THE DEVELOPER'S CONFERENCE

Python – Sistemas legados, qualidade de código e bad smells

Gisele Zomer Rossi

THE DEVELOPER'S CONFERENCE

Gisele Zomer Rossi

Mestre em computação aplicada

Trabalho 10 anos com desenvolvimento

Professora Cesusc – Programação com python, programação web, Orientação a objetos e dispositivos móveis.

Construção e design de APIs utilizando Python e flask

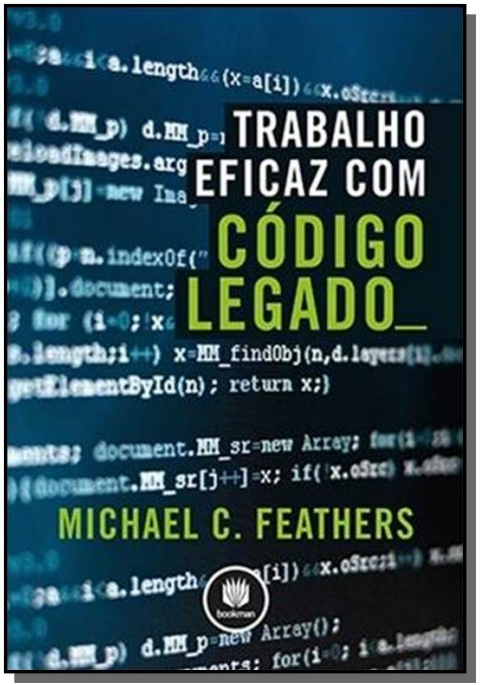
THE
DEVELOPER'S
CONFERENCE

Sistemas legados

```php
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```

THE DEVELOPER'S CONFERENCE

Código legado é código que recebemos de alguém, ou seja, é o código de outra pessoa.



THE DEVELOPER'S CONFERENCE

É sobre transformar sistemas que se deterioram gradualmente em sistemas que melhoram gradualmente!

THE
DEVELOPER'S
CONFERENCE

Refatoração

THE DEVELOPER'S CONFERENCE

- Refatorar é melhorar o código depois de ele ter sido escrito

- "Refatoramento é o processo de mudança de um sistema de software de tal maneira que não altere o comportamento externo do código e ainda melhore a sua estrutura interna"

Bad smells

- *Bad smells* são possíveis falhas existentes no código que podem ser melhoradas por meio de refatorações.

- Cada *Bad smells* examina um tipo específico de elemento do sistema, como, por exemplo, classes ou métodos.

THE
DEVELOPER'S
CONFERENCE

Código duplicado

THE DEVELOPER'S CONFERENCE

- Código duplicado quebra um dos princípios da orientação a objetos que é reutilização de código
- Programadores duplicam código para mostrar "trabalho" (métrica de commit), e por dificuldade de entender o código já existente
- Corrigir bugs em dois lugares
- Duplicar testes

THE
DEVELOPER'S
CONFERENCE

ACOPLAMENTO E COESÃO

- Classes e métodos devem ter baixo acoplamento e alta coesão
- Classes ou método grandes podem indicar baixa coesão

```python
class Fornecedor:

    def salvar(self):
        pass

    def deletar(self):
        pass

    def enviar_email(self):
        pass

    def validar_cliente(self):
        pass
```

THE
DEVELOPER'S
CONFERENCE

Use nomes que revelem o seu propósito

THE DEVELOPER'S CONFERENCE

- Se um nome necessita de um comentário, então ele não revela o seu propósito

- Evitar que nome de variáveis contenham a palavra variável, assim como, evitar que o nome de uma tabela tenha a palavra tabela/table

- **A diferença do programador amador e do profissional é que o profissional entende que clareza é fundamental e não tenta ficar se exibindo com códigos super difíceis**

- Nome de métodos devem ser verbos:

  postar_pagamento
  excluir_pagina
  salvar_pessoa

THE DEVELOPER'S CONFERENCE

- Definir um padrão de nomenclatura

```python
class Cliente:

    def obter_cliente(self):
        pass

class Fornecedor:

    def recuperar_fornecedor(self):
        pass

class Aluno:

    def pegar_aluno(self):
        pass
```

```python
class Cliente:

    def obter_cliente(self):
        pass

class Fornecedor:

    def obter_fornecedor(self):
        pass

class Aluno:

    def obter_aluno(self):
        pass
```

Utilize constantes

THE DEVELOPER'S CONFERENCE

- O que é o 45?!
- É uma lista e que?!
- O que tem na posição 2?!

```
if lista[2] / 45:
    #faz algo
```

THE
DEVELOPER'S
CONFERENCE

Comentários

THE DEVELOPER'S CONFERENCE

```
// Dear future me. Please forgive me.
// I can't even begin to express how sorry I am.


// John! If you'll svn remove this once more,
// I'll shut you, for God's sake!
// That piece of code is not "something strange"!
// That is THE AUTH VALIDATION.


// I have to find a better job


// If you're reading this, that means you have been
// put in charge of my previous project.
// I am so, so sorry for you. God speed.


// This is crap code but it's 3 a.m. and I need to get this working.
```

```
// When I wrote this, only God and I understood what I was doing
// Now, God only knows


// I am not sure if we need this, but too scared to delete.


// Dear maintainer:
//
// Once you are done trying to 'optimize' this routine,
// and have realized what a terrible mistake that was,
// please increment the following counter as a warning
// to the next guy:
//
// total_hours_wasted_here = 42


// somedev1 -  6/7/02 Adding temporary tracking of Login screen
// somedev2 -  5/22/07 Temporary my ass


// Magic. Do not touch.
```

THE DEVELOPER'S CONFERENCE

THE
DEVELOPER'S
CONFERENCE

•Não insira comentários em código ruim, **reescreva-o**

•Desejamos colocar comentário pois nem sempre encontramos uma forma de nos expressar sem eles

THE DEVELOPER'S CONFERENCE

- Quando estiver em uma situação que seja colocar comentários no seu código, pense se isso não pode ser transcrito no código
- Comentários errados ou desatualizados são piores que nenhum comentário

THE
DEVELOPER'S
CONFERENCE

Métodos/Funções

THE DEVELOPER'S CONFERENCE

- O programa deve ser lido como se fosse uma série de parágrafos, cada um descrevendo o nível atual e fazendo referência aos parágrafos consecutivos
- Métodos devem ser pequenos
- Os métodos devem fazer apenas uma coisa (coesão)

THE DEVELOPER'S CONFERENCE

O código deve ser lido de cima para baixo com uma narrativa

```
def enviar_nota_fiscal(nota_fiscal):
    validar_campos_obrigatorios(nota_fiscal)
    inserir_nota_fiscal(nota_fiscal)
    enviar_nota_fiscal_receita(nota_fiscal)
    enviar_email_cliente(nota_fiscal)
```

Blocos dentro de if/else/while devem ter apenas uma linha, uma chamada a um método (método com um nome significativo)

```
def enviar_nota_fiscal(nota_fiscal):
    if nota_fiscal:
        inserir_nota_fiscal(nota_fiscal)
    else:
        inserir_cliente()
```

THE DEVELOPER'S CONFERENCE

- Princípio de responsabilidade única
- Antigamente se falava que as funções/métodos devem caber no monitor
- O ideal é ter 4 ou 5 linhas.

THE DEVELOPER'S CONFERENCE

Parâmetros de métodos

```
def calcular_imposto(n1,n2,n3,n4,n5,n6,n7, n8, n9, n10):
    pass
```

Recomendação de utilizar no máximo 3 parâmetros

- Evite passar booleano por parâmetro
- Dificulta o entendimento e mostra que o método faz mais de uma coisa
- Prefira criar dois métodos ao invés de um que receba true e false

```
def cacular_media_aluno(nota_1, nota_2, aluno_regular):
    if aluno_regular:
        faz_uma_coisa(nota_1, nota_2)
    else:
        faz_outra_coisa(nota_1, nota_2)
```

THE DEVELOPER'S CONFERENCE

- Evite muitas condições no mesmo if
- Refatore essas condições para um método

```
def calcular_bonificacao_funcionario(n1, n2, n3):
    if n1 > 49 and n2 == 50 and n3 == 45:
        print("essa validacção pode ser melhorada")

    if possui_beneficio(n1) and possui_participacao_lucros(n2, n3):
        print("validação ficou mais clara")
```

THE DEVELOPER'S CONFERENCE

- Coloque os blocos que estiverem dentro do try em um outro método
- Os métodos devem fazer apenas uma coisa e tratar o erro é só uma coisa

```
try:
    self.faz_alguma_coisa()
except:
    self.trata_excecao()
```

THE
DEVELOPER'S
CONFERENCE

Formatação

THE DEVELOPER'S CONFERENCE

- Deve-se declarar as variáveis o mais próximo possível de onde serão usadas
- Variáveis de instância devem ser declaradas no início da classe
- Métodos dependentes devem ficar próximos. De preferência quem estiver chamando deve ficam em cima.

THE
DEVELOPER'S
CONFERENCE

```
pessoa = self.obter_pessoa()
    if pessoa:
        self.inserir_pessoa()

def obter_pessoa(self):
    #obtem pessoa

def inserir_pessoa(self):
    #insere pessoa
```

THE
DEVELOPER'S
CONFERENCE

Testes

THE DEVELOPER'S CONFERENCE

- **Quanto maior a cobertura de testes menor o medo**
- Os testes não devem ter qualidade menor do que o código de produção, eles devem evoluir junto
- Um método de teste deve ter um único assert, mostrando que o testes faz uma única coisa (mesma lógica nos métodos)

THE DEVELOPER'S CONFERENCE

- Como testar esse código?
- Quantos testes serão necessários para validar todas as condições?

```
for i in ist:
    for r in ist[i]:
        for s in ist[i][r]:
            for ni in new:
                for ni_r in ni['r']:
                    if r == ni_r['r_id']:
                        for y in ist[i][r]:
                            if s == y:
                                ni_r['s'].append(ist[i][r][y])

                            ni['r_date'] = ni_r['ss'][0]['r_date']
```

THE DEVELOPER'S CONFERENCE

Métricas de qualidade de código

THE
DEVELOPER'S
CONFERENCE

Por meio delas é possível chegar a maior compreensão do código fonte, proporcionando, também, a avaliação da qualidade e a evolução das melhorias do código que está sendo produzido

THE
DEVELOPER'S
CONFERENCE

Complexidade ciclomática

THE DEVELOPER'S CONFERENCE

- Indica a complexidade de trecho de código
- O resultado da complexidade ciclomática indica quantos testes precisam ser executados para que se verifique todos os fluxos possíveis que o código pode tomar, a fim de garantir uma completa cobertura de testes.

THE DEVELOPER'S CONFERENCE

Quanto custa entender esse código?!

```
for i in lista_1:
    for r in lista2[i]:
        for s in lista3[i][r]:
            for ni in nova_lista:
                for ni_r in ni['obj']:
                    if r == ni_r['obj_id']:
                        for y in inst[i][r]:
                            if s == y:
                                ni_r['date'] = inst[i][r][y]['date']
                                ni_r['ex'] = inst[i][r][y]['ex']
                                ni_r['o'].append(inst[i][r][y])
```

THE DEVELOPER'S CONFERENCE

Métodos ponderados por classe

THE DEVELOPER'S CONFERENCE

- Esta é uma métrica de complexidade de código que indica que uma classe não pode ter muitos métodos

- Geralmente, se uma classe possui muito métodos ela não é coesa, ou seja, não faz uma única coisa

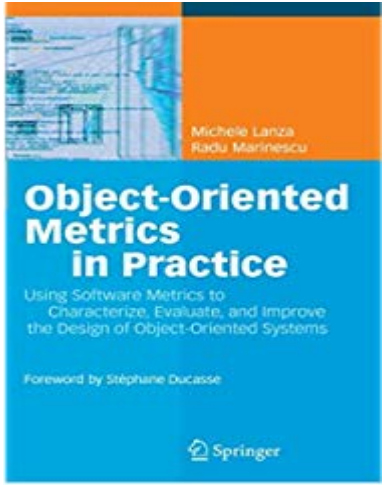
THE DEVELOPER'S CONFERENCE

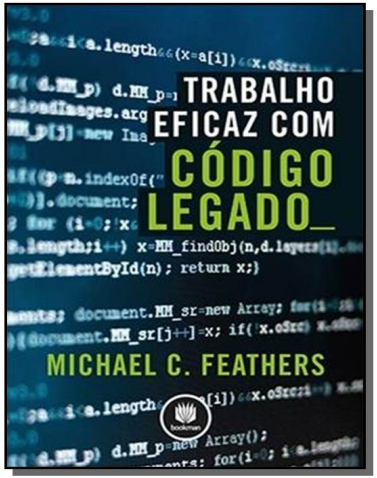
Profundidade na árvore de herança

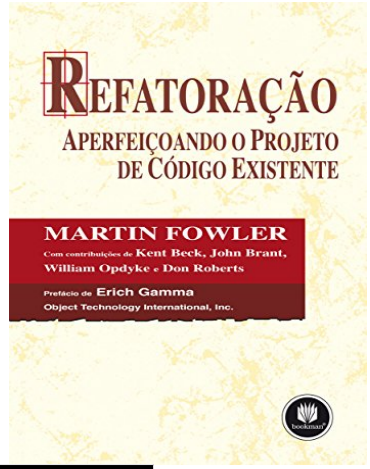
THE DEVELOPER'S CONFERENCE

- Uma classe que possui muitos filhos ou mais que três pais, gera um alto nível de acoplamento no código, além de aumentar drasticamente a complexidade

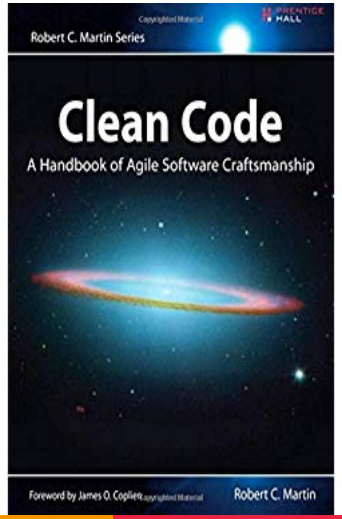
- Deste caso, o programador deve, conhecer os métodos da classe atual, e os métodos de todas as classes pai.

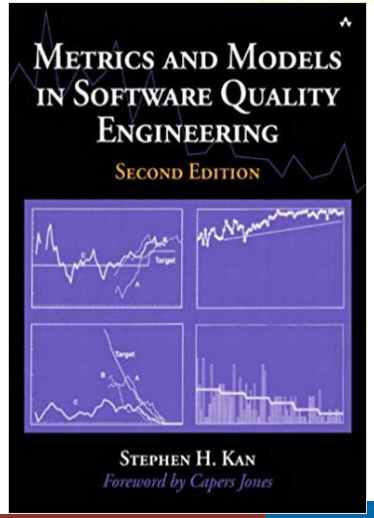


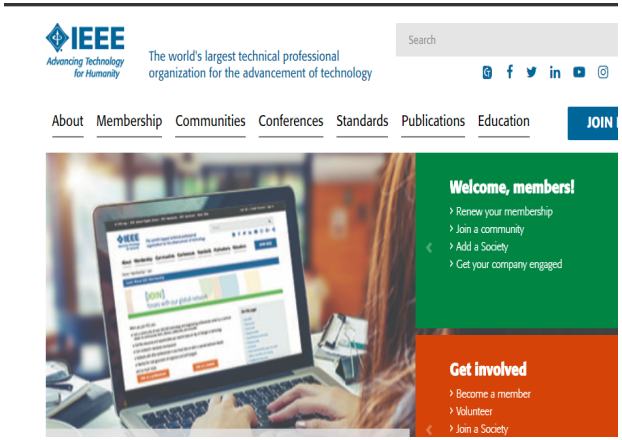




THE DEVELOPER'S CONFERENCE







THE DEVELOPER'S CONFERENCE

Obrigada!

https://www.linkedin.com/in/gisele-zomer-rossi