

PROJETO, IMPLEMENTAÇÃO E TESTE DE SOFTWARE

Aula 06 – Implementação de Software

Professor Fabricio Freire



METAS DE APRENDIZAGEM

- Introdução
 - Recapitulando
- Depuração de código
- Asserções e Programação Defensiva
- Otimização de desempenho
- Refatoração

- Atividades Realizadas
- Características
- Estilo de programação e codificação
- Comentários



ARQUITETURA ORTOGONAL

Conceito do final do anos 70 (David Parnas)

Abordagem de componentes (modular)

Aplicação em softwares complexos

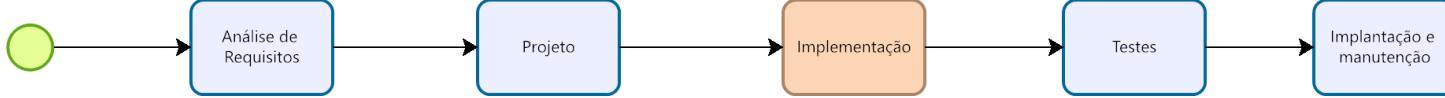
Facilita entendimento e manutenção

Oferece flexibilidade e escalabilidade



ONDE ESTAMOS?

Processo de Desenvolvimento de Software



DEPURAÇÃO DE CÓDIGO

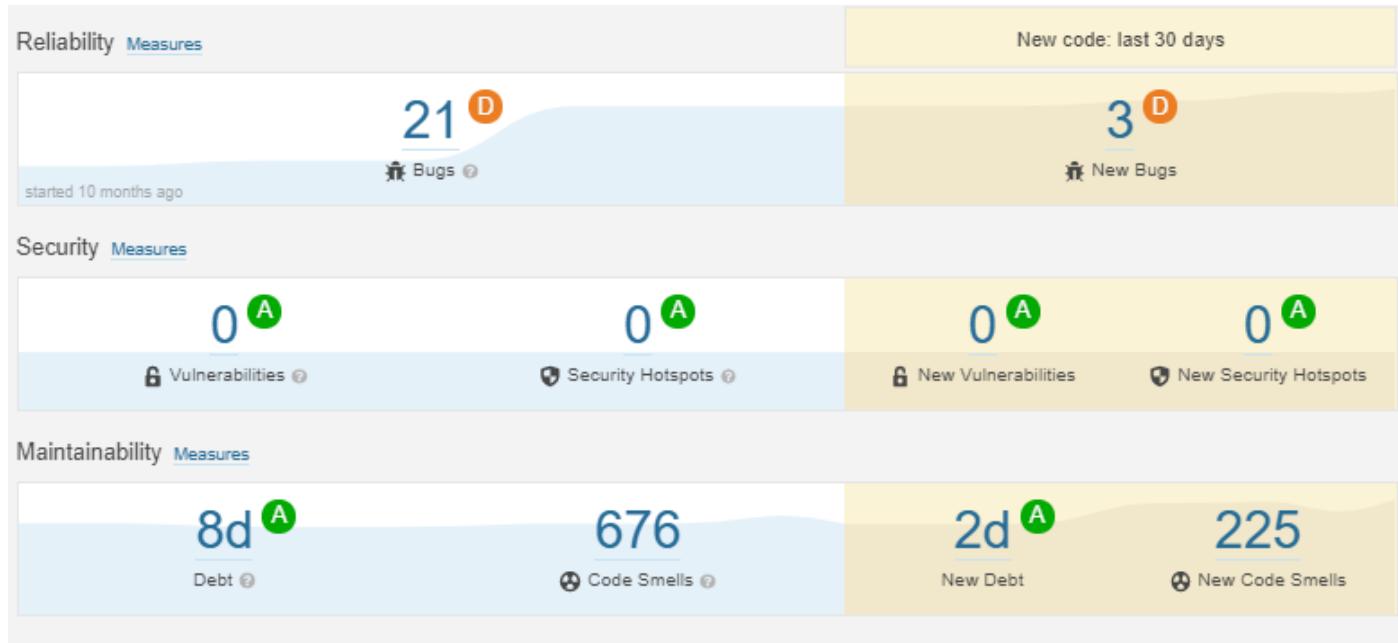
“depuração é o ato de localizar e corrigir erros no código. Os erros geralmente são descobertos através de testes, mas podem ser encontrados por outros meios, incluindo as inspeções de códigos e por meio do uso normal do programa”

Tsui e Karam (2013, p. 139)

DEPURAÇÃO DE CÓDIGO

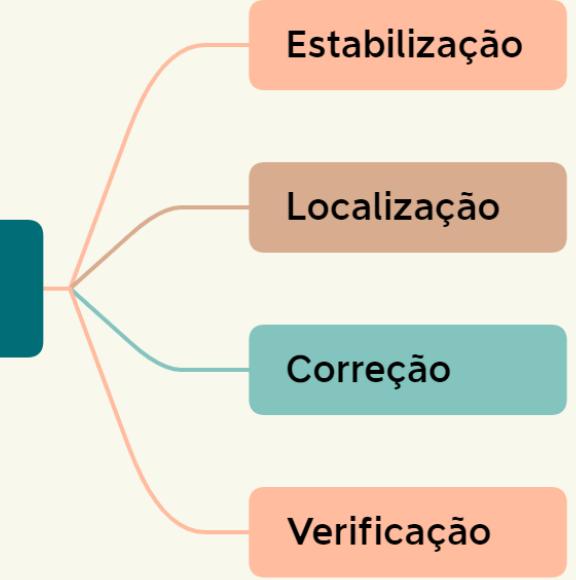
- Correção de Bugs – problemas no funcionamento
- Eliminação de Code Smell – pontos de melhoria
- Diminuição da dívida técnica – custo futuro associado à manutenção de um software mal desenvolvido.

DEPURAÇÃO DE CÓDIGO - SONARQUBE



FASES DO PROCESSO DE DEPURAÇÃO

Fases Depuração



FASES DO PROCESSO DE DEPURAÇÃO

Estabilização

- Reprodução do erro em uma configuração particular
- Identificar as condições para ocorrência do erro
- Resultam em um conjunto de testes
- Pode ser de execução complexa

FASES DO PROCESSO DE DEPURAÇÃO

Localização

- Seções do código que podem levar aos erros
- Parte mais complexa de se verificar
- Pode ser facilitada pela fase de estabilização

FASES DO PROCESSO DE DEPURAÇÃO

Correção

- Dependente das fases de Estabilização e Localização
- Correção aleatória pode gerar novos erros

FASES DO PROCESSO DE DEPURAÇÃO

Verificação

- Check sobre as correções realizadas
- Verifica se novos erros foram introduzidos durante a correção

TIPOS DE ERROS

Erros de sintaxe

- Problemas na escrita
- Utilização de recursos indevidamente

Erros de lógica

- Problemas na compreensão do problema
- Dificuldade técnica

FERRAMENTAS PARA DEPURAÇÃO

- Comparadores de código-fonte

Ex: WinMerge, KDiff3, Code Compare

- Verificadores Ampliados

Ex: Java Language Code, GCC

- Depuradores Interativos

Ex: Eclipse Debugger

FERRAMENTAS PARA DEPURAÇÃO

- Bibliotecas especiais

Ex: Eclipse Debugging Library

- Traçadores de perfil

Ex: Eclipse Profiler

ASSERÇÕES E PROGRAMAÇÃO DEFENSIVA

“uma técnica muito útil consiste na utilização de asserções, o que está relacionado aos conceitos mais formais de precondições e pós-condições”.

Tsui e Karam (2013, p.140)

ASSEGURAÇÕES E PROGRAMAÇÃO DEFENSIVA

Técnicas assertivas:

- Validações lógicas de uma condição que é julgada necessária para o correto funcionamento do código no momento em que elas são verificadas.
- Significa que uma condição assumida instrui o sistema a notificar sempre que essa condição não for satisfeita.

ASSERÇÕES E PROGRAMAÇÃO DEFENSIVA

Pré condição

- condição em que seu módulo solicita condições de produzir resultados corretos

Pós condição

- condição que deve se manter após a validação do seu código e que a precondição tenha sido atendida

ASSEGURAÇÕES E PROGRAMAÇÃO DEFENSIVA

- Instrui o código fonte a identificar falhas e comportamentos que não foram originalmente previstos e, assim, detectar bugs em seus estágios iniciais
- Validar as entradas de métodos/rotinas

OTIMIZAÇÃO DE DESEMPENHO

“o desempenho é um aspecto importante de praticamente qualquer programa, mas temos que entender as contrapartidas. A otimização para o desempenho geralmente (mas nem sempre) piora a mantinabilidade e a legibilidade”.

Tsui e Karam (2013, p.140)

OTIMIZAÇÃO DE DESEMPENHO

- Preocupação secundária à escrita correta e de fácil manutenção;
- Erro comum em programadores iniciantes;
- Ferramentas de apoio: Profiler.

REFATORAÇÃO

“mesmo quando se utiliza as melhores práticas e se faz um esforço consciente para produzir softwares de alta qualidade, é altamente improvável que se produza consistentemente programas que não possam ser melhorados”

Tsui e Karam (2013, p. 141)

REFATORAÇÃO

- Mudança de um código fonte, na estrutura interna do software visando melhorar o entendimento e a manutenibilidade sem alterar seu comportamento e suas funções externas.
- Visão sobre código já escrito - aprimoramento de seu estilo de código

REFATORAÇÃO – ELIMINAÇÃO DE CODE SMELL

- Código duplicado mostrando desperdício
- Método longo
- Classe grande
- Instruções switch
- Inveja da funcionalidade
- Intimidade inapropriada

REFATORAÇÃO – CUIDADO!

Pode apresentar riscos se aplicada da forma errada, tais como:

- atraso do projeto,
- introdução de falhas no sistema,
- tornar o código ilegível e não modificável

É uma mudança que deve ser efetuada com cuidado.

“Algumas pessoas pensam que Refatoração é apenas uma limpeza de código, mas ela vai, além disso, porque fornece técnicas específicas para cada tipo de alteração.”

Barrozo, Vinhas e Reis (2013)

BONS ESTUDOS