Livro Engenharia de Software Moderna

Capítulo 9

A refatoração é um processo essencial no desenvolvimento de software, que consiste na melhoria do código sem alterar seu comportamento externo. A refatoração surge como uma estratégia para evitar o declínio do sistema, promovendo um código mais organizado, legível e fácil de modificar.

Diversas técnicas de refatoração são utilizadas no desenvolvimento de software. Uma das mais comuns é a Extração de Método, que consiste em isolar um trecho de código dentro de um novo método e substituí-lo por uma chamada a esse novo método. Outra técnica é o Inline de Método, que funciona de maneira inversa, incorporando o corpo de um método pequeno diretamente nos pontos onde ele é chamado, reduzindo a fragmentação desnecessária. Outro método é a Movimentação de Método, que transfere um método de uma classe para outra quando ele está mais relacionado com os dados e funcionalidades dessa nova classe, melhorando a coesão e reduzindo o acoplamento.

A Extração de Classe é outro refactoring, utilizado quando uma classe acumula muitas responsabilidades. Nesse caso, alguns de seus atributos e métodos são extraídos para uma nova classe. A renomeação de métodos, variáveis e classes é uma prática para garantir que os nomes dos elementos do código reflitam claramente suas responsabilidades para facilitar a compreensão e manutenção.

A prática de refatoração é fortemente apoiada por ferramentas automatizadas disponíveis em IDEs modernas, como IntelliJ, Eclipse e Visual Studio. Essas ferramentas permitem que desenvolvedores realizem refatorações com segurança, verificando as pré-condições necessárias para que a modificação não cause erros no código.

Além das técnicas de refatoração, existe os code smells, que são sinais de que um código pode estar mal estruturado e precisa ser melhorado. Um dos principais code smells é o Código Duplicado, que aumenta o esforço de manutenção, pois qualquer modificação precisa ser replicada em várias partes do código. Para eliminar essa redundância, pode-se utilizar a Extração de Método ou a Extração de Classe.

Outro problema é o de Classes Grandes, que acumulam muitas responsabilidades, tornando-se difíceis de modificar e testar. Para resolver essa questão, pode-se dividir a classe em partes menores, cada uma com uma responsabilidade bem definida.

A imutabilidade também é um conceito importante para reduzir problemas de concorrência e garantir previsibilidade no comportamento do código. Objetos Mutáveis podem causar efeitos colaterais indesejados, enquanto objetos imutáveis garantem maior segurança, pois seu estado não pode ser alterado após a criação.

Um aspecto controverso é a presença excessiva de Comentários no código. Embora comentários sejam úteis em determinadas situações, muitas vezes são usados para

explicar código mal escrito. Em vez disso, uma abordagem melhor é refatorar o código para que ele seja autoexplicativo, eliminando a necessidade de comentários excessivos.

A refatoração está intimamente ligada ao conceito de dívida técnica. Se um sistema acumula problemas como falta de testes, estrutura desorganizada e presença de code smells, seu desenvolvimento se torna cada vez mais lento e caro.