O que é um processo de software?

- Um conjunto de atividades realizadas por pessoas cujo objetivo é desenvolvimento ou evolução de software e sua documentação.
- Atividades genéricas em todos os processos:
  - Especificação – o que o sistema deve fazer (funcionalidade) e quais as restrições
  - Desenvolvimento – produção do software
  - Verificação – avaliar correção, validação e outros aspectos de qualidade
  - Manutenção – mudanças no software
- Um modelo de processo de software é uma representação abstrata das atividades, papéis e artefatos, cronograma.

Processo 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

Modelo X Processo

- Um modelo é algo teórico, um conjunto de possíveis ações.
- O processo deve determinar ações práticas a serem realizadas pela equipe como prazos definidos e métricas para se avaliar como elas estão sendo realizadas.
- Define *quem* faz o *que*, *quando* e *como.*

Modelo + Planejamento = Processo

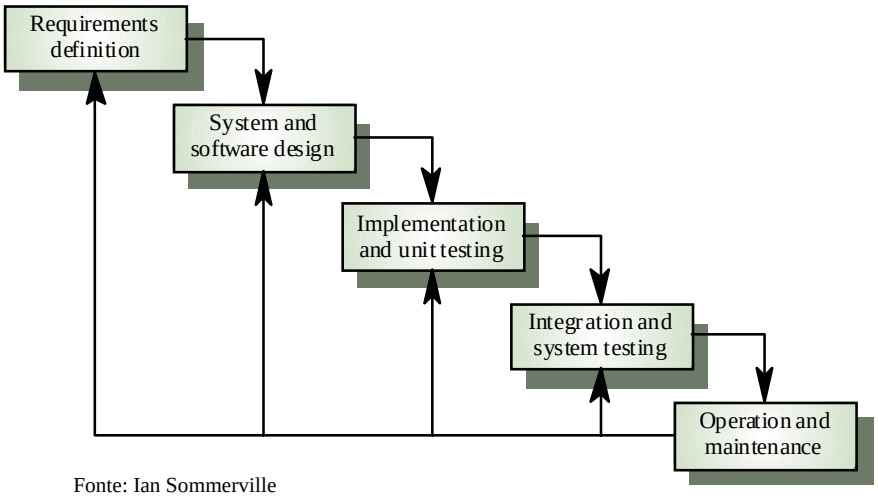
Atividades, Artefatos, Marcos e Entregas

- Um processo é organizado em atividades.
- Atividades são de responsabilidade de um membro da equipe (trabalhador).
- Atividades devem gerar um artefato de saída, que possa ser verificado, e podem requisitar um artefato de entrada.
- Um artefato é um modelo, documento ou código produzido por uma atividade.
- Uma entrega (liberação) é um artefato entregue ao cliente
- Um processo deve estabelecer uma série de marcos.
- Um marco é um ponto final de uma atividade de processo.

Modelos de Processo

- Um modelo de processo ou método define um conjunto de atividades específicas.
- Principais modelos:
  - Cascata (Waterfall)
  - Espiral (Spiral)
  - Evolutivo
  - Incremental
  - Processo Unificado da Rational (Rational Unified Process – RUP)
  - eXtreme Programming (XP)

Modelo Cascata



Fonte: Ian Sommerville

Atividades típicas do modelo Cascata

- Análise e definição de requisitos
  - Objetivos, funções e restrições são definidos, com ajuda de clientes e usuários, e servem como uma especificação do sistema, indicando o que deve ser implementado.
- Design de sistemas e software
  - Envolve a descrição do sistema e do software em termos de unidades abstratas e de suas relações, indicando como o software deve ser implementado.
- Implementação e testes de unidade
  - As unidades do software devem ser codificadas e testadas individualmente.
- Integração e testes de sistema
  - A unidades são integradas e testadas
- Entrega, operação e manutenção
  - O sistema é instalado e colocando em operação. A manutenção envolve a correção de erros e evolução do sistema para atender a novos requisitos.

Características do modelo Cascata

- Divisão inflexível do projeto em estágios distintos. A fase seguinte só deve iniciar quando a anterior tiver sido concluída e aprovada pelas partes envolvidas.
- Por exemplo,o design apenas deve começar quando os requisitos estiverem totalmente definidos e aprovados
- Dificuldade em realizar mudanças com o processo em andamento – requisitos sempre mudam.
- O modelo em cascata é apropriado quanto se tem um entendimento claro dos requisitos.
- Oferece maior previsibilidade de prazos e custo: melhor planejamento e gerenciamento.
- A engenharia do sistema segue este tipo de modelo.

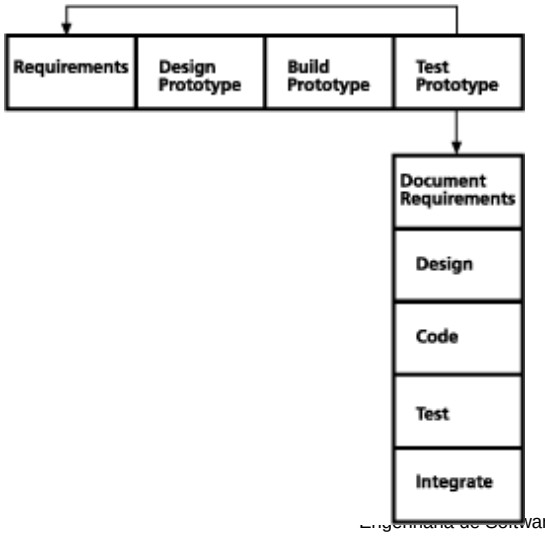
Prototipação

- Abordagem baseada numa visão evolutiva do desenvolvimento de software, afetando o processo como um todo
- Protótipo de software é um sistema que...
  - funciona
  - não tem tempo de vida definido
  - pode servir a múltiplos propósitos
  - deve ser construído rapidamente e com baixo custo
  - é parte integrante de um design centrado no usuário, para avaliação e modificação

Tipos e objetivos dos protótipos

- Tipos
  - Falso
  - Autêntico
  - Funcional
  - Sistema piloto
- Objetivos
  - Exploração
  - Experimentação
  - Evolução

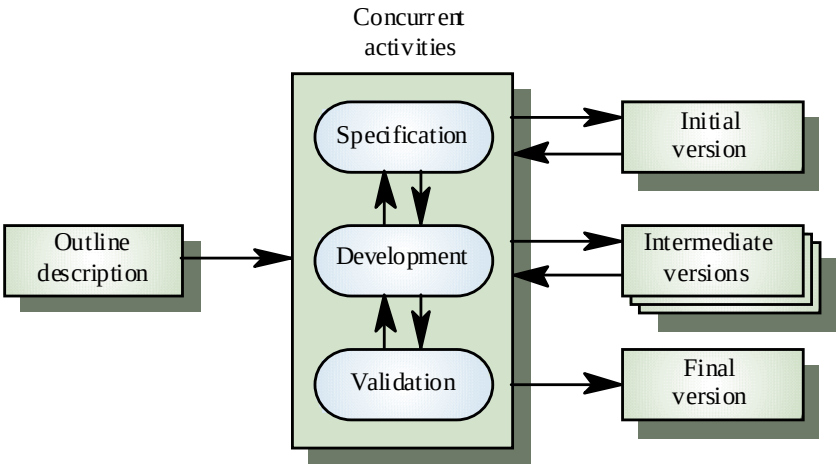
Prototipação



Desenvolvimento Evolutivo

- Desenvolvimento Exploratório
  - A partir de requisitos iniciais, é elaborado um protótipo que permite, junto ao cliente, explorar novos requisitos.
- Prototipagem
  - Protótipo descartáveis
  - Protótipo evolutivos - evolui para o produto final
- Características
  - Útil quando os requisitos estão obscuros
  - Especificação é construída gradativamente
  - Possibilitam um rápido desenvolvimento da aplicação
  - Requer ferramentas específicas
  - Os sistemas são freqüentemente mal-estruturados e mal-documentados.
  - Processo não é claro, dificuldade de planejamento e gerenciamento

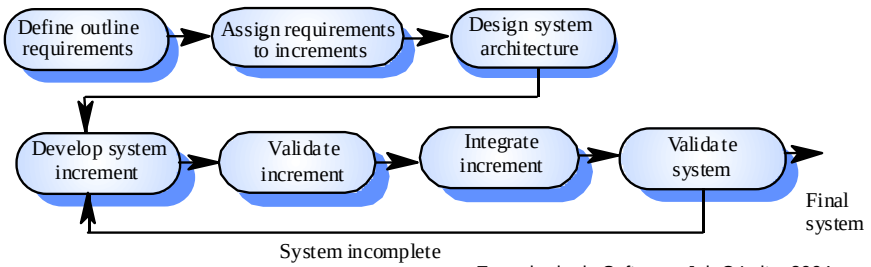
Desenvolvimento Evolutivo



Fonte: Ian Sommerville

Desenvolvimento incremental

- Desenvolvimento iterativo, em ciclos que permitem revisões de atividades anteriores.
- O sistema é particionado em partes independentes que podem ser entregues à medida que forem ficando prontas e avaliadas.
- A arquitetura do sistema deve possibilitar a entrega das partes independentes.



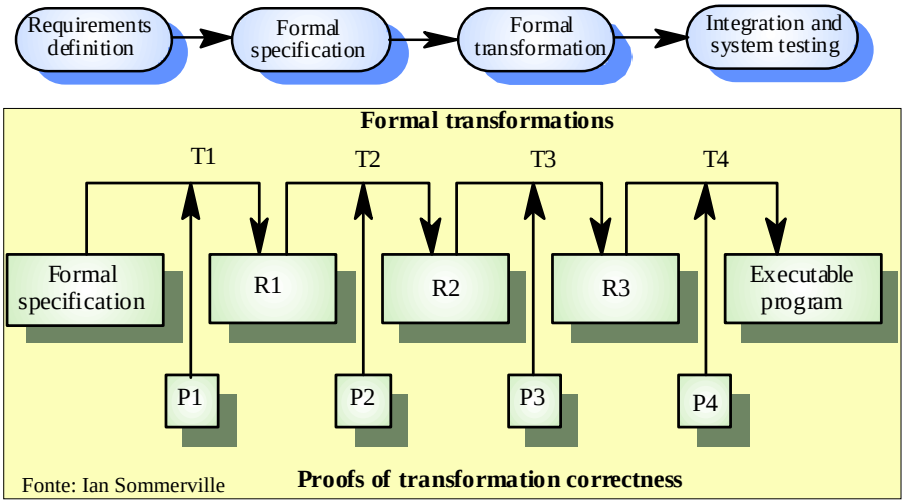
Características do modelo incremental

- Os incrementos funcionam como protótipos do sistema.
- A avaliação do cliente pode ser feitas a partir das experiências do usuário com as entregas parciais.
- Novos requisitos podem ser incorporados aos outros incrementos do sistema.
- Deve-se identificar funcionalidade prioritárias que serão desenvolvidas primeiro.
- Os processos de cada incremento podem ser independentes. Pode-se utilizar Cascata.
- Risco menor de fracasso completo do sistema.
- A funções entregues primeiro são testadas mais vezes, à medida que os incrementos são entregues.

Desenvolvimento formal de sistemas

- Baseado na transformação de uma série representações matemáticas da especificação em um programa executável.
- A especificação de requisitos, informal ou semi-formal, é redefinida em uma especificação formal detalhada, utilizando uma notação matemática:
  - VDM, Z, B, Larch e outras
- As atividades de design, implementação e testes são substituídas por um processo transformacional.
- As transformações devem ser matematicamente corretas, baseadas em provas-de-correção.
- As provas são longas e impraticáveis para sistemas de grande porte.
- É adequado a sistemas com exigências rigorosas de segurança, confiança e garantia.

Transformações formais

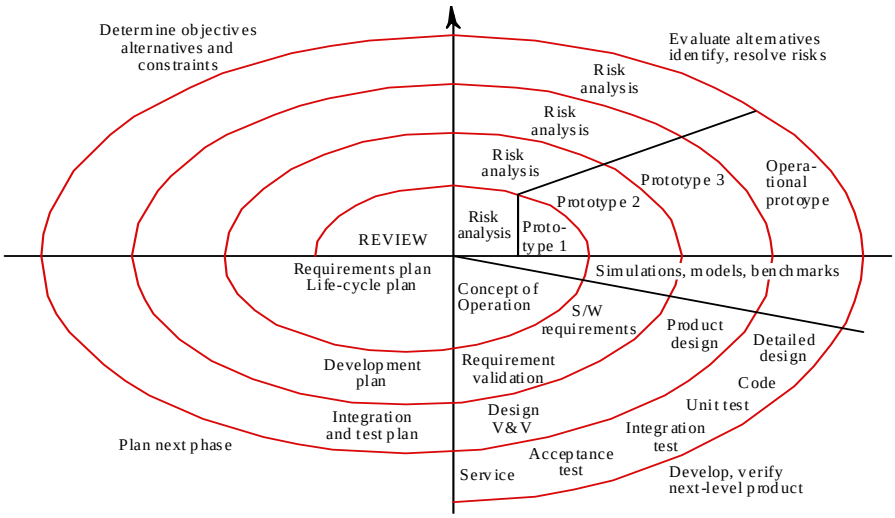


Fonte: Ian Sommerville

Modelo Espiral

- O processo é representado como uma espiral. Cada ciclo do espiral é uma fase do processo
- Cada ciclo determina quatro etapas fundamentais:
  - Definição de objetivos, alternativas e restrições
  - Análise e redução de riscos
  - Desenvolvimento e validação
  - Planejamento do próximo ciclo
- Protótipos são construídos em cada ciclo.
- Não há fases fixas pré-definidas. Elas são definidas de acordo com os objetivos.
- É um meta-modelo: qualquer modelo pode ser derivado a partir do modelo espiral

Modelo Espiral (Spiral)



Programação eXtrema (XP)

- Abordagem baseada no desenvolvimento e entrega de pequenas partes da funcionalidade do software.
- As partes devem ser incrementadas e requer a melhoria constante do código (re-trabalho).
- Requer o envolvimento constante do cliente
- Os sistema é concebido a partir de uma metáfora e descritos em *estórias do usuário*.
- É necessário a definição de testes de aceitação.
- Não existe um processo de design tradicional e não são gerados modelos da arquitetura do software
- A programação é feita por pares de programadores. Prática do codifica-e-conserta.

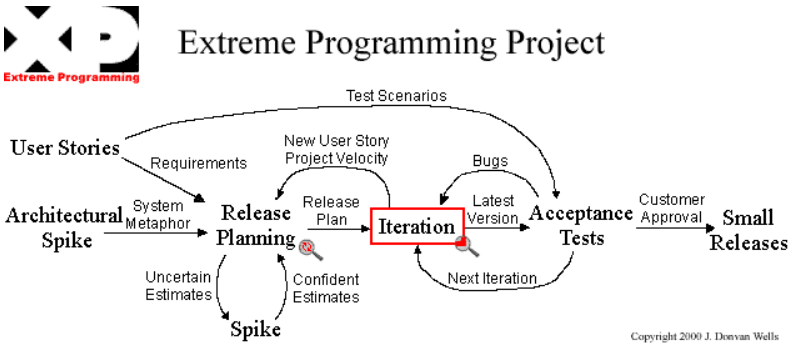
Regras e Práticas - 1

- Planejando
  - Estórias do usuário
  - Planejando liberações (releases) e pequenas liberações
  - Dividir projetos em iterações (ciclos)
  - Medindo velocidade do projeto
  - Dinâmica de pessoal
  - Reuniões diárias em pé
- Projetando (designing)
  - Simplicidade (não adicione funcionalidades antes do tempo)
  - Metáfora
  - Cartões CRC (Classes, Responsabilidades e Colaboração)
  - Re-fabricar (refactor)

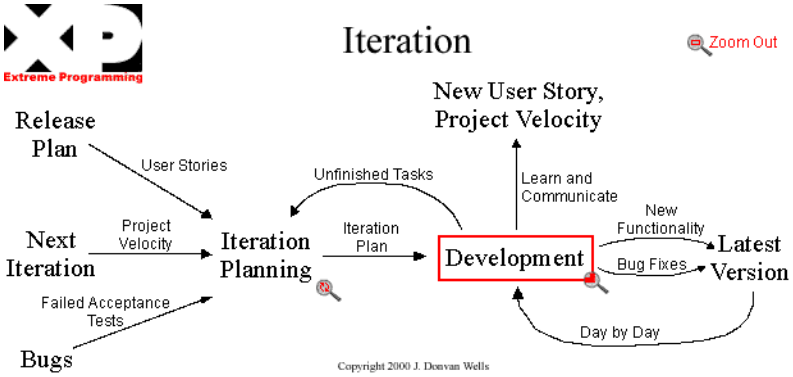
Regras e Práticas - 2

- Codificando
  - O cliente deve estar sempre disponível.
  - Programação em pares.
  - Codificar de acordo com padrões acordados.
  - Codificar testes de unidade primeiro.
  - Integrar com freqüência.
  - O código é propriedade coletiva.
  - Não atrase.
- Testando
  - Todo código deve ter os testes de unidade.
  - Cada unidade deve ser testada antes de liberada.
  - Quando um erro é encontrado, testes devem ser realizados.
  - Testes de aceitação devem ser freqüentes.

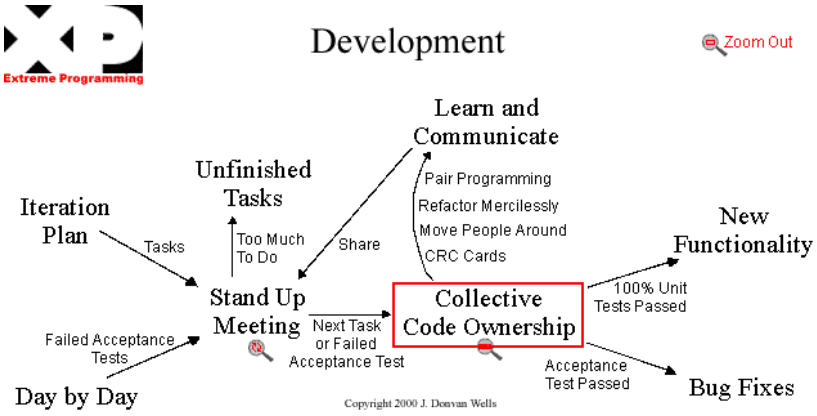
XP - Projeto



XP - Iteração



XP - Desenvolvimento



XP – Propriedade coletiva do código