



cesar.edu

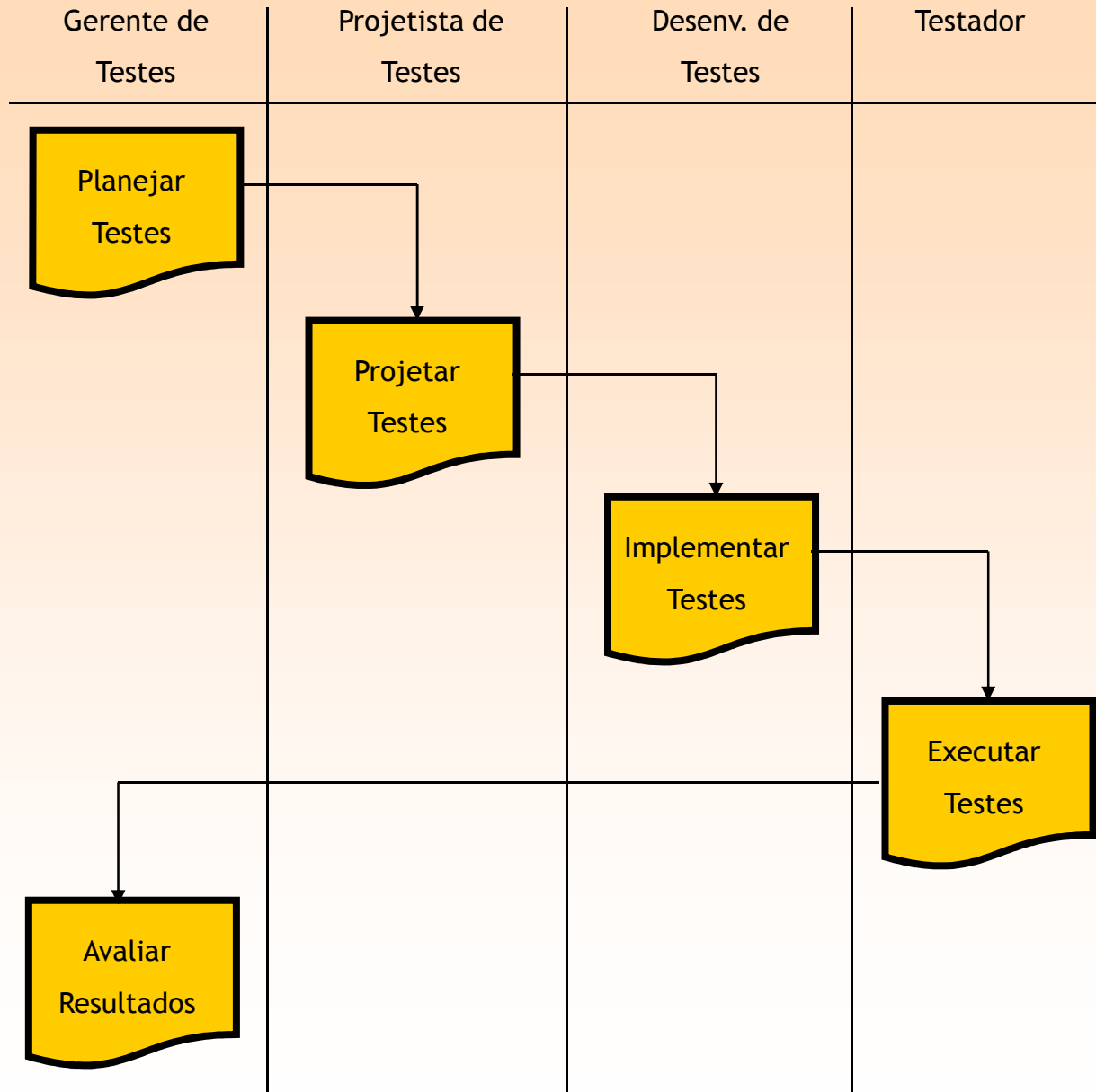
Testes integrados durante o ciclo de desenvolvimento: diferentes abordagens de acordo com o ciclo de vida do projeto

15.07.08



Testes NÃO é só execução!!!!!!!!!!!!

Processo de Testes





As atividades de testes são inseridas no contexto das atividades de engenharia de software

TODO software segue um ciclo de vida

- Existem vários modelos de ciclo de vida que podem ser adotados durante o desenvolvimento

O modelo de ciclo de vida usualmente direciona

- Quais testes serão rodados
- Quando os testes serão rodados

Alguns possíveis modelos

- Cascata
- V
- Incremental
- Codificar e testar



Fases seqüências que produzem o software

Ao final de cada fase uma “revisão interna” deve validar se é possível passar a fase seguinte

A princípio só se deve passar a fase seguinte com a anterior finalizada

Foco principal é garantir que a especificação é bem feita antes de iniciar as fases seguintes



O modelo em cascata é ideal para softwares bem conhecidos

- Sem grandes surpresas durante a especificação
- Tecnologia bem conhecida da equipe

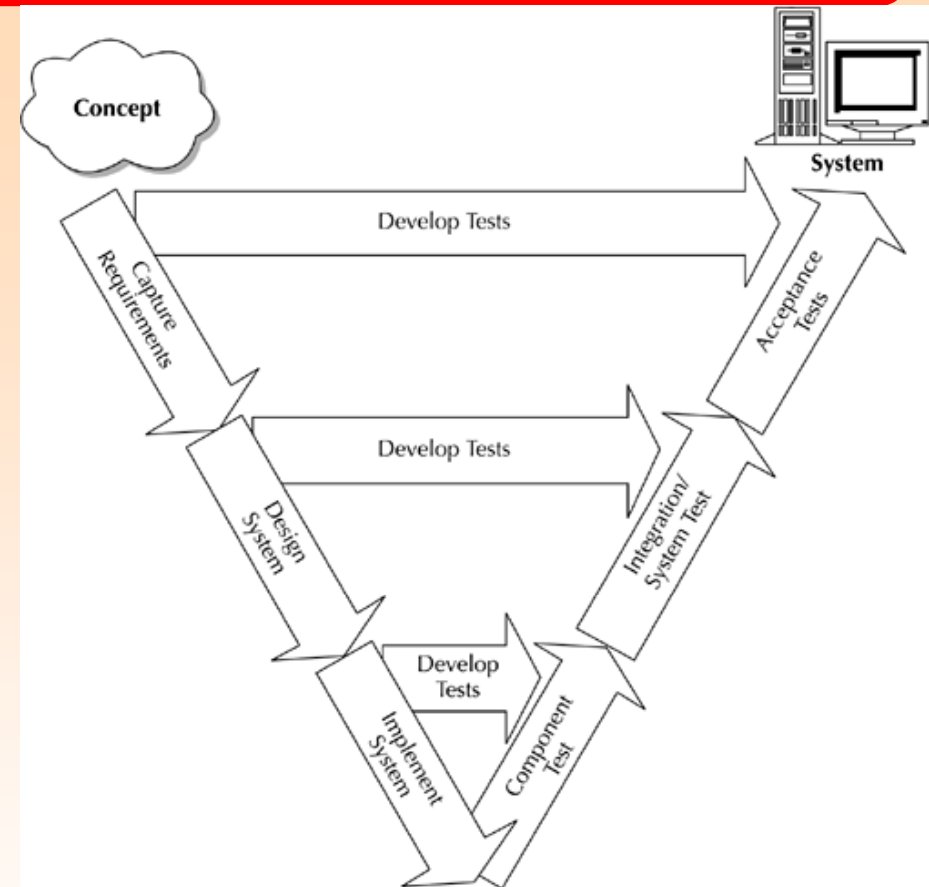
A fase de testes é bem natural e é executada após toda a codificação

Principal vantagem para os testes

- A especificação normalmente é bem feita e detalhada

Principal problema

- Os testes só são executados no final
- O custo de corrigir um problema é mais alto
- Bugs não são antecipados



Variação do modelo em cascata

Associa diferentes tipos de teste de acordo com a fase do ciclo de vida

- Testes de aceitação usualmente estão associados aos requisitos
- Testes de Integração usualmente estão associados ao design
- Testes de componente usualmente estão associados ao código



O modelo V define diferentes ESTÁGIOS de testes para serem executados

Pode ser visto como uma extensão do modelo em cascata

- Ainda existe a ênfase na especificação
- Cada fase “precisa” ser finalizada antes do início da fase seguinte
- Também é ideal para softwares com requisitos conhecidos

A grande vantagem do modelo é:

- Identificar os diferentes estágios de teste que validam aspectos do ciclo de vida do software
- Permitir um melhor planejamento dos testes que precisam ser executados
- “Quebrar” os testes em diferentes focos

A principal desvantagem é:

- Os testes ainda são rodados apenas após o código estar pronto

Modelo Incremental



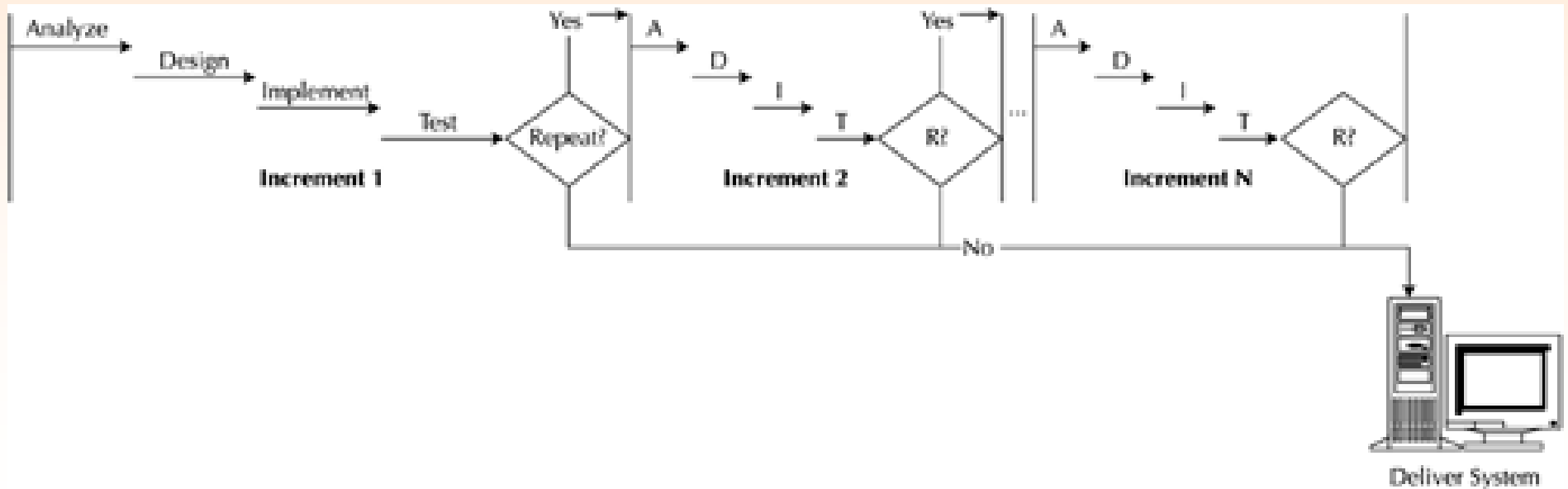
Desenvolvimento é “quebrado” em várias iterações

Cada interação pode ser vista como uma pequena cascata

Cada interação passa por TODAS as fases do ciclo

- Em maior ou menor escala

Requisitos são adicionados ao software em cada iteração





O modelo incremental busca usar as vantagens do cascata

- Boa especificação

Mas sendo realista

- Nem toda a especificação pode estar pronta no início do desenvolvimento

As iterações podem ser repetidas até que o software esteja estável

Ao final cada iteração é necessário

- Determinar os objetivos da próxima iteração
- Avaliar os riscos
- Avaliar os testes
- Avaliar os requisitos que já estão implementados
- Planejar a iteração seguinte



A principal vantagem para os testes é

- Os testes são rodados várias vezes (ao final de cada iteração)
- Os bugs podem ser antecipados
- Os testadores usualmente podem (e devem ser) envolvidos no projeto desde o início

A principal desvantagem para os testes é

- Algumas vezes não fica claro que testes devem ser rodados ao final de cada iteração
- O esforço de planejamento de testes é maior
 - Mas esse esforço normalmente é compensado
- É necessária uma equipe de testes mais experiente para planejar e executar testes neste modelo



Parte de uma especificação informal

Ênfase em codificação

- Não existe ênfase em outras fases do desenvolvimento

Como a especificação não é clara os testes também não são

- Mas os testes são executados várias vezes

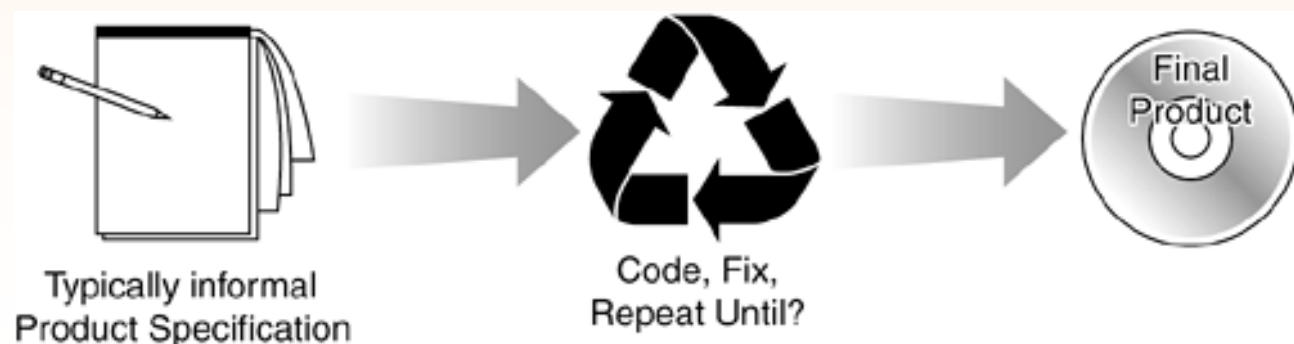
É ideal para projetos pequenos e com equipe pequena

Principal vantagem para os testes

- Vários ciclos de testes
- Custo de correção não é alto

Principal desvantagem para os testes

- Planejamento e qualidade dos testes normalmente não é boa





Muito parecido com o codificar e testar

Foco em fechar várias versões do software

- Integração contínua

Especificação de requisitos informal

- Mas o cliente final DEVE ficar próximo a equipe de desenvolvimento para validar os requisitos

Mas os testes

- São mais formais
- “Devem” ser automatizados
 - Garantir que sempre serão rodados
- São executados a cada integração
- Cliente pode participar da validação



Principal vantagem para os testes

- **Automatização**
 - “Sai” a figura do testador e “entra” a do desenvolvedor de testes
- **Mais fácil de executar os testes**

Principal desvantagem

- **É necessário o uso de uma ferramenta para automatização**
- **Custo para automatizar e manter os testes pode ser bem alto**



**É possível substituir totalmente o testador
por testes automatizados?**



Test Driven Development

- Modelo de Desenvolvimento Orientado a Testes

É uma das principais técnicas associadas a modelos ágeis

A idéia é que os testes sejam desenvolvidos antes do código

- TFD: Test First Design

O passos para usar TDD são:

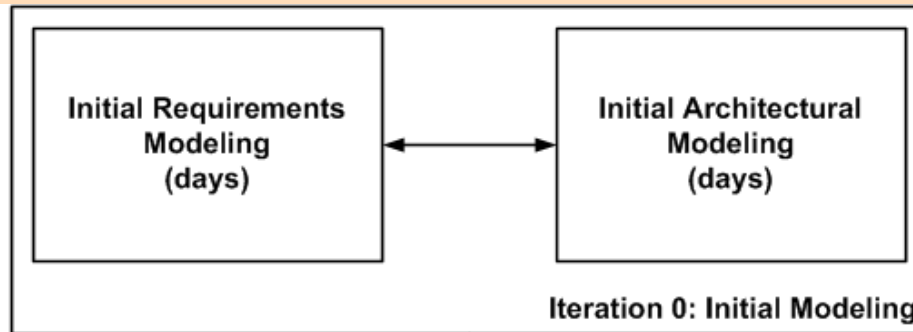
- Escreve um caso de teste e executa um caso de teste
- Falhando, implementa o código para passar no teste
- Passando, continua o desenvolvimento e escreve um novo caso de testes

TODA a fase de implementação de um modelo ágil pode ser feita com TDD

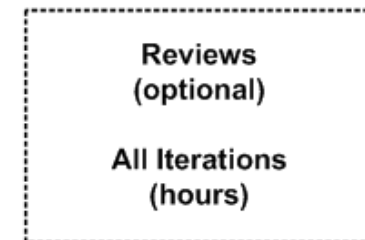
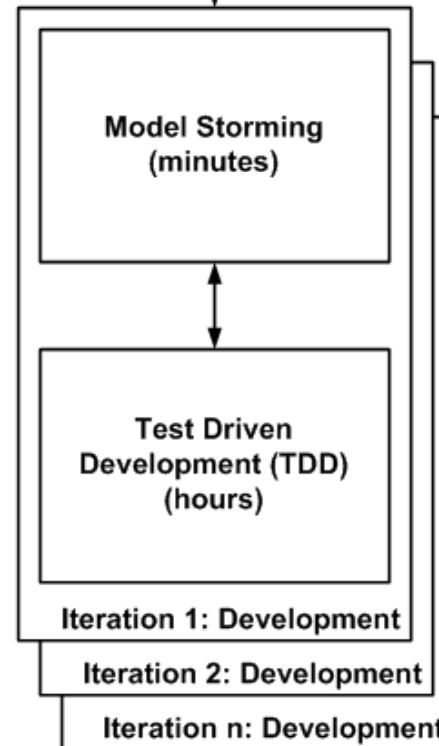
Modelo Ágeis: TDD



- Identify the high-level scope
- Identify initial "requirements stack"
- Identify an architectural vision



- Work through specific issues on a JIT manner
- Stakeholders actively participate
- Requirements evolve throughout project
- Model just enough for now, you can always come back later



- Develop working software via a test-first approach

Copyright 2003-2007
Scott W. Ambler

[Beck 2003; Astels 2003]



Projetos que usam TDD precisam de testadores?



O ciclo de vida do software é uma decisão fundamental para o desenvolvimento de qualquer software

Essa decisão impacta diretamente nas atividades de testes

As atividades de testes compreendem bem mais do que apenas executar os testes



Fim