



INSTITUTO DE  
INFORMÁTICA  
UFG



ENGENHARIA  
DE SOFTWARE

**Atividade: Dia de Trabalho**

**Tarefa: Desenvolvimento Dirigido Por Testes (TDD)**

**Versão 0.1**

Atividade: Dia de Trabalho	Versão: 0.1
Tarefa: Desenvolvimento Dirigido por Testes (TDD)	Data: 18/12/2012
LMP-P03.2.1	

## Histórico da Revisão

<b>Data</b>	<b>Versão</b>	<b>Descrição</b>	<b>Autor</b>
03/12/2012	0.1	Elaboração Inicial	Emerson José Porfírio
16/12/2012	0.1	Diagrama da tarefa	Emerson José Porfírio

Atividade: Dia de Trabalho	Versão: 0.1
Tarefa: Desenvolvimento Dirigido por Testes (TDD)	Data: 18/12/2012
LMP-P03.2.1	

## Sumário

1.	Objetivos	4
1.1	Escopo	4
2.	Introdução	4
3.	Tarefa: Desenvolvimento Dirigido Por Testes (TDD)	4
4.	Metas	4
5.	<i>Input</i>	5
5.1	Pré-condições	5
5.2	Entradas	5
6.	<i>Output</i>	5
6.1	Pós-condições	5
6.2	Saídas	5
7.	Diagrama da tarefa: Desenvolvimento Dirigido Por Testes (TDD)	6
8.	Passos	6
9.	Template	7
10.	Papéis	7
11.	Padrões Relacionados	7
12.	Riscos	7
13.	Referências	8

Atividade: Dia de Trabalho	Versão: 0.1
Tarefa: Desenvolvimento Dirigido por Testes (TDD)	Data: 18/12/2012
LMP-P03.2.1	

# Tarefa: Desenvolvimento Dirigido Por Testes

## 1. Objetivo

Apresentar e documentar a tarefa Desenvolvimento Dirigido Por Testes (TDD) da atividade Dia de Trabalho (Processo Produzir) que faz parte do LMP – Logiciel Mobile Process a ser utilizado pelo Grupo de Estudo Logiciel como trabalho prático para as disciplinas de Integração I e de Desenvolvimento de Software para Dispositivos Móveis do curso de Bacharelado em Engenharia de Software do INF - UFG.

### 1.1 Escopo

Tarefa 1 da atividade Dia de Trabalho do LMP – Logiciel Mobile Process (Processo Produzir).

## 2. Introdução

O processo de desenvolvimento LMP – Logiciel Mobile Process abrange atividades acadêmicas referentes aos processos de engenharia de software do INF-UFG. Este oferecerá o apoio ao processo de desenvolvimento para dispositivos móveis, permitindo que o mesmo seja realizado de acordo com o planejamento de tempo e de recursos e com os requisitos funcionais e de qualidade definidos para os projetos propostos.

O modelo foi baseado no Processo MobileD e no RUP, além de seguir as orientações do Guia do MPS.Br 2011 (nível F).

## 3. Tarefa: Desenvolvimento Dirigido Por Testes

O objetivo do TDD é passar confiança aos desenvolvedores e orientar a concepção de código com estrutura clara e mais facilmente verificável. O TDD também está rigidamente acoplado com a prática de refatoração porque o conjunto de teste que é produzido pelo TDD é utilizado durante a refatoração para garantir que a mudança não afete a funcionalidade existente no sistema. Em TDD os testes de unidade são escritos antes do código do programa. O código do programa é então desenvolvido para trabalhar com os testes já escritos.

## 4. Metas

1. Passar confiança ao desenvolvedor de que o código criado funciona;
2. Auxiliar os desenvolvedores para evitar o excesso de engenharia, definindo o escopo do código;
3. Permitir uma depuração rápida e eficiente através de extensa refatoração e de um conjunto de teste que ajuda a identificar os defeitos;
4. Melhorar o desenho de software, produzindo código menos acoplado e mais coeso. Isto é possível

Atividade: Dia de Trabalho	Versão: 0.1
Tarefa: Desenvolvimento Dirigido por Testes (TDD)	Data: 18/12/2012
LMP-P03.2.1	

porque escrever primeiro os testes força o código a ser composto por pequenos blocos, consistentes e independentes e que são mais fáceis de testar;

5. Permitir mudanças mais seguras. Se o código é alterado na manutenção ou na fase de desenvolvimento a funcionalidade do sistema pode ser verificada pela execução do conjunto de testes produzidos com TDD; e

6. Criar documentação de teste do código. Testes descrevem a forma como o código pode ser utilizado e funcionam como uma documentação técnica para o desenvolvedor.

## **5. Inputs**

### **5.1 Pré-condições**

1. **Um ambiente de teste é necessário para TDD.** É virtualmente impossível realizar a prática sem um ambiente de teste viável e fácil de utilizar que permita que os testes de unidade sejam automatizados. Se os testes não forem automatizados, terão menos probabilidade de serem executados com a frequência ou da mesma forma a cada vez e seus benefícios estarão perdidos; e

2. **Os desenvolvedores precisam estar familiarizados com a tecnologia para uma utilização eficiente da presente prática.** É importante saber como a tecnologia funciona. O TDD pode se tornar um fardo muito grande se for apresentado a um projeto em que a equipe seja inexperiente.

### **5.2 Entradas**

1. Definição de tarefas definidas nos guias de planejamento de desenvolvimento de testes e código.

## **6. Outputs**

### **6.1 Pós-condição**

1. Código associado a testes, a tarefa de programação não pode ser considerada concluída sem que o código possua um conjunto de testes com todas as funcionalidades implementadas na tarefa.

2. Todos os testes escritos precisam passar antes que a tarefa de programação definida no cartão de tarefas possa ser considerada completa.

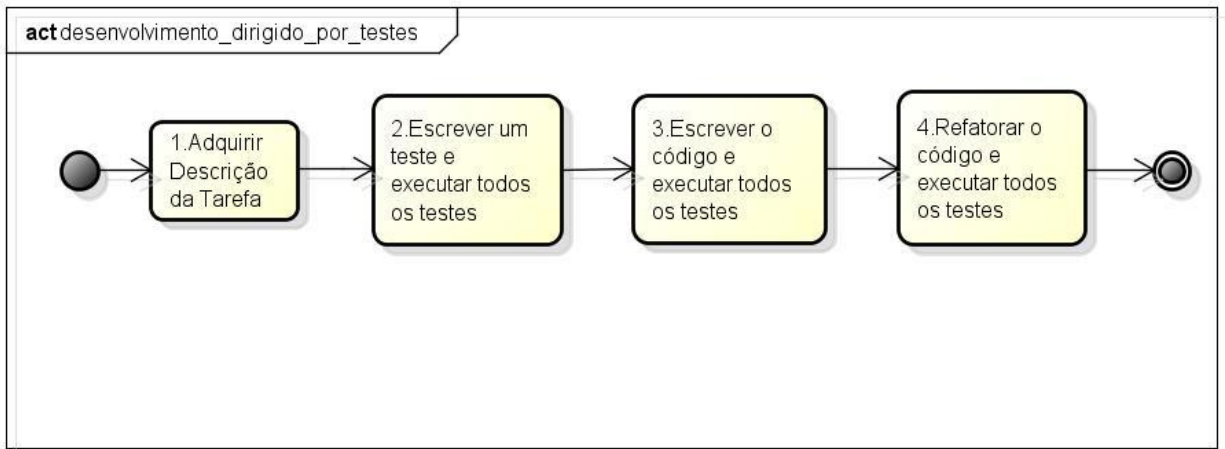
### **6.2 Saída**

1. O código do programa desenvolvido

2. O código do teste desenvolvido

Atividade: Dia de Trabalho	Versão: 0.1
Tarefa: Desenvolvimento Dirigido por Testes (TDD)	Data: 18/12/2012
LMP-P03.2.1	

## 7. Diagrama da tarefa: Desenvolvimento Dirigido Por Testes



powered by Astah

## 8. Passos

As etapas individuais do Desenvolvimento Dirigido Por Testes (TDD) são:

- 1. Adquirir descrição da tarefa.** Todas as tarefas de programação são definidas no Dia de planejamento. Estas definições de tarefas são então utilizadas como uma definição para um ciclo de TDD único (1 a 4).
- 2. Escrever um teste e executar todos os testes.** Um teste simples será escrito antes dos testes de funcionalidade de programação. Tal teste será criado para ser a interface de uma classe ou módulo. Em TDD o objetivo não é o de testar classes ou módulos de funcionalidade interna, apenas a funcionalidade externa. Por exemplo, funções *privates* para as classes não são testados. O primeiro teste não testa todas as funcionalidades da programação porque o processo deve ser incremental e adicionado a cada iteração dos passos 2 e 3. Após o teste ser escrito a suíte é executada. Este resultado provavelmente irá falhar porque o código de programa que este vai testar ainda não foi escrito.
- 3. Escrever o código e executar todos os testes** Escrever código mais simples possível de programa que faz com que o teste passe com sucesso. Verificar se deu certo executando os testes. Se os testes passarem e toda a funcionalidade do cartão de tarefas for implementado, siga para o próximo passo. Se toda a funcionalidade não estiver implementada, repita o ciclo escrever teste – código (passos 2 e 3) até que a passagem de todos os ensaios e todas as funcionalidades sejam implementadas.
- 4. Refatorar código e executar todos os testes.** Quando todos os testes passarem e todas as funcionalidades do cartão de tarefas forem implementados, o código deve ser refatorado para maior clareza e para remover a lógica duplicada. Mais informações no padrão de Refatoração. Executar testes para verificar que nada foi danificado durante a refatoração. Se todos os testes passam, iniciar um novo ciclo a partir do passo 1.

Atividade: Dia de Trabalho	Versão: 0.1
Tarefa: Desenvolvimento Dirigido por Testes (TDD)	Data: 18/12/2012
LMP-P03.2.1	

## 9. Templates

N/A

## 10. Papéis

As seguintes funções podem ser identificadas na execução do Desenvolvimento Dirigido Por Testes:

1. A equipe de projeto / Desenvolvedor. Geralmente o TDD é realizado com programação em pares, os papéis apresentados no padrão de Programação em Pares serão aplicadas aqui também.

## 11. Padrões Relacionados

Outros padrões que fazem parte desta ou são associados com a tarefa são identificados abaixo:

- **Dia de Planejamento:** As tarefas utilizadas como base para o TDD são definidas em Dia de Planejamento.
- **Programação em pares:** As tarefas de programação são feitas utilizando tanto programação em pares quanto TDD.
- **Refatoração:** Após a todas as funcionalidades serem implementadas e todos os testes passarem durante o ciclo único de TDD, a refatoração será utilizada para “limpar” e gerar o código.
- **Integração Contínua:** Depois que a tarefa única é implementada com o TDD, o código e a suíte de testes serão integrados à base do código comum. Após esta integração, todos os testes TDD no sistema são executados para verificar se a integração “quebrou” a funcionalidade do sistema.

## 12. Riscos

Os possíveis riscos que podem resultar de Desenvolvimento Dirigido Por Testes (TDD), bem como as soluções incluindo ações preventivas para evitá-los e medidas a tomar para minimizar seus efeitos são discutidos aqui:

- **Não há desenvolvedores dispostos a utilizar TDD.** O TDD pode parecer muito trabalho para o desenvolvedor. Eles têm que escrever código de teste sobre o código de programa. Além disso, muitas vezes há um preconceito contra TDD: os desenvolvedores afirmam que o código não pode ser testado. Tais razões podem levar à falta de vontade em praticar TDD. Solução: Uma pessoa do grupo de apoio que possua experiência em TDD deve ser incluída na equipe no início do projeto. Seu papel seria o de fornecer à equipe conselhos sobre a prática e como manter a equipe no caminho certo.

Atividade: Dia de Trabalho	Versão: 0.1
Tarefa: Desenvolvimento Dirigido por Testes (TDD)	Data: 18/12/2012
LMP-P03.2.1	

- **O ambiente para o TDD não está disponível.** As vantagens às quais o TDD traz para desenvolvimento de software são em sua maioria perdidos se o ambiente não estiver disponível. Solução: o ambiente deve estar disponível para o trabalho antes do início do projeto. Também a formação adequada deve ser fornecida para a equipe de desenvolvimento ao utilizar o TDD.

### 13. Referências

- <http://agile.vtt.fi/mobiled.html>