12 - Emergence

2025-10-03 17:08 pm

Queremos regras que naturalmente gerem boas diretrizes de design, como **SRP** e **DIP**

Regras de um bom design simples:

1. **Executa todos os teste**
2. **Não tem duplicações**
3. **Expressa a intenção do programador**
4. **Minimiza o número de classes e métodos**

1 - Executa todos os testes

- Design deve permitir sistema completamente testável
- Incentiva classes pequenas e de responsabilidade única (SRP) - facilita teste.
- Acoplamento excessivo dificulta testes - incentiva DIP e outros.
- Com os testes podemos pensar em refatoração e melhorias, como mudar nomes, diminuir acomplamento, etc. Regra 1 permite as regras 2 a 4.

2 - Não tem duplicações

- Duplicação é trabalho e risco a mais desnecessários

```
#Métodos diferentes similares
int size() {}
boolean isEmpty() {}

# Eliminamos uma duplicação juntando uma à outra
boolean isEmpty() {
return 0 == size()
}
```

- Template Method pattern é uma alternativa para evitar duplicações

3 - Expressivo

- Maior custo de software é manutenção
- Quanto mais claro, menos tempo para entender.
- Nomes claros, funções pequenas, usar padrões de nomenclatura e de design (e.g. Visitor).
- Bons testes também são expressivos
- Não basta ter o código só funcionando, tem que estar claro - inclusive para nós mesmos no futuro

4 - Classes e Métodos Mínimos

- SRP pode induzir excesso de pequenas classes - isso é ruim
- Importante evitar dogmas
- Esta é a regra menos importante das quatro.