UNIVERSIDADE DE BRASÍLIA - UNB
Faculdade UnB Gama - FGA

Curso de Engenharia de Software

Trabalho 2 (TP2)

Técnicas de Programação em Plataformas Emergentes - Turma 01

Professor: André Luiz Peron Martins Lanna

Kess Jhones Gomes Tavares - 18/0124498

Pedro de Miranda Haick - 18/0129147

Rafael Berto Pereira - 18/0108344

Rodrigo Balbino Azevedo de Brito - 19/0048221

Victor Hugo Carvalho Silva - 19/0038969

Brasília, DF

2023

1. Simplicidade

Descrição:

A simplicidade é uma das características-chave de um código bem escrito. Envolve a criação de um código que seja de fácil compreensão, seguindo uma estrutura clara e consistente. Um código simples é aquele que é conciso, apresenta uma organização coerente e é facilmente legível, facilitando seu entendimento e manutenção.

Relação da característica com mau-cheiros:

Essa característica está estreitamente ligada aos "maus cheiros de código" conhecidos como *bloaters*, conforme definido por Martin Fowler. Esses *bloaters* são trechos de código que se tornaram excessivamente longos, geralmente devido à falta de uma delimitação clara de responsabilidades durante o planejamento da aplicação.

Em outras palavras, a simplicidade é um princípio essencial que visa evitar o crescimento desordenado do código-fonte. Ela envolve garantir que cada parte do código cumpra apenas uma responsabilidade específica e seja mantida em um tamanho gerenciável. Um código simples é aquele que evita a complexidade desnecessária, facilitando a leitura, a manutenção e a compreensão tanto para o desenvolvedor original quanto para outros membros da equipe.

Operação(ões) de refatoração capaz de levar o projeto de código a ter a característica em análise :

Uma operação de refatoração que contribui para alcançar essa característica é a "Extrair Método". Essa operação desempenha um papel importante ao simplificar o código, permitindo a criação de métodos mais curtos e coesos.

Ao aplicar a técnica de "Extrair Método", trechos de código repetitivos ou complexos podem ser isolados em métodos separados. Isso promove a reutilização de código e melhora a legibilidade, pois o código principal se torna mais enxuto e focado em sua

responsabilidade principal.

2. Elegância

Descrição:

A elegância engloba os aspectos estéticos do design e, frequentemente, está intimamente ligada à simplicidade. Em termos simples, ela significa que o código escrito não é confuso ou excessivamente complexo. Seguindo as ideias de Goodliffe, podemos identificar algumas características que contribuem para a elegância no código:

- Ausência de casos especiais: Um código elegante evita uma grande quantidade de condições especiais. Ele busca soluções mais generalizadas e simplificadas, evitando a complexidade desnecessária que decorre de considerar casos específicos separadamente.
- Complementaridade entre partes: Cada parte do código desempenha um papel valioso e distinto, contribuindo para o todo. Em vez de ter partes do código que são redundantes ou não adicionam valor significativo, o código elegante busca uma integração harmoniosa entre os diferentes componentes.
- Associação de coisas semelhantes: O código elegante busca agrupar e organizar elementos semelhantes, criando uma estrutura lógica e coerente. Isso melhora a legibilidade e a manutenibilidade, permitindo que os desenvolvedores identifiquem padrões e relacionamentos entre diferentes partes do código.
- Localidade de mudança: Um código elegante tem a característica de que uma única e simples alteração em um local específico não resulta em modificações complexas em várias partes do código. Isso torna a manutenção mais fácil, pois as alterações podem ser feitas de forma localizada, sem efeitos colaterais indesejados.

Relação da característica com mau-cheiros:

Essa característica se relaciona com o mau-cheiro inibidores de modificação, também conhecido com código espaguete. O código espaguete é aquele no qual, para alterar um ponto, precisamos também fazer alterações em diversos outros, o

que torna a manutenção uma dor de cabeça.

Operação(ões) de refatoração capaz de levar o projeto de código a ter a característica em análise :

Uma das operações que ajudam a atingir essa característica é a de Extrair Método e Parametrizar Método. Como já descrito anteriormente, ao extrair um método, você pode melhorar a legibilidade e a manutenibilidade do código, reduzindo a complexidade dos trechos de código e promovendo a reutilização.Já Parametrizar Método é uma técnica que envolve a identificação de valores fixos ou constantes dentro de um método e a substituição desses valores por parâmetros, tornando o método mais flexível e reutilizável.

Ao parametrizar um método, você permite que diferentes valores sejam passados como argumentos, adaptando o comportamento do método de acordo com as necessidades do contexto em que é utilizado. Isso promove a flexibilidade do código, evitando a repetição de lógica similar em vários lugares e reduzindo a complexidade.

3. Modularidade (baixo acoplamento e alta coesão)

Descrição:

Um código considerado modular é o código construído de forma distribuída, utilizando de módulos, interfaces e componentes para diminuir a complexidade de cada módulo utilizado, fazendo com que cada módulo seja mais fácil de entender, testar e refatorar de forma independente das outras partes do código.

A qualidade da aplicação dessa característica está ligada a coesão e acoplamento do código. Podemos identificar erros na aplicação desta característica de código em métodos que possuem várias funcionalidades, funcionalidades não relacionadas ou a dependência de muitos outros métodos.

Relação da característica com mau-cheiros:

Essa característica está associada ao mau-cheiro de código e inibidores de

modificações, também conhecidos como código espaguete. O código espaguete é aquele no qual, ao modificar um ponto específico, é necessário realizar alterações em várias outras partes, o que acaba dificultando a manutenção e tornando-se uma fonte de frustração.

Operação(ões) de refatoração capaz de levar o projeto de código a ter a característica em análise :

Uma das técnicas que contribui para alcançar essa característica é a operação de Extrair Método e Extrair Classe. Essa última técnica envolve identificar funcionalidades relacionadas dentro de uma classe existente e criar uma nova classe para encapsular essas responsabilidades.

Ao extrair uma classe, você divide as responsabilidades em módulos mais coesos e reduz o acoplamento entre as diferentes partes do código. Isso resulta em um código mais organizado, flexível e fácil de manter.

4. Ausência de duplicidades

Descrição:

A duplicação de código é um obstáculo para o design simples e elegante, pois gera redundâncias desnecessárias. Essas duplicações podem levar a problemas, como encontrar e corrigir um bug em uma parte do código, mas esquecer de corrigi-lo em outras partes, comprometendo a segurança do sistema.

Portanto, é essencial unificar trechos de código distintos que realizam tarefas semelhantes. Essa prática não só facilita a correção de problemas, mas também torna o código mais claro e legível. Ao eliminar a duplicação, os desenvolvedores podem focar em uma única implementação, garantindo que todas as partes do código estejam alinhadas e consistentes.

Relação da característica com mau-cheiros:

Essa característica está intimamente ligada ao mau-cheiro identificado por Fowler

como código duplicado. Ele define que, se um determinado trecho de código aparecer em várias partes do projeto, é melhor unificá-los para uma solução mais eficiente. Além disso, essa característica também se relaciona com o mau cheiro conhecido como Classe Grande, que também pode indicar a presença de código duplicado.

Operação(ões) de refatoração capaz de levar o projeto de código a ter a característica em análise :

Uma das práticas que contribui para alcançar essa característica é a realização das operações de Extrair Método e Extrair Constante. Quando há valores literais usados em várias partes do código, a extrapolação de uma constante pode ajudar a eliminar duplicações e melhorar a legibilidade e a manutenibilidade do código. A constante nomeada permite que o valor seja definido apenas uma vez e reutilizado em todo o código, evitando duplicações desnecessárias e facilitando possíveis modificações futuras.

5. Boa documentação

Descrição:

A presença de uma documentação adequada é essencial para permitir o rápido e qualificado desenvolvimento de um software. Uma documentação de qualidade vai além de apenas documentos separados, exigindo atenção na escrita do código em si, especialmente na escolha de nomes de variáveis, inclusão de comentários adequados e organização do fluxo de execução.

Relação da característica com mau-cheiros:

Essa característica está relacionada principalmente com os code smells de comentários excessivos e nomes misteriosos. Esses problemas surgem devido à falta de clareza durante o processo de desenvolvimento de software, especialmente quando várias equipes trabalham simultaneamente no mesmo projeto, aumentando

a complexidade da lógica e a padronização. Nesse contexto, uma documentação abrangente é fundamental para facilitar a manutenção e evitar possíveis problemas, como os mencionados anteriormente.

Operação(ões) de refatoração capaz de levar o projeto de código a ter a característica em análise:

Uma das práticas que contribui para alcançar essa característica é a realização das operações de Renomear Variável e Renomear Campo. Essas operações envolvem a alteração do nome de variáveis locais ou campos de objetos para nomes mais descritivos e significativos.

Ao renomear variáveis e campos, você melhora a legibilidade do código, facilitando a compreensão de sua finalidade e evitando ambiguidades. Nomes bem escolhidos tornam o código mais autoexplicativo e ajudam os desenvolvedores a entenderem a função e o propósito das variáveis e campos.

Referências Bibliográficas

***Code Craft: The Practice of Writing Excellent Code*, No Starch Press, Incorporated, 2006. *ProQuest Ebook Central*.**

***Refactoring: Improving the design of Existing Code*, Pearson Education, Incorporated, 2019.**