INSTITUTO DE INFORMÁTICA
UFG

ENGENHARIA DE SOFTWARE

Atividade: Dia de Trabalho

Tarefa: Refatoração

Versão 0.1

Histórico da Revisão

Data	Versão	Descrição	Autor
26/12/2012	0.1	Elaboração Inicial	Emerson José Porfírio
26/12/2012	0.1	Diagrama da tarefa	Emerson José Porfírio

Sumário

Tarefa: Refatoração

1. Objetivo

Apresentar e documentar a tarefa Refatoração da atividade Dia de Trabalho (Processo Testar&Fixar) 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 5 da atividade Dia de Trabalho do LMP – Logiciel Mobile Process (Processo Testar&Fixar).

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: Refatoração

O objetivo da refatoração é melhorar a estrutura interna do software existente, sem modificar seu comportamento externo. Com pequenas melhorias no código, a refatoração garante que o software seja modificável, prorrogável e mais legível. Quando a refatoração torna-se um hábito regular durante o desenvolvimento isso reduz a necessidade de projetar mais a frente. Em vez disso o software é desenvolvido através da adaptação às mudanças.

4. Metas

1. Melhorar o design do software existente;
2. Tornar o software mais fácil de entender;
3. Ajudar a localizar bugs;
4. Produzir código mais rapidamente; e
5. Melhorar os requisitos não-funcionais (por exemplo: segurança e desempenho).

5. *Inputs*

5.1 Pré-condição

1. Uma suíte de teste automatizada é necessária para garantir que refatorações possam ser realizadas com segurança. Possuindo um extenso conjunto de testes de unidade, o desenvolvedor pode ter a certeza de que o comportamento externo não será quebrado durante a refatoração. Os testes de unidade devem ser capazes de serem executados automaticamente para acelerar o processo de refatoração.

5.2 Entradas

1. O Código fonte existente é atualizado a partir do repositório de código.
2. Os testes de unidade são criados para cada código refatorado se já não estiverem presentes.

6. *Outputs*

6.1 Pós-condições

1. O Projeto existente é melhorado,

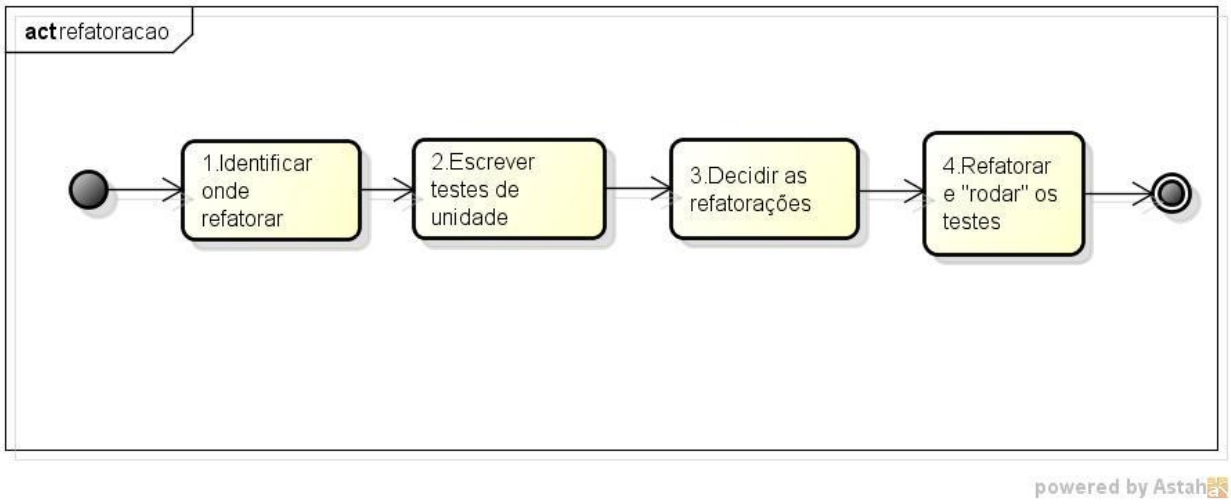
2. Os Testes de unidade relevantes são refeitos,

3. Todos os testes devem passar até a refatoração ser considerada feita.

6.2 Saída

1. Programa de código desenvolvido.
2. Teste de código desenvolvido.

7. Diagrama da tarefa: Refatoração



8. Passos

As etapas individuais de **Refatoração** são:

1. **Identificar onde refatorações precisam ser aplicadas**. A abordagem mais comum para identificar onde a refatoração precisa ser feita é localizar maus cheiros no código. Estes são, por exemplo, as duplicidades de código e as grandes classes com muitas responsabilidades.

2. **Escrever testes de unidade**. Antes da refatoração poder ser aplicada com segurança o código refatorado deve possuir um conjunto de testes.

3. **Decidir quais refatorações precisam ser aplicadas**. Quando um mau cheiro é detectado no código, um método de refatoração para melhorar o mau cheiro específico deve ser identificado;

4. **Refatorar e testar**. O processo de refatoração consiste na aplicação de pequenos passos de refatoração juntamente com o teste contínuo do software refatorado.

9. Templates

1. "Maus cheiros". Listas de maus cheiros comuns podem ser utilizadas para identificar as necessidades de refatoração. As lista também identificam refatorações elementares se necessário.

10. Papéis

As seguintes funções podem ser identificadas na execução da **Refatoração**:

A **equipe de projeto / desenvolvedor**. A comunicação entre o par de programadores também pode produzir idéias para a refatoração eficiente.

11. Padrões Relacionados

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

- **Programação em pares**: As tarefas de programação são feitas utilizando tanto programação em pares como o TDD;
- **Test Driven Development**: casos de teste são refeitos antes do código sofrer refatoração;
- **Integração Contínua**: Depois da refatoração ter sido implementada com TDD o código e a suíte de testes são integrados a base de código comum. Após a 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 Refatoração, bem como as soluções incluindo ações preventivas para evitá-los e medidas a tomar para minimizar seus efeitos são discutidos aqui:

- **Os testes completos não são feitos e novos erros são introduzidos**. Existe a possibilidade da refatoração introduzir novos bugs ao sistema, se este não possui um bom conjunto de testes Solução: Executar extensos casos de teste em primeiro lugar e melhorá-los assim que erros não detectados anteriormente sejam encontrados.
- **Uma ferramenta de refatoração não está disponível**. Uma bem sucedida e econômica refatoração necessita de apoio ferramental. Ferramentas para refatoração dependem de linguagens e ambientes de desenvolvimento específicos. Ferramentas para refatoração em Symbian e C + +, em geral, são poucas em número. Solução: Mudança do ambiente de desenvolvimento para um que possua ferramentas de refatoração ou implementar refatorações manualmente.

13. Referências

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