Ambientes de Desenvolvimento Avançados

http://www.dei.isep.ipp.pt/~jtavares/ADAV/ADAV.htm

Aula Teórico-Prática

Engenharia Informática

2005/2006

José António Tavares
jrt@isep.ipp.pt

Test Driven Development – Part II

Current status	Stable
Created date	01/26/2004
Audience	Beginner, Intermediate, Advanced
License	View license
Language	C#
Technology	ASP.NET, Web services
Access	Public

http://workspaces.gotdotnet.com/tdd

Um ficheiro com o capítulo 2 do livro estará disponível na página da disciplina

O que é o TDD?

- Técnica de desenvolvimento
- Teste antes do código!
- Testes unitários
  - Testam funcionalidade de módulo/classe em isolamento
- Testes pelo/para Programador
  - Mesma linguagem/Mesmo IDE

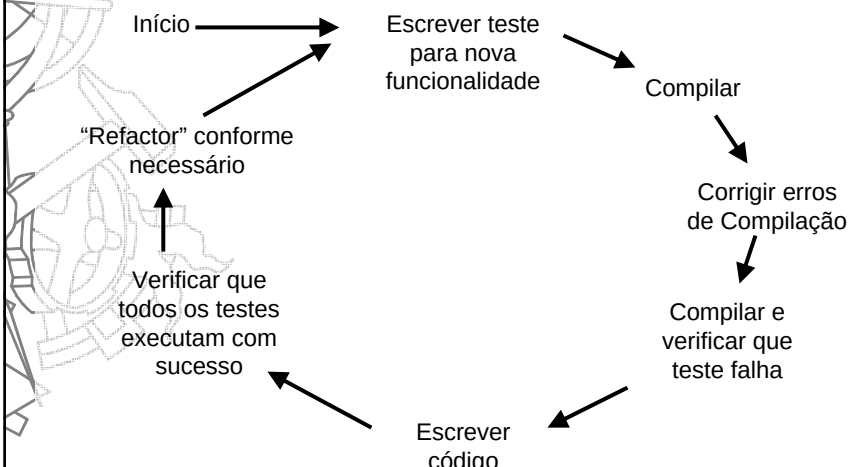
Motivação para o TDD

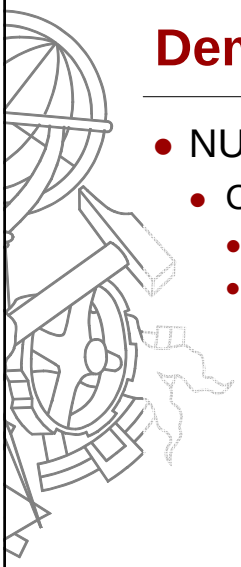
- Melhorar qualidade do código produzido
- Diminuir defeitos
- Diminuir risco
- Potenciar/Encorajar a Mudança!

Origem do TDD

- Uma das práticas principais do eXtreme Programming (XP) – Kent Beck, Martin Fowler – chamadas “Metodologias Ágeis”
- Exploração do potencial de frameworks OO actuais (Java, .Net,…)
- Testes Unitários

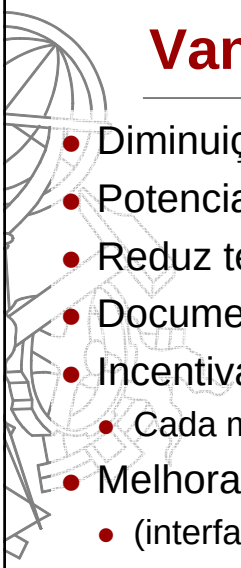
Processo TDD





Demo!

- NUnit 2.2 & TDD
  - Calculadora
    - TestFixtureSetup/TestFixtureTearDown
    - Setup/TearDown



Vantagens

- Diminuição de erros e defeitos
- Potencia Refactoring agressivo
- Reduz tempos de Debugging
- Documentação “Executável”
- Incentiva à Reusabilidade
  - Cada módulo tem pelo menos 2 clientes
- Melhora skills de arquitectura e desenho
  - (interface based design, padrões)

Vantagens (cont.)

- Separação clara de competências entre módulos e classes
- Programação por “intenção”
- Medição efectiva de progresso do projecto
- Maior confiança na evolução do produto
- Adesão cada vez maior (mundo Open Source)
- Vantagem competitiva para a empresa e para o programador!

Vantagens (cont.)

- Ciclos desenvolvimento menores/esforço mais nivelado
- Bugs ficam claramente registados – testes!
- Adaptação a ambientes de integração contínua
- Conceito multi-plataforma

Best Practices

- Teste em 1º Lugar!
- Cobertura de código tão extensa quanto possível
- Testar tudo o que possa falhar
- Teste de classes em isolamento
  - Testes imunes a influências externas
- Aplicação de Model-View-Controller para teste de Interfaces

Refactoring

- É definido como o melhoramento do código e ao mesmo tempo não mudando a sua funcionalidade;
- É uma tarefa critica no TDD ...

- exemplos:
  - remoção de código duplicado;
  - código mais legível;
  - ...

Red/Green/Refactor

- Define o processo de implementação de cada teste na lista de testes. O objectivo é "*Work in Small*", passos verificaveis que proporcionam *feedback* imediato.
  1. Escrever código de teste
  2. Compilar o código de teste (deve falhar – nada implementado)
  3. Implementar o necessário para compilar
  4. Correr o teste e observar que falha
  5. Implementar apenas o necessário para passar o teste
  6. Correr o teste e observar o sucesso
  7. Refactor para eliminar duplicação
  8. Repetir tudo ...

Demo!

Test-Driven Development in .NET—By Example

- we'll demonstrate how to implement a *Stack* using Test-Driven Development (TDD).

Demo: Stack Test List

- Create a *Stack* and verify that *IsEmpty* is true.
- *Push* a single object on the *Stack* and verify that *IsEmpty* is false.
- *Push* a single object, *Pop* the object, and verify that *IsEmpty* is true.
- *Push* a single object, remembering what it is; *Pop* the object, and verify that the two objects are equal.
- *Push* three objects, remembering what they are; *Pop* each one, and verify that they are removed in the correct order.
- *Pop* a *Stack* that has no elements.
- *Push* a single object and then call *Top*. Verify that *IsEmpty* is false.
- *Push* a single object, remembering what it is; and then call *Top*. Verify that the object that is returned is the same as the one that was pushed.
- Call *Top* on a *Stack* with no elements.

Demo: Stack Test List

- As we were implementing this test, a few additional tests came to mind, so we need to add them to our *test list*. We want to add tests to verify that the *Stack* works when the arguments are equal to *null*.
- The new tests are as follows:
  - *Push null* onto the *Stack* and verify that *IsEmpty* returns false.
  - *Push null* onto the *Stack*, *Pop* the *Stack*, and verify that the value returned is *null*.
  - *Push null* onto the *Stack*, call *Top*, and verify that the value returned is *null*.

Desafios

- Testes de BD
- Testes de Interfaces Gráficos
- Testes de Aplicações Web
- Testes em Isolamento
  - Mock frameworks
- Grandes Aplicações

Referências

- Kent Beck, "Test Driven by Example“
- Martin Fowler, "Refactoring: Improving the design of existing code"
- James W. Newkirk & Alexei A. Vorontsov,"Test-Driven Development in Microsoft.Net"
- Ronald E. Jeffries, "Extreme Programming Adventures in C#"

Links

James Newkirk
blogs.msdn.com/jamesnewkirk
Brian Marick
www.testing.com/cgi-bin/blog
Jonathan de Halleux
http://blog.dotnetwiki.org/
Jose Almeida
blogs.msdn.com/josealmeida/
Scott Densmore –
blogs.msdn.com/scottdensmore
Code Project
www.codeproject.com/dotnet/tdd_in_dotnet.asp
TestDriven.com
www.testdriven.com

Questões

?