Teste de Software – Projeto de Testes com xUnit

Alunos: Juliana Parreiras, Pedro Marques, Gabriel Henrique, Amanda Bicalho
Disciplina: Teste de Software – PUC MG

O xUnit é um framework de testes unitários para aplicações .NET, desenvolvido como uma evolução dos frameworks NUnit e MSTest. Ele surgiu com o objetivo de oferecer uma abordagem mais moderna, leve e extensível para testes automatizados, permitindo que desenvolvedores criem, executem e organizem testes de maneira eficiente. O framework adota conceitos de atributos `[Fact]` e `[Theory]` para identificação de métodos de teste, possibilitando a execução automatizada, a parametrização de casos e a integração com ferramentas de Continuous Integration (CI). Além disso, o xUnit segue padrões que incentivam boas práticas de desenvolvimento, como Test Driven Development (TDD), garantindo que testes sejam criados paralelamente ao código-fonte e promovendo uma cultura de qualidade desde o início do desenvolvimento.

O funcionamento do xUnit se dá pela identificação automática de métodos de teste através de atributos específicos. O `[Fact]` é utilizado para testes unitários simples, que não dependem de parâmetros, enquanto o `[Theory]` permite a execução de testes parametrizados com múltiplos conjuntos de dados. Durante a execução, o framework realiza asserções sobre os resultados obtidos, utilizando métodos como `Assert.Equal`, `Assert.True` e `Assert.Throws`, entre outros. Além disso, o xUnit suporta configuração e teardown de testes através de construtores e interfaces como `IClassFixture`, garantindo que cada teste seja executado em um ambiente isolado e consistente. A instalação e integração do framework são realizadas de forma simples por meio do .NET CLI, criando projetos específicos de teste e adicionando referência ao projeto principal, possibilitando a execução dos testes via comando `dotnet test`.

O projeto desenvolvido pelos alunos consiste em uma calculadora em C# denominada **ConsoleApp1**, implementando quatro operações básicas: adição, subtração, multiplicação e divisão. Para cada operação, foram criados testes unitários que validam a funcionalidade de cada método de forma isolada. Os testes contemplam casos com valores positivos, negativos e zero, além de valores decimais ou fracionários. Foram também incluídos casos de overflow utilizando `double.MaxValue` e casos de underflow com `double.Epsilon`. No caso da divisão, foram tratados cenários que resultam em exceções, como a divisão por zero, garantindo que uma `DivideByZeroException` seja lançada corretamente. Cada teste unitário valida o comportamento esperado de cada operação, assegurando que os métodos funcionem corretamente de forma independente.

Além dos testes unitários, foram implementados testes de integração que verificam a interação entre múltiplos métodos da calculadora. Um exemplo de cenário testado foi a sequência de operações `(Add → Multiply → Divide → Subtract)`, garantindo que o

resultado final esteja correto quando várias operações são combinadas. Em um dos casos, `(2 + 3) * 4 / 2 - 1`, o resultado final esperado foi 9, validando que cada operação intermediária produz o resultado correto e que a sequência completa de cálculos funciona conforme o esperado. Esses testes de integração são fundamentais para assegurar que o sistema se comporte corretamente quando os métodos interagem entre si, aumentando a confiabilidade do projeto.

A derivação dos casos de teste foi baseada em uma análise criteriosa das operações da calculadora. Foram considerados valores válidos e comuns, como números positivos, negativos e zero, bem como valores extremos, incluindo `double.MaxValue`, `double.MinValue` e `double.Epsilon`. Casos de exceção, como a divisão por zero, também foram contemplados, assim como situações envolvendo comportamento especial de ponto flutuante, como NaN e Infinity. Para os testes de integração, foram elaboradas sequências de operações que combinam múltiplos métodos, garantindo a validação de cenários compostos e o correto funcionamento do sistema como um todo.

O uso do xUnit neste projeto proporcionou uma validação confiável das operações matemáticas da calculadora, garantindo que futuras alterações não comprometam funcionalidades existentes. A combinação de testes unitários e testes de integração permitiu verificar tanto a correção individual de cada operação quanto a interação entre elas em sequências de cálculo. Além disso, os testes documentam o comportamento esperado de cada método, servindo como referência para futuras modificações e como evidência de qualidade. Todo o projeto está organizado com a aplicação principal no **ConsoleApp1** e o projeto de testes em **CalculatorTests**, podendo ser executado via `dotnet test` ou integrado a pipelines de CI/CD.

Em conclusão, este trabalho evidencia a importância de se utilizar frameworks de teste como o xUnit para garantir a confiabilidade e a robustez de sistemas de software. A implementação de testes unitários e de integração na calculadora demonstra como é possível verificar tanto funcionalidades isoladas quanto a interação entre métodos, promovendo a qualidade do código e oferecendo segurança para alterações futuras. A prática consolidou conceitos da disciplina de Teste de Software, proporcionando experiência real no desenvolvimento e execução de testes automatizados em projetos .NET.