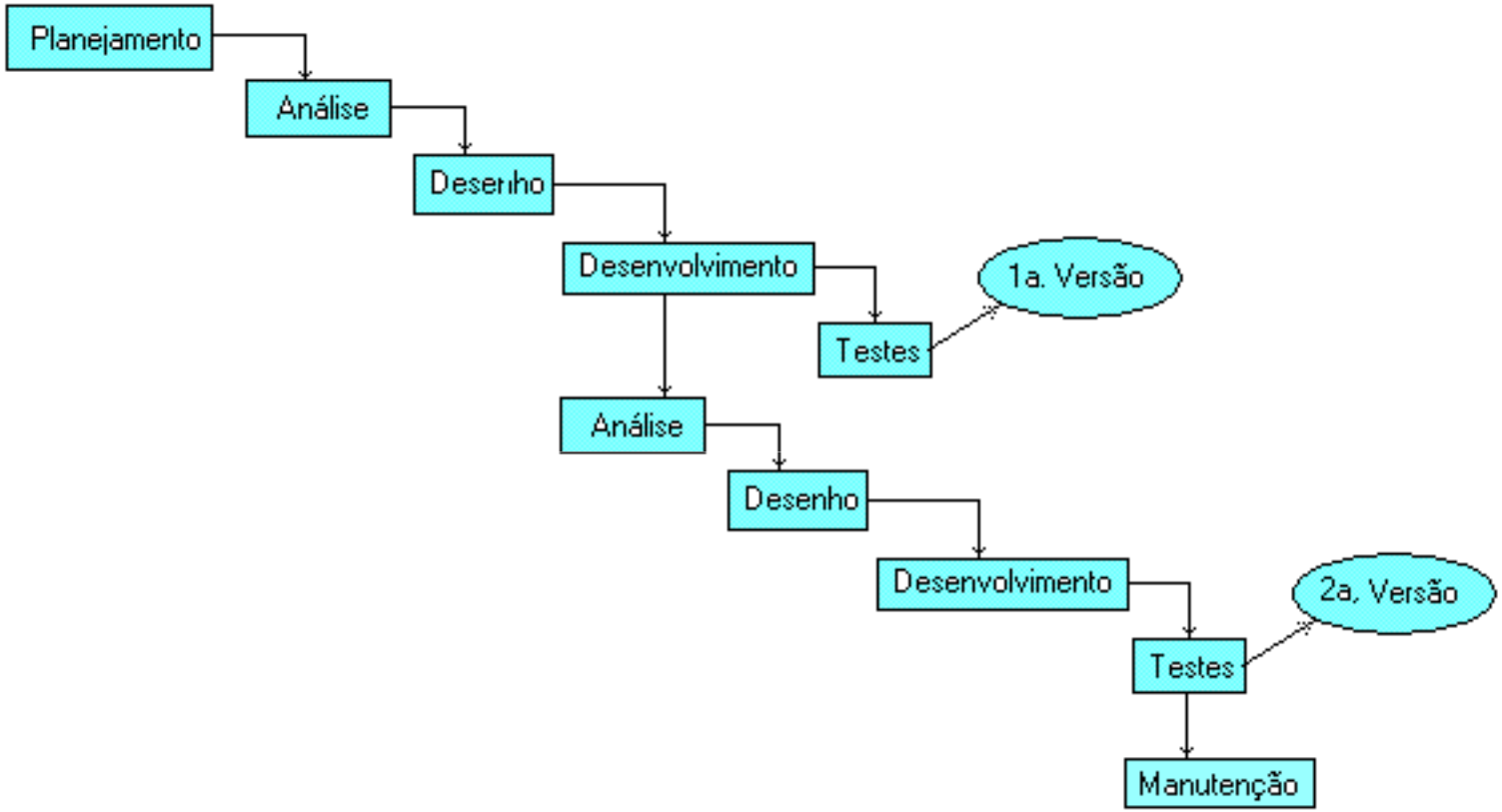
Desvantagens

- Gerência de processo mais complexa;
- Necessidade de maior experiência da equipe de desenvolvimento, sobretudo dos responsáveis pela gerência;
- Maior experiência da equipe e maior esforço para o desenvolvimento podem aumentar consideravelmente os custos.

Modelo Iterativo Incremental

- Tenta combinar os benefícios do modelo cascata e da prototipação. A idéia básica é que um software deveria ser desenvolvido em partes, cada qual adicionando alguma capacidade funcional ao mesmo até que o software completo esteja implementado.
- A cada incremento, podem ser feitas extensões e modificações do projeto. Para o controle dessas alterações, o modelo faz uso de uma lista de controle, que contém, em ordem, todas as tarefas que devem ser executadas até a implementação final do produto.
- Cada passo consiste em remover a próxima tarefa selecionada, codificar e testar a implementação, fazer uma análise do produto obtido após a execução dessa fase e atualizar a lista com o resultado da análise realizada.


Levantamento de Requisitos
Análise de Requisitos
Projeto
Implementação
Testes
Implantação
Levantamento de Requisitos
Análise de Requisitos
Projeto
Implementação
Testes
Implantação
Levantamento de Requisitos
Análise de Requisitos
Projeto
Implementação
Testes
Implantação


Planejamento
Análise
Desenho
Desenvolvimento
Testes
1a. Versão
Análise
Desenho
Desenvolvimento
Testes
2a. Versão
Manutenção

Vantagens

- Disponibilidade de partes prontas do sistema mais cedo;
- Facilidade nos testes: Geralmente, testar cada incremento é mais fácil do que testar o software pronto e tudo de uma vez só;
- Feedback do cliente a cada incremento feito;
- A aprendizagem do desenvolvedor numa linguagem é favorecida: Pode se optar em resolver as partes mais fáceis antes, enquanto ele aprende a linguagem, e deixar as partes mais complexas do sistema para depois.

Desvantagens

- A possibilidade de o sistema ser dividido em partes como pré-requisito, já que nem sempre um sistema pode ser dividido;
- Dificuldade na integração das partes desenvolvidas;
- Negociação com o cliente a respeito do pagamento do produto de software final pode ser problemática, uma vez que o desenvolvimento em passos incrementais costuma induzir o cliente a acrescentar requisitos e funcionalidades que não estavam previstas no escopo inicial do projeto, o que resulta no encarecimento do desenvolvimento do produto final.

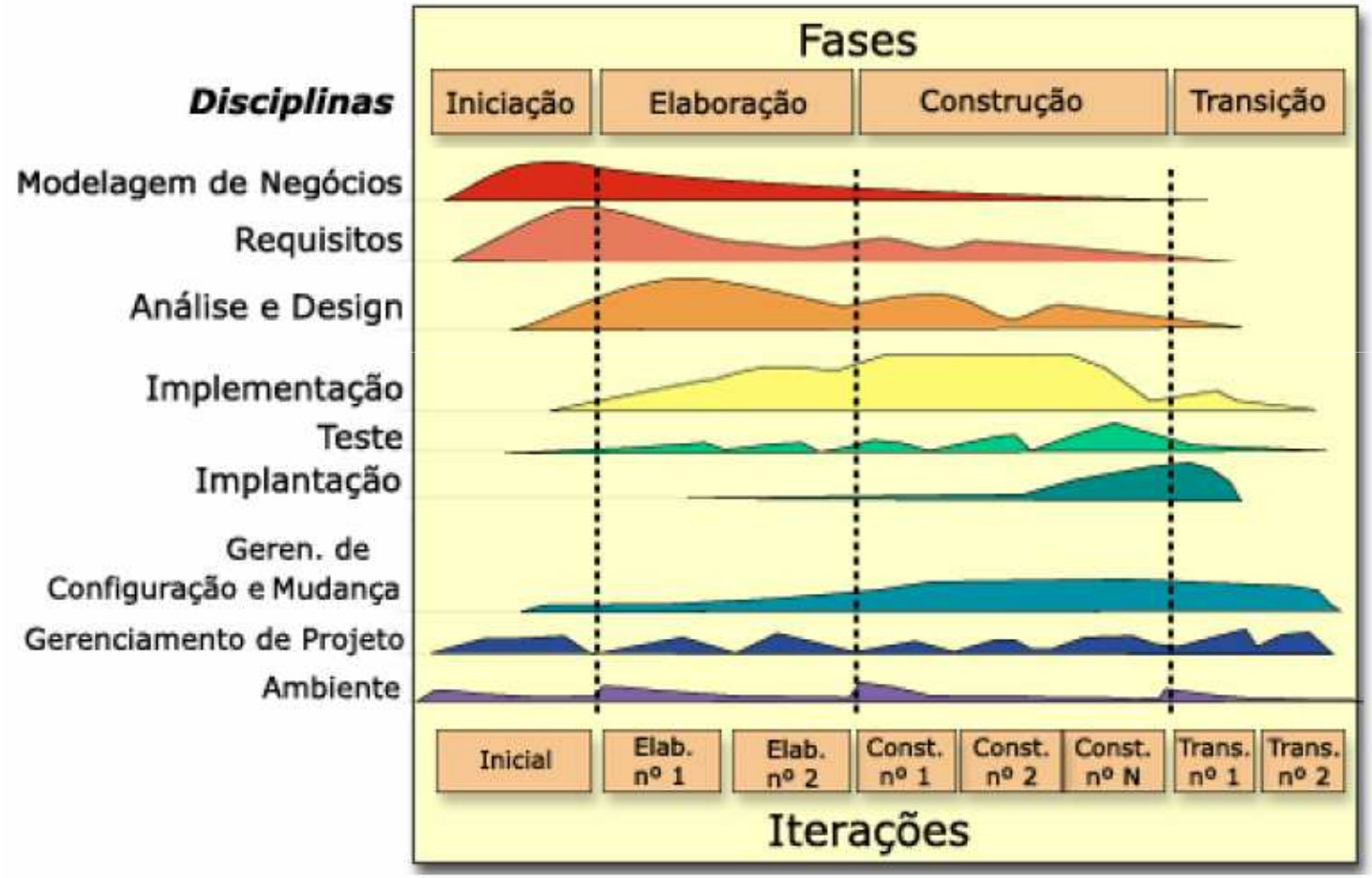
Conclusões

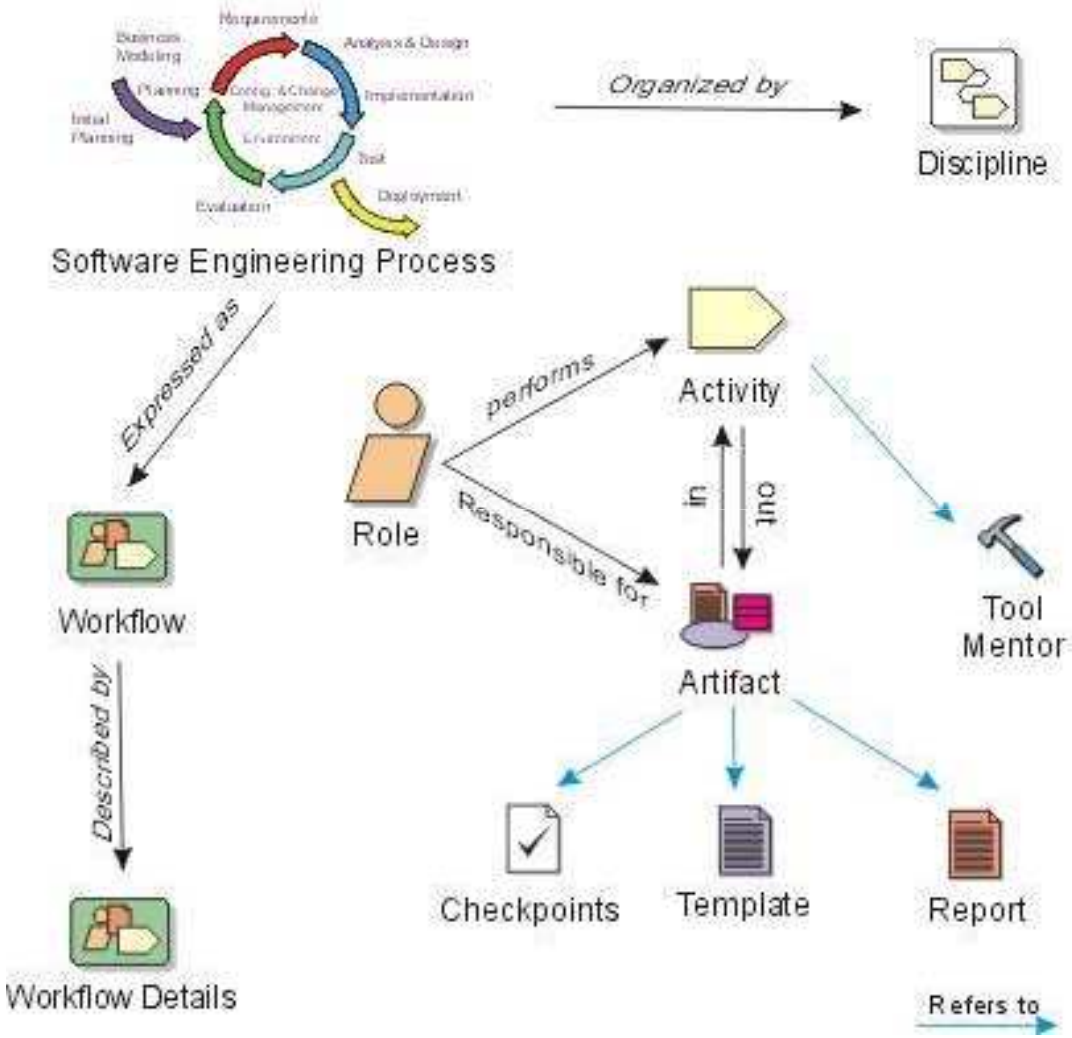
- Além dos modelos de ciclo de vida descritos, encontramos na literatura muitos outros modelos e cada um deles geralmente enfoca um determinado tipo de projeto específico;
- Podemos perceber que todos os modelos de ciclo de vida possuem particularidades próprias e apresentam vantagens e desvantagens. Assim, a escolha de um modelo depende do contexto do projeto onde ele será aplicado.

Conclusões - continua

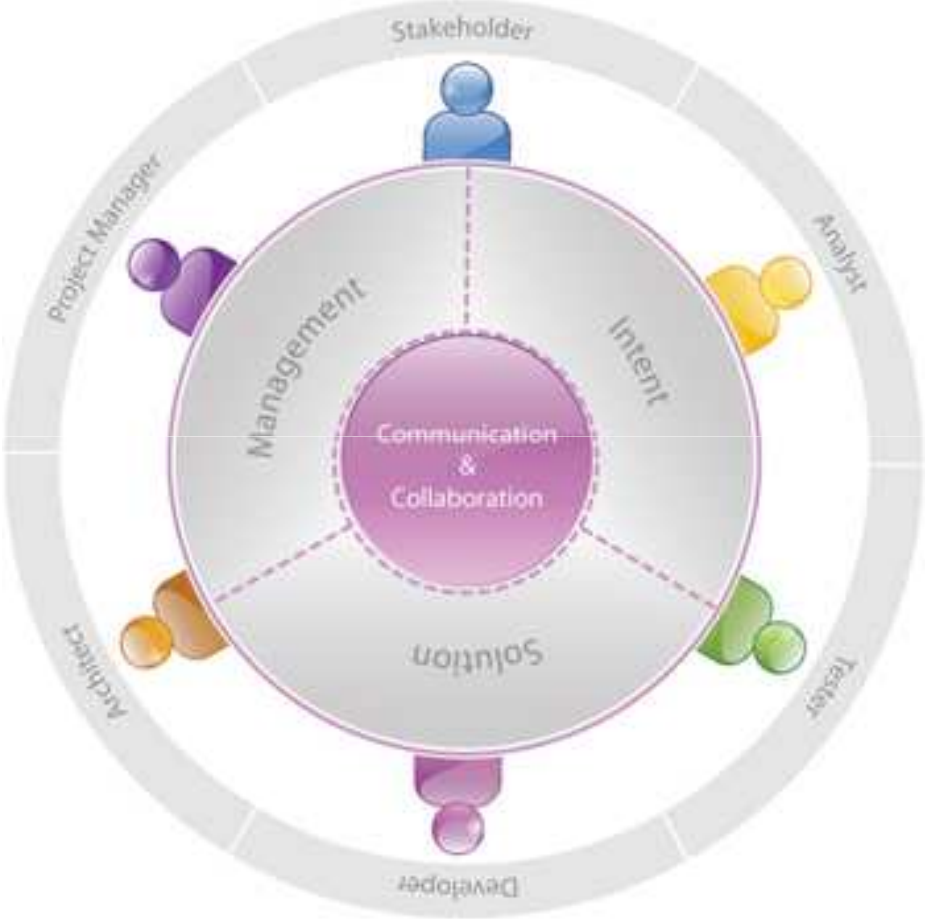
- Nenhum modelo de ciclo de vida é tido como ideal, pois aquele que é mais rápido e eficiente em certos projetos pode ser mais lento ou complexo quando o domínio do problema é trocado.
- O mais importante é que as empresas de desenvolvimento de software tenham algum modelo de ciclo de vida, ou melhor, um "catálogo" de modelos personalizado de acordo com a sua estrutura organizacional e projetos (demandas e expectativas dos clientes) para que consigam níveis de excelência em seus produtos e serviços.
- Vale ainda lembrar que esse é um dos pré-requisitos fundamentais para que elas almejem uma certificação do tipo CMM.

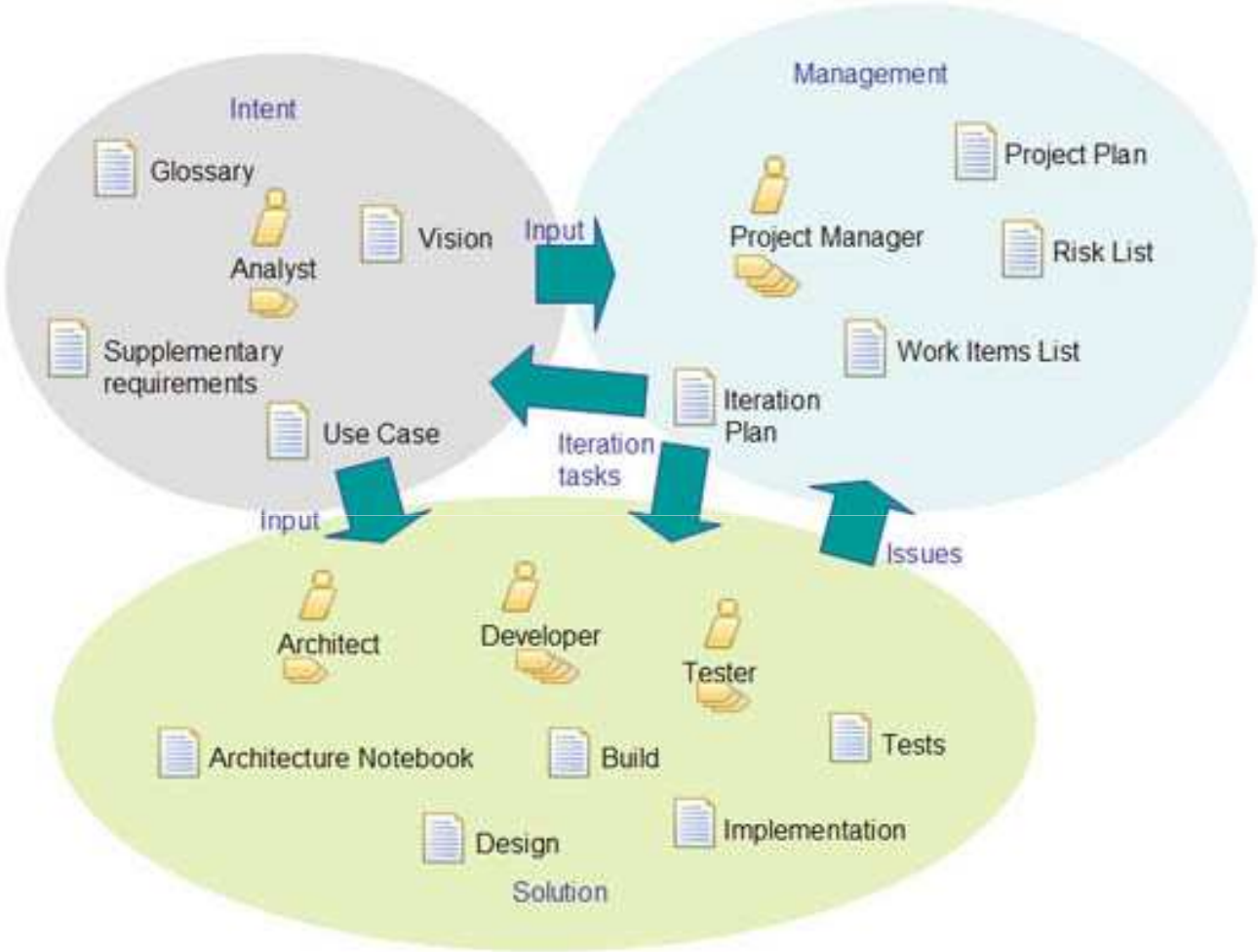
Processo RUP



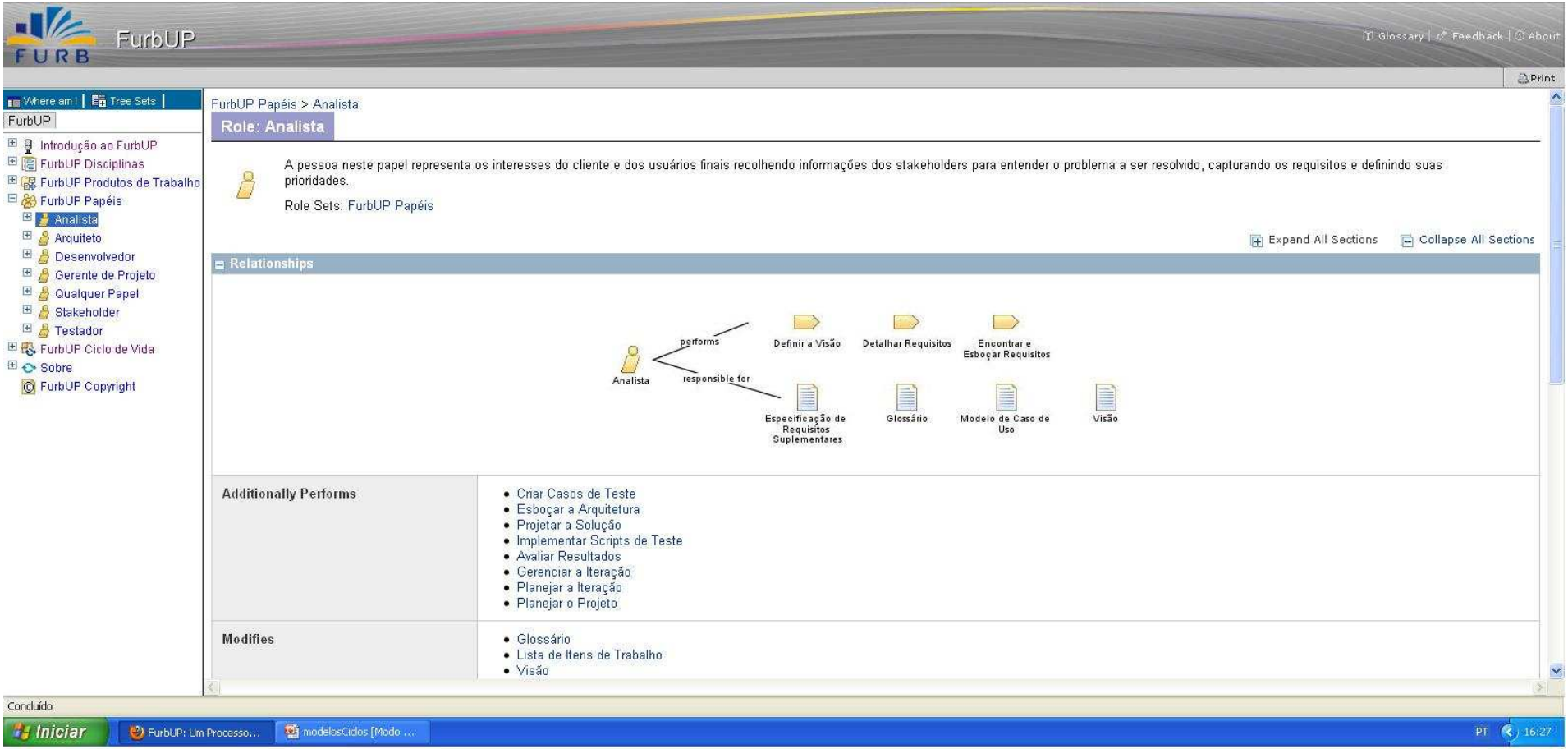

Organized by
Discipline
Software Engineering Process
Expressed as
Workflow
Described by
Workflow Details
Role
performs
Responsible for
Activity
in
out
Artifact
Tool Mentor
Checkpoints
Template
Report
Refers to

Open-Up


Stakeholder
Analyst
Tester
Developer
Architect
Project Manager
Management
Intent
Solution
Communication
&
Collaboration


Intent
Glossary
Analyst
Vision
Supplementary requirements
Use Case
Input
Management
Project Plan
Project Manager
Risk List
Work Items List
Iteration Plan
Iteration tasks
Input
Issues
Architect
Developer
Tester
Architecture Notebook
Build
Tests
Design
Implementation
Solution

campeche.inf.furb.br/lqs/site/furbup/



Manifesto Ágil - 2001

Valorizar:

Visão da Equipe e Disciplina mais do que *Indivíduos e Interações* (e mais do que *Processos e Ferramentas);*

Aprendizado Validado mais do que *Software em Funcionamento* (e mais do que *Documentação Abrangente);*

Descoberta do Cliente mais do que *Colaboração com o Cliente* (e mais do que *Negociação de Contratos*);

Iniciar as Mudanças mais do que *Responder às Mudanças* (e mais do que *Seguir um Plano*);

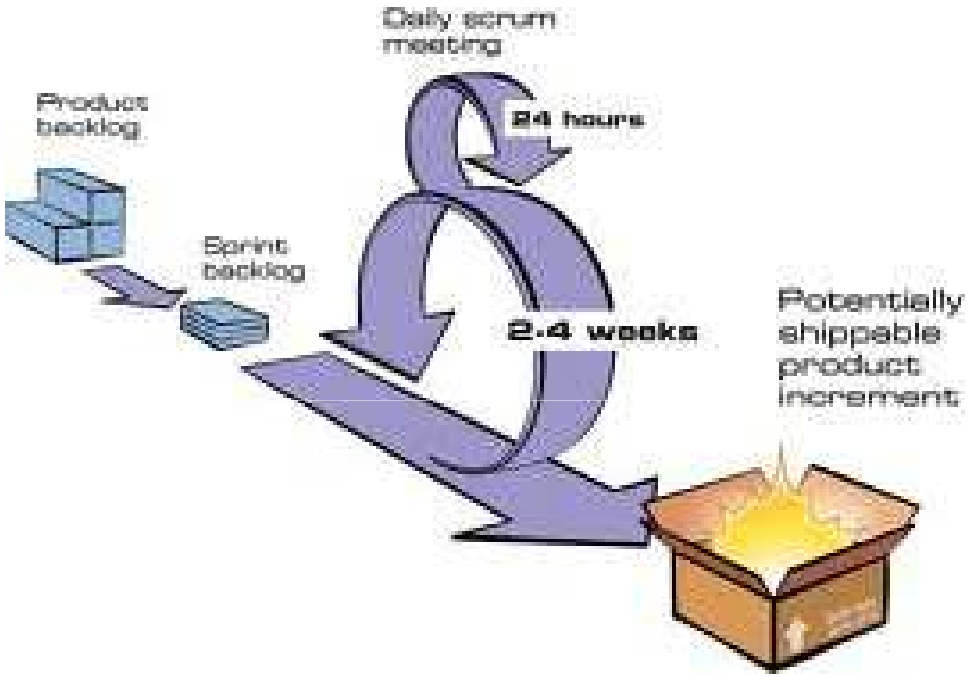
SCRUM

- Scrum é um framework Ágil de gestão de projetos usado para entregar aos clientes, de forma iterativa, incrementos de produto de alto valor.
- Scrum depende de equipes hábeis e auto-organizadas para entregar os incrementos do produto. Também depende de um cliente, ou Dono do Produto, que indique uma equipe com uma lista de funcionalidades desejadas, e que use o valor de negocio como mecanismo de priorização.


24 HORAS
3
CICLO
DE 2 A 4
SEMANAS
1
REQUISITOS
2
ATIVIDADES
4
SOFTWARE
RODANDO

Atividades no SCRUM

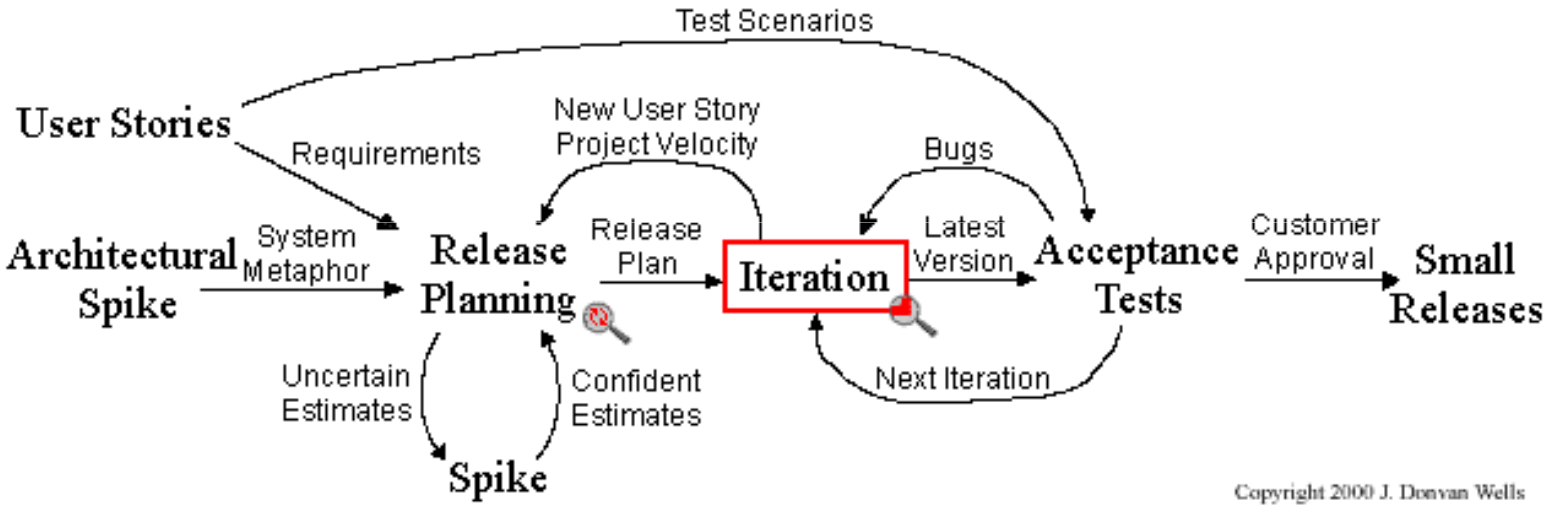
- No Scrum, os projetos acontecem em uma série de iterações, com um mês de duração, chamadas *incrementos (sprints)*.
- Scrum é ideal para projetos cujos requisitos mudam rapidamente ou são altamente emergentes. O trabalho a ser feito em um projeto Scrum é registrado nas Pendências do Produto (Product Backlog), que é uma lista de todos os desejos de mudança no produto.
- No inicio de cada incremento é feita uma Reunião de Planejamento de Incremento (Sprint Planning Meeting) na qual o Dono do Produto (Product Owner) prioriza as Pendências do Produto (Product Backlog), e a Equipe Scrum (Scrum Team) seleciona as tarefas que ela pode completar durante o próximo Incremento. Essas tarefas são então movidas das Pendências do Produto para as Pendências do Incremento.
- Durante um incremento, são conduzidas curtas reuniões diárias chamadas de Scrum Diário (Daily Scrum), que ajudam a equipe a manter-se no rumo.
- Ao final de cada incremento a equipe demonstra a funcionalidade concluída, na Reunião de Revisão do Incremento (Sprint Review Meeting).


Daily scrum meeting
Product backlog
24 hours
Sprint backlog
2-4 weeks
Potentially shippable product increment

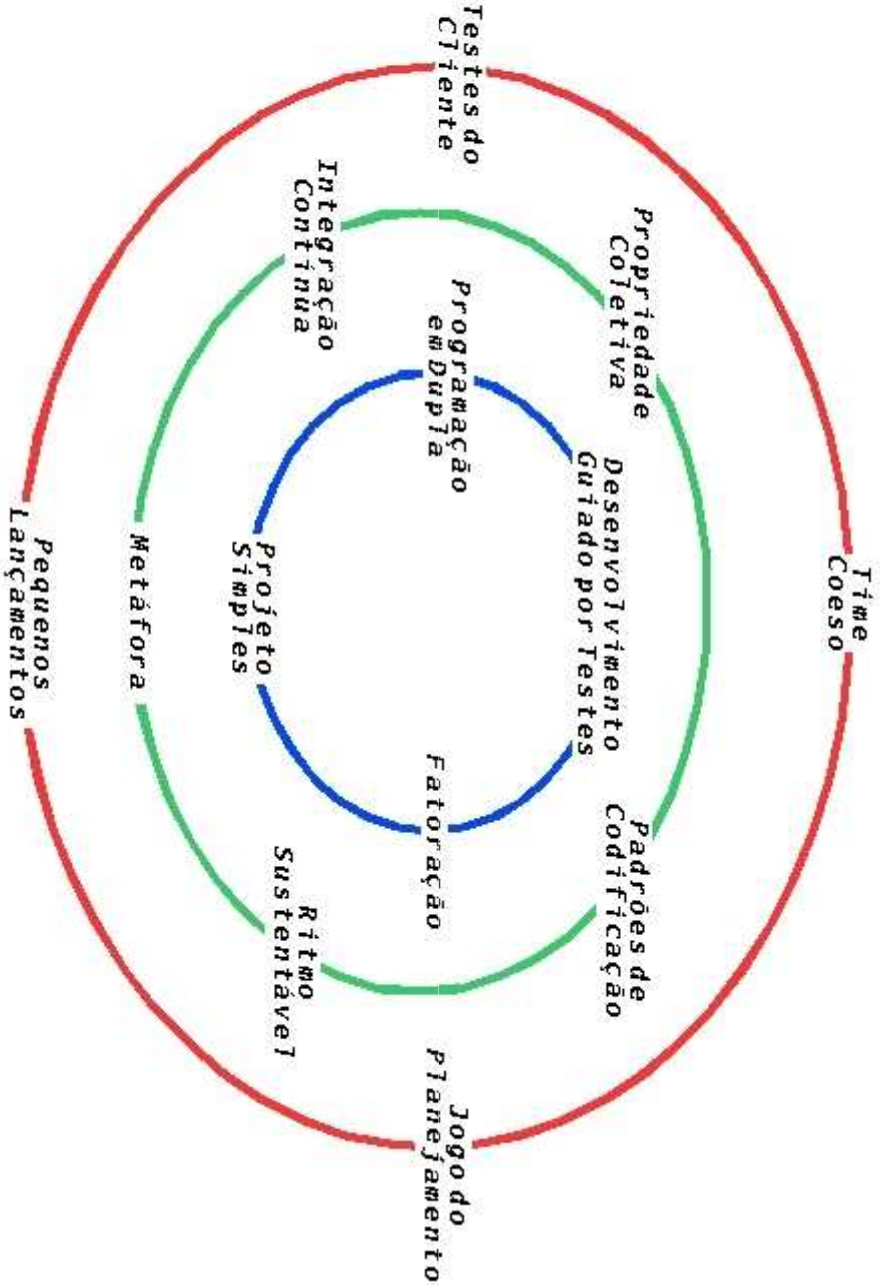
XP – programação extrema

- **Programação extrema** (do inglês *eXtreme Programming*), ou simplesmente **XP**, é uma metodologia ágil para equipes pequenas e médias e que irão desenvolver software com requisitos vagos e em constante mudança. Para isso, adota a estratégia de constante acompanhamento e realização de vários pequenos ajustes durante o desenvolvimento de software.
- Os cinco valores fundamentais da metodologia **XP** são: comunicação, simplicidade, *feedback*, coragem e respeito.
- A partir desses valores, possui como princípios básicos: *feedback* rápido, presumir simplicidade, mudanças incrementais, abraçar mudanças e trabalho de qualidade.

Extreme Programming Project



Práticas XP


Testes do Cliente
Propriedade Coletiva
Time Coeso
Jogo do Planejamento
Padrões de Codificação
Ritmo Sustentável
Fatoração
Desenvolvimento Guiado por Testes
Programação em Dupla
Projeto Simples
Metáfora
Integração Contínua
Pequenos Lançamentos

Estudar para Exercício 1

- Saber diferenças básicas entre **modelos de ciclo de vida**
- Selecionar o modelo mais adequado de acordo com tipo de aplicação e contexto
- Saber diferenças básicas entre **processos tradicionais de software e métodos ágeis**
- Selecionar o processo mais adequado de acordo com tipo de aplicação e contexto