práticas do eXtreme Programming: desenvolvimento guiado pelos testes

josé augusto fabri
fabri@femanet.com.br

Atividade

- Desenvolva um programa em C que leia os seguintes campos:
  - Código
  - Nome do funcionário
  - Endereço
  - Cidade
  - CEP
  - E-mail
  - Salário.
  - Leia a quantidade de horas extras para o calculo do salário líquido.
  - Slq = [Salário + ((Salário / 220) * horasExtras)] * 0.85

Tópicos

- Introdução;
- Por que devemos testar?
- Testar é investir;
- Testando no XP.

Introdução

- Testar: Atividade valorizada depois que o sistema entra em produção. Pois, com o sistema em produção vários problemas começam a surgir.
- Grande parte dos projetos deixam os testes para serem feitos no final (quando são feitos).
- Porém os projetos costumam atrasar e o período de testes é suprimido.
- XP: O ato de testar deve ser parte natural do ato de programar.

Por que devemos testar?

- Ao programar trabalhamos com um cenário complexo e infinito.
- A interação dos elementos programáveis pode gerar uma infinidade de resultados, alguns dos quais indesejáveis e imprevisíveis.
- Embora os desenvolvedores tentem verificar todos os cenários enquanto programam, é comum aparecer erros devido as situações completamente imprevistas, ou devido a erros de lógica, falta de atenção, interpretação incorreta dos requisitos, entre outros fatores.

Por que devemos testar?

- É impossível evitar que aconteçam defeitos no sistema ao longo do projeto.
- É possível fazer alguma coisa em relação à quando estes defeitos são detectados e qual o impacto que causarão no cronograma do projeto.
- Se os desenvolvedores erram e descobrem segundos depois, eles, também, descobrem rapidamente o que gerou o erro.
- Porém se passar muito tempo para descobrir o erro, este descobrimento será lento.

Por que devemos testar?

- Enfim,
  - Quando os testes fazem parte do desenvolvimento, os programadores recebem feedback rapidamente sobre aquilo que estão fazendo.
  - Desta forma, aprendem mais, resolvem defeitos de forma rápida e têm mais tempo para desenvolver funcionalidades e gerar valor para o cliente.

Testar é Investir

- Ao testar, estamos fazendo um investimento.
- O que esperemos deste investimento?

QUE ELE GERE RETORNO NO FUTURO

Testar é Investir

- No caso dos testes, este retorno acontece quando:
  - O desenvolvedor testa alguma coisa que deveria falhar e ela funciona.
  - O teste falha quando já vinha funcionando normalmente, ou seja, surgiu um defeito no software e o teste detectou.
- A RAPIDEZ NA DETECÇÃO DE DEFEITO, NÃO GERA SESSÕES DE DEPURAÇÃO POSTERIORES, ESTRATÉGIA ESTA QUE TOMA UMA BOA PARTE DE TEMPO DO PROJETO.

Testar é Investir

- Metáfora:
  - Nossa vida: Estamos saudáveis ou não.
  - Gostaríamos de sempre estar com saúde.
  - Como buscar isto?
    - Prevenir e não remediar.
  - Prevenção: Bons hábitos na nossa vida: alimentação, exercícios físicos, estrutura familiar, consulta regulares aos médicos.
  - Buscar um plano de saúde também é uma prevenção.
  - Todo mundo sabe que é muito mais barato prevenir do que remediar.
    - Se necessitar de uma UTI por alguns dias, quando chegar a conta você lamentará por não ter tido um bom plano de saúde.

Testar é Investir

- A necessidade de depurar aparece quando o software está doente, ou seja, com um ou mais erros.
- Depurar é uma atividade extremamente custosa, significa tempo e dinheiro jogado na lata de lixo.
- O que torna a depuração custosa é o tempo que existe entre o momento em que o erro é inserido e o momento em que ele é detectado.
- Se o tempo for muito elástico, o programador tem que recuperar o contexto e entender o que havia sido feito no código.

Testar é investir

- O desenvolvimento guiado por teste é o caminho para a prevenção.

- Princípios fundamentais:
  - O teste expõe o defeito assim que ele entra no software.
  - Evita que o erro apareça em uma outra parte do sistema que não é aquela na qual você está trabalhando. “A recorrência de erros”.
  - A programação guiada pelos testes busca reduzir o tempo total dedicado a depuração em um projeto. Nota: Este tempo deve ser o mínimo.

Testando no XP

- O XP trabalha com dois tipos de testes:
  - Teste de unidade;
  - Teste de aceitação.
- Teste de unidade são realizados sobre um conjunto de funcionalidades para verificar se os resultados gerados são corretos.
- Já o teste de aceitação é caracterizado pelo teste das estórias. Cada estória é testada na presença do cliente.

Testando no XP

- Em ambos os casos, os testes devem ser gerados em primeiro lugar, isto é, devem ser escritos antes da definição estrutural do programa.
- Seria interessante utilizar algum programa de automatização dos testes. Isto é, programas que testam programas (jUnit).
- Os testes são alimentados, constantemente, pelos programadores, isto ocorre com a evolução do programa.

Testando no XP

- Quando escrever os testes?
  - Se a interface de um método ou de um programa não for clara, você deve escrever o teste antes de implementá-lo.
  - Se você verifica uma situação incomum na qual o código deva funcionar da forma como está escrito, você deve escrever um teste.
  - Se você encontrar problema em um código que já está escrito, você deve escrever um teste.

Testando no XP

- Ah... Mas como eu vou escrever um teste?
- Simples, veja o artefato que vai te ajudar. Click aqui para acessar o artefato.

Testando no XP

- Teste de aceitação:
  - Execução de todos os casos de testes gerados na presença do cliente no momento da implantação do software.
  - Busca simular a ação do usuário durante a utilização do software.

dúvidas

josé augusto fabri
fabri@femanet.com.br