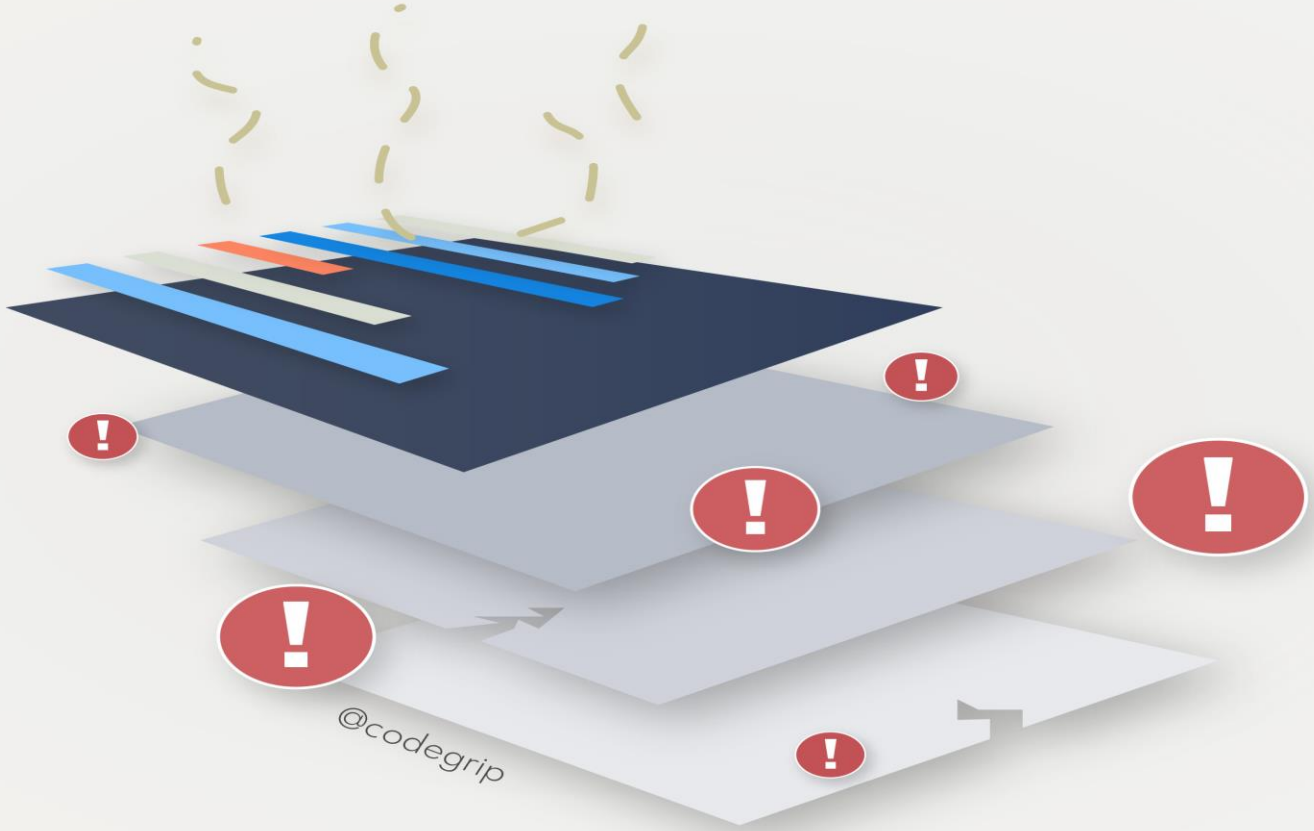


CodeSmell

Prof. Edson Mota, PhD, MSc, PMP

O que é Code Smell?

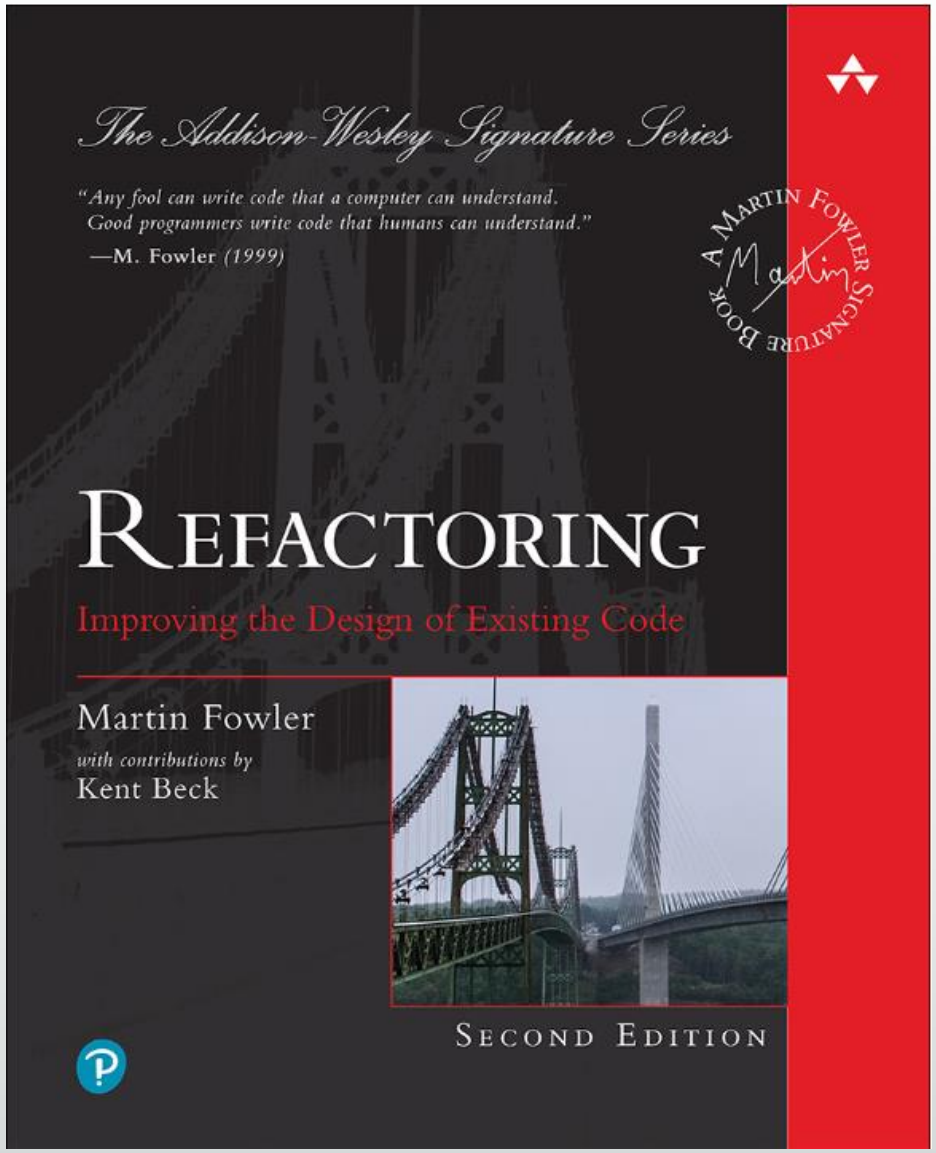
Pode ser entendido como um **sintoma** que ocorre dentro do código e que pode ser um **indicador** de **problemas** e futuros bugs

Outra definição

[…] Certain structures in the code that suggest (sometimes they scream for) the possibility of refactoring […]

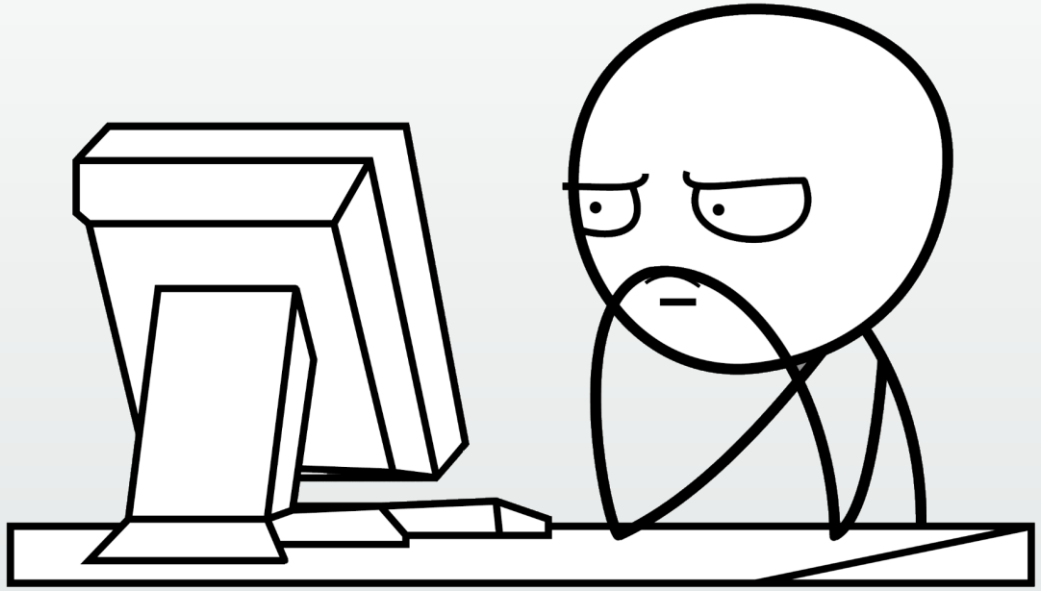
[…] Certas estruturas no código que sugerem (às vezes gritam) a possibilidade de refatoração […]

(Martin Fowler, 2018)



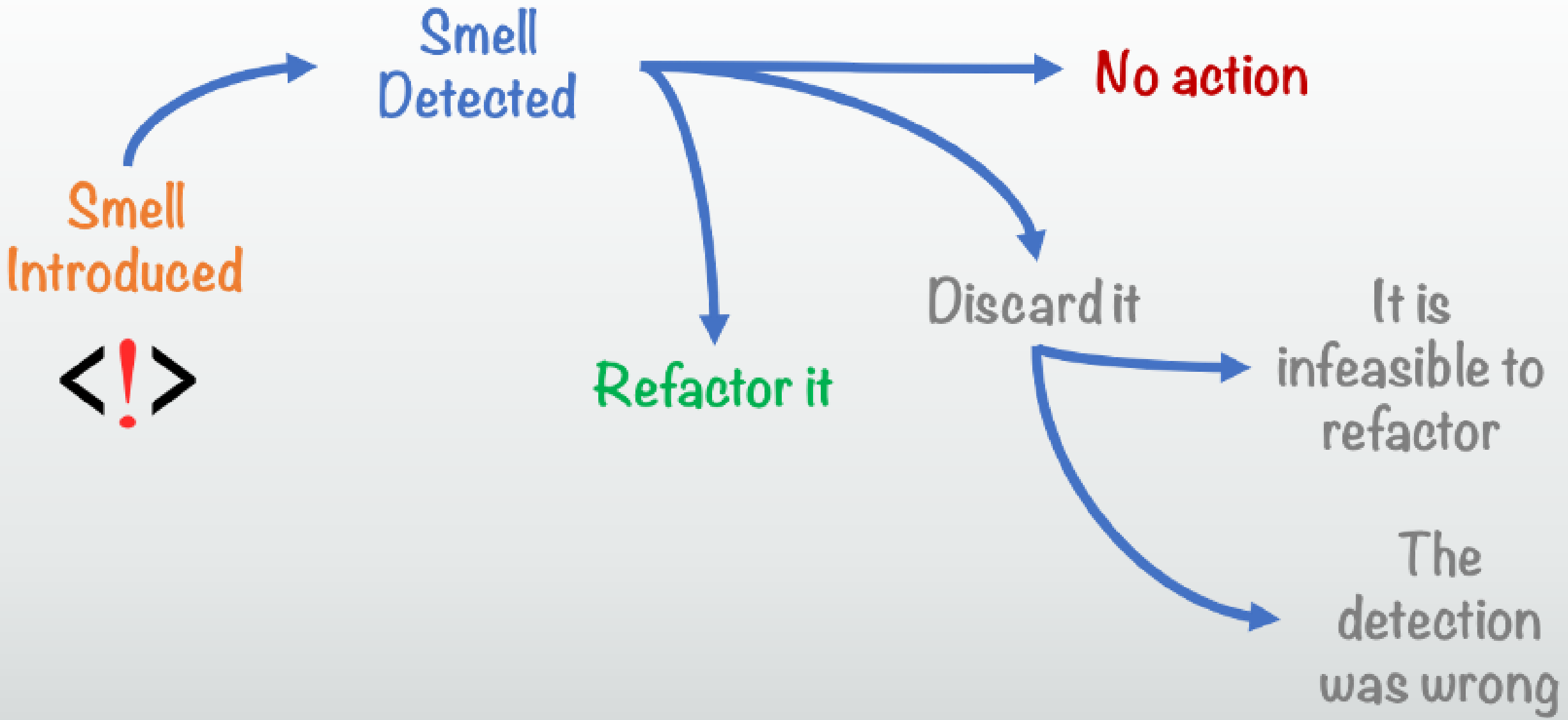
Refatorar

- A ação de tornar o seu código
  - Mais fácil de entender
  - Mais barato para modificar



- Tudo isso, sem impactar no comportamento

Qual a relação entre CodeSmell x Defeitos?


Smell Introduced
<!>
Smell Detected
No action
Refactor it
Discard it
It is infeasible to refactor
The detection was wrong

Code Smells mais comuns..

- Duplicação de código
- Métodos muito longos
- Classes muito grandes
- Longas listas de parâmetros
- Mensagens encadeadas
- Inveja de recursos
- Nomes inapropriados
- Classe de dados
- Declarações switch
- Generalidade especulativa
- Campos temporários
- Herança rejeitada
- Muitos comentários
- Código morto

e muitos outros....

Refatorando!

Analise esse código!

- Existem **muitos comentários**, mas a maior parte deles não contribuir para o melhor entendimento do código.
  - Isso é um **forte indício** de que o código está muito **complexo** e que precisa de **refatoração**.

```
0 referências
public class Calculo_Fatorial {
    // Função que calcula o fatorial de um número
    0 referências
    public int Fatorial(int n)
    {
        /*
        Esta função calcula o fatorial de um número.
        É importante lembrar que o fatorial só pode ser calculado
        para números inteiros positivos.
        */

        // Verifica se o número é inteiro e positivo
        if (n < 0 || !int.TryParse(n.ToString(), out _))
        {
            // Se não for, retorna uma mensagem de erro
            return -1;
        }

        // Se o número for igual a zero ou um, retorna um
        if (n == 0 || n == 1)
        {
            return 1;
        }

        // Inicializa o resultado com 1
        int resultado = 1;

        // Calcula o fatorial
        for (int i = 2; i <= n; i++)
        {
            resultado *= i;
        }

        return resultado;
    }
}
```

Algum problema aqui?

```
namespace CodeSmell
{
    0 referências
    internal class Pedido
    {
        0 referências
        public void EnviarPedido(string nomeCliente, string enderecoCliente,
            string cidadeCliente, string estadoCliente, string paisCliente,
            string emailCliente, string telefoneCliente, string nomeProduto,
            int quantidadeProduto, decimal precoProduto, string codigoProduto)
        {
            // Regras de negócio para enviar pedido
        }
    }
}
```

```
1 referência
internal class Pedido
{
    0 referências
    public string NomeCliente { get; set; }
    0 referências
    public string EnderecoCliente { get; set; }
    0 referências
    public string CidadeCliente { get; set; }
    0 referências
    public string EstadoCliente { get; set; }
    0 referências
    public string PaisCliente { get; set; }
    0 referências
    public string EmailCliente { get; set; }
    0 referências
    public string TelefoneCliente { get; set; }
    0 referências
    public string NomeProduto { get; set; }
    0 referências
    public int QuantidadeProduto { get; set; }
    0 referências
    public decimal PrecoProduto { get; set; }
    0 referências
    public string CodigoProduto { get; set; }


    0 referências
    public void EnviarPedido(Pedido pedido)
    {
        // Faz alguma coisa com as informações do pedido
    }
}
```

Código Refatorado

Algum problema com esse código?

```
namespace CodeSmell
{
    0 referências
    public class xyz
    {
        0 referências
        public int x { get; set; }
        0 referências
        public int y { get; set; }

        0 referências
        public void mn()
        {
            int a = 1;
            int b = 2;

            int c = a + b;

            Console.WriteLine(c);
        }
    }
}
```

Nomes inapropriados para variáveis, classes e métodos.

```
0 referências
public class Pedido
{
    0 referências
    public int Id { get; set; }
    0 referências
    public string Descricao { get; set; }
    0 referências
    public DateTime DataPedido { get; set; }
    0 referências
    public List<ItemPedido> Itens { get; set; }
    // Propriedades de informações de pagamento
    0 referências
    public string NomeCliente { get; set; }
    0 referências
    public string EnderecoCliente { get; set; }
    0 referências
    public string TelefoneCliente { get; set; }
    0 referências
    public string NumeroCartaoCredito { get; set; }
    0 referências
    public string CodigoSegurancaCartaoCredito { get; set; }
    0 referências
    public DateTime DataVencimentoCartaoCredito { get; set; }
    0 referências
    public decimal ValorTotal { get; set; }

    // Métodos para processamento de pagamento
    0 referências
    public void ValidarInformacoesPagamento()
    {
        // Código para validar informações de pagamento
    }
    0 referências
    public void ProcessarPagamento()
    {
        // Código para processar o pagamento
    }

    // Métodos de manipulação de Itens do Pedido
    0 referências
    public void AdicionarItem(ItemPedido item)
    {
        // Código para adicionar item no pedido
    }

    0 referências
    public void RemoverItem(ItemPedido item)
    {
        // Código para remover item do pedido
    }

    // Métodos de obtenção de informações do Pedido
    0 referências
    public decimal ObterValorItens()
    {
        // Código para obter o valor total dos itens
        return 0;
    }

    0 referências
    public decimal ObterValorDesconto()
    {
        // Código para obter o valor total dos descontos
        return 0;
    }

    0 referências
    public decimal ObterValorTotal()
    {
        // Código para obter o valor total do pedido
        return 0;
    }
}
```

Baixa coesão

Bons estudos!