Modalidade do trabalho: Relatório técnico-científico
Evento: XX Seminário de Iniciação Científica

ESTUDO DA APLICABILIDADE DO TDD COM DESENVOLVIMENTO GENEXUS[1]

Cristiano Rafael Steffens[2], Ivan Luis Gunkel[3], Cristiano Schwening[4].

[1] Trabalho de Conclusão do Curso de Bacharelado em Sistemas de Informação da Faculdade Três de Maio - SETREM
[2] Bacharelando em Sistemas de Informação - SETREM
[3] Bacharelando do Curso de Sistemas de Informação - SETREM
[4] Mestre em Computação Aplicada pela Universidade do Vale do Itajaí

Resumo: Este resumo tem por objetivo apresentar o estudo do Desenvolvimento Dirigido por Testes em uma empresa de desenvolvimento de sistemas computacionais. Buscou-se aprofundar o conhecimento da técnica de TDD e avaliar a possibilidade de aplicação em projetos estruturados em ciclo cascata com realimentação que utilizam ferramenta de desenvolvimento geradora de código. O problema proposto inicialmente questionava como utilizar o TDD dentro de um modelo clássico, avaliando quais seriam os impactos sobre as pessoas envolvidas e os resultados, sendo que as hipóteses questionavam como seria a adoção do TDD pela equipe, se GeneXus como ferramenta de desenvolvimento seria um fator limitante e se a utilização de TDD poderia produzir um software com menos falhas e erros. Constatou-se durante o andamento do trabalho, que a ferramenta de desenvolvimento GeneXus inviabiliza a aplicação da prática. Através do método hipotético-dedutivo, aliado a técnica de pesquisa bibliográfica, procurou-se testar as possibilidades, restrições e condições de forma a justificar ou não sua adoção definitiva nos demais projetos da empresa.

Palavras-Chave: desenvolvimento dirigido por testes, automação de testes.

Introdução

A engenharia de software busca fornecer ferramentas, recursos e estrutura a fim de guiar o profissional de forma adequada na fabricação de software de qualidade. Para tanto, propõe um conjunto de melhores práticas, técnicas de gestão e organização da equipe, especificação de requisitos, modelagem, implementação, testes, implantação e manutenção de sistemas.

Este trabalho enfoca uma pequena parte dos aspectos relacionados às fases de codificação e testes de software. Embasando-se no um princípio de que um bom código é fundamental para a continuidade e o sucesso das empresas. O Desenvolvimento Dirigido por Testes é uma técnica que visa melhorar a arquitetura do programa através da antecipação dos testes em relação à codificação, obrigando o programador a refletir sobre seu trabalho antes de iniciá-lo. Além disso, encoraja a utilização de testes unitários e refatoração frequente, resultando em um produto testado e enxuto, que atende as necessidades do cliente e seja de fácil manutenção.

Esta pesquisa foi realizada junto à empresa Migrate Company Sistemas de Informação LTDA, de Três de Maio - RS, tendo como objeto de estudo a equipe responsável pelo produto FiscalDocs com o

objetivo de estudar e aplicar o TDD. Com esta definição, buscou-se planejar o tempo disponível organizando as tarefas e estudando a bibliografia relacionada ao Desenvolvimento Dirigido por Testes, bem como as ferramentas e práticas de desenvolvimento e testes utilizadas atualmente pela empresa.

Metodologia

Esta pesquisa fez uso do método hipotético-dedutivo relacionando as hipóteses do trabalho ao processo dedutivo. Considerado puramente lógico, é relacionado à experimentação, observação e prova das hipóteses construídas, buscando encontrar uma solução, através de tentativas e eliminação de erros (Lakatos e Marconi, 1992). Aliado ao método hipotético dedutivo, utilizou-se a técnica de documentação indireta, especificamente através de pesquisa bibliográfica e documental, no sentido de embasar as considerações apresentadas.

De forma geral a pesquisa seguiu as etapas de estudo e documentação bibliográfica dos temas abordados, estudo da ferramenta de programação GeneXus e outras ferramentas relacionadas, documentação do processo atual de desenvolvimento na empresa, elaboração da documentação para orientar a introdução dos testes unitários aos desenvolvedores, aplicação, coleta dos dados e mensuração dos resultados obtidos e análise dos resultados.

Resultados e discussão

Para o desenvolvimento desta pesquisa, deu-se ênfase à fundamentação teórica com objetivo de dar credibilidade ao trabalho, trazendo referência às pesquisas e conhecimentos já construídos e publicados, estabelecendo a evolução do assunto e dando sustentação ao tema estudado.

Como grande área, teve-se a engenharia de software, que é, para Sommerville (2003, p.05) uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, depois que ele entrou em operação.

Uma das preocupações da engenharia é com os processos de software. Processos de software representam o conjunto de atividades e resultados agregados, que podem levar à produção de um produto de software. Pressman (2006) agrega que além de atividades e resultados, são somados a estes, métodos, ferramentas e práticas, que devem ser aplicadas sistematicamente, agrupadas em fases, possuindo responsáveis e gerando entradas e saídas. Define-se quem faz o quê, quando e como deve ser feito para o objetivo ser atingido.

Pressman (2006) e Sommerville (2003) comentam que existem passos e atividades comuns que devem ser observadas, sendo aplicáveis a grande maioria de projetos de software. Estes são, dentro de um modelo cascata, seqüenciados da seguinte forma: especificação e comunicação, projeto e planejamento, modelagem, construção e validação, implantação e evolução do software.

Este trabalho teve um foco específico sobre a etapa de codificação. Para Paula Filho (2009), é nesta etapa que todas as visões, percepções sobre a análise construída, tanto de requisitos como fluxos e atividades são convertidas e aplicadas em uma linguagem de programação. Este processo é feito pelos programadores, que devem compreender além da semântica da linguagem para transcrever a visão do

usuário, o negócio implícito na análise e como este afeta performance, usabilidade e manutenibilidade do código e do software.

O Desenvolvimento Dirigido por Testes é um conceito introduzido junto com o os métodos ágeis e principalmente com o Extreme Programming. O XP destaca-se por ser orientado a testes. Myers (2004, p.179) afirma que o modelo de desenvolvimento tradicional sugere que o desenvolvimento ocorra antes, ocorrendo posteriormente o desenvolvimento e aplicação dos testes. Já no XP os testes devem obrigatoriamente ser criados antes, e depois produzir-se um código fonte que seja aprovado nos testes.

Kent Beck, na década de 80, foi um dos primeiros a direcionar as atenções para as metodologias de desenvolvimento ágeis, o modelo de processo XP e também as técnicas de programação dirigida por testes. Beck é o autor das principais obras na área de TDD. Koskela (2008, p.4) afirma, no entanto, que a prática do TDD surgiu muito antes do conceito em si. De acordo com o autor, são conhecidos casos onde a equipe preparava os testes antes da implementação do código. É preciso considerar que muitas das técnicas que hoje são consideradas triviais e corriqueiras foram a algumas décadas dominadas apenas por pessoas que se dispunham a trabalhar e melhorá-las.

Trucchia e Romei (2010, p.49) explicam o funcionamento do Test-Driven Development afirmando que se trata de uma técnica de desenvolvimento de software que consiste na repetição de um curto ciclo de desenvolvimento dividido em cinco etapas assim resumidas:

1. Adicionar um teste.
2. Executar todos os testes e ver se o novo código falha.
3. Escrever o código mais simples que fará com que o teste passe.
4. Executar testes e vê-los bem sucedidos.
5. Refatorar o código.

De acordo com Trucchia e Romei (2010, p.61), Bender e McWerther (2011, p.43) e Koskela (2008, p.16), o mantra principal do Test-Driven Development é Red, Green, Refactor ou em português: Vermelho, Verde, Refatorar. Trata-se de um mnemônico que se refere ao ciclo de produzir testes que falham, implementar as funcionalidades de forma a satisfazer os testes e posteriormente refatorar o código produzido sem falhar nos testes. Em alguns casos o próprio teste pode ser refatorado.

Os testes unitários, diferentemente dos testes funcionais, operacionais e de sistema, enfocam o processo de verificação na menor unidade do projeto do software, que é um componente ou módulo do software. Nesta definição de Pressman (2006), se utiliza da descrição do projeto no nível de componente para testar e descobrir erros dentro dos limites do módulo. Paula Filho (2009) defende que o teste de unidade é um teste de desenvolvimento, realizado pelos desenvolvedores do software e que exercita detalhadamente e exaustivamente uma unidade de código, por exemplo, partes funcionais do código e classes lógicas.

Bender e McWerther (2011, p.24) situam o ano de 2005 como um marco na popularização do TDD, afirmando que a partir deste período ocorreu a massificação e o desenvolvimento de muitas ferramentas e frameworks para testes unitários. Se anteriormente a verificação dos códigos-fonte produzidos era feita de forma quase artesanal, a partir da utilização destas ferramentas permitia desenhar os testes unitários, controlar sua execução e permitir que o desenvolvedor visualizasse os resultados dentro de sua ferramenta de desenvolvimento.

Modalidade do trabalho: Relatório técnico-científico
Evento: XX Seminário de Iniciação Científica

De forma a justificar a aplicabilidade do TDD, estudou-se também sobre a ferramenta de desenvolvimento GeneXus utilizada pela empresa. Segundo Lisboa (2006), GeneXus é uma ferramenta de desenvolvimento de software baseada no conhecimento do analista, onde este ao modelar o sistema o faz em alto nível de abstração, tendo seu foco no conhecimento do negócio, deixando que GeneXus se dedique a programação de baixo nível.

GeneXus parte das visões dos usuários, capturando seu conhecimento e o sistematizando em uma base de conhecimento (KB). A partir desta base de conhecimento, GeneXus é capaz de desenhar, gerar e manter, de forma totalmente automática a estrutura da base de dados e os programas da aplicação, os quais vão permitir que os usuários interajam com as visões das quais os analistas partiram. Resumidamente, GeneXus é um programa para fazer programas (ARTECH, 2011).

Para Lisboa (2006), existe um grande mito sobre GeneXus ser uma plataforma de desenvolvimento ou uma linguagem de programação. Por um lado, trata-se de uma linguagem, pois mesmo desenhando, gerando e mantendo automaticamente a base de dados e os aplicativos, muito código necessita ser escrito para atender os requisitos dos usuários. Por outro lado, parece não ser, pois não devemos escrever qualquer linha de código para fazer o mesmo programa de forma simples.

Para aplicação de testes unitários em um programa desenvolvido utilizando-se a ferramenta GeneXus, utiliza-se o framework GxUnit. Trata-se de um complemento que deve ser instalado junto ao IDE (Ambiente Integrado do Desenvolvimento) principal do GeneXus. De acordo com Carro (2012) os principais objetivos do GxUnit são: criar e manter testes unitários automatizados, integrar-se com a IDE, executar testes unitários automatizados e manter registros dos testes.

Basicamente, o fluxo de criação dos testes unitários se dá pela seguinte forma:

- A partir da base de conhecimento, o analista define os casos de teste unitários em GeneXus.
- A criação de um caso de teste está ligado a um objeto GeneXus, que pode ser do tipo Procedimento, Transação e Provedor de Dados. Portanto, ao criar o caso de teste, o analista será questionado a qual objeto este teste está relacionado.
- Após implementar o código a ser testado, selecionam-se quais casos de teste serão testados na bateria atual.
- Ao clicar em Test, GeneXus irá especificar e construir em linguagem de baixo nível a bateria de testes e executa-la, apresentado ao desenvolvedor um relatório detalhado sobre o tempo de execução de cada caso de teste, qual falhou e qual foi executado com sucesso, bem como os valores esperados para o teste passar e o que foi passado pelo programa.

Dada a incompatibilidade entre o processo proposto pelo TDD e o processo de aplicação de testes unitários comprovou-se que a ferramenta de desenvolvimento GeneXus é um fator impeditivo para a aplicação da técnica.

Conclusões

A pesquisa permite comprovar a inviabilidade da aplicação do Desenvolvimento Dirigido por Testes utilizando a ferramenta RADD GeneXus. A referência bibliográfica adequada permite compreender os princípios norteadores da técnica de TDD, fundamentando uma avaliação precisa e correta, levando em conta as limitações impostas pelo contexto da empresa e da ferramenta de desenvolvimento utilizada.

Comprova-se, conceitualmente, que o trabalho em um modelo clássico não é fator limitante, desde que pequenas adequações sejam feitas na etapa de codificação. Por outro lado, ao evidenciar-se a impossibilidade da aplicação, fica impossibilitada a comprovação das outras hipóteses do trabalho. Assim não se corroboram as hipóteses que afirmavam que o TDD teria fácil aceitação da equipe e que sua utilização permite produzir um software com menos falhas e erros

Referências Bibliográficas

ARTECH. GeneXus. Disponível em <www.genexus.com>. Acesso em: 27 de novembro de 2011.
BECK, Kent. Extreme Programming Explained. ISBN: 0201616416. Addison-Wesley, 1999.
BECK, Kent; BEEDLE, Mike; BENNEKUM, Arie Van; COCKBURN, Alistair; CUNNINGHAM, Ward; FOWLER, Martin; GRENNING, James; HIGHSMITH, Jim; HUNT, Andrew; JEFFRIES, Ron; KERN, Jon; MARICK, Brian; MARTIN, Robert C.; MELLOR, Steve; SCHWABER, Ken; SUTHERLAND, Jeff; THOMAS, Dave. Manifesto para Desenvolvimento Ágil de Software. Disponível em < http://agilemanifesto.org/isso/ptbr/ >. Acesso em 20 de dezembro de 2011
BENDER, James; MCWERTHER, Jeff. Professional Test Driven Development With C#: Developing Real-World Applications with TDD. 1ª Ed. Indianapolis: Wiley Publishing Inc, 2011.
CARRO, Nicolás. Acerca de GXUnit. Disponível em <https://sites.google.com/site/proyectogxunit/project-definition>. Acesso em 14 de abril de 2012.
GÜLLICH, Roque Ismael da Costa; LOVATO, Adalberto; EVANGELISTA, Mário dos Santos. Metodologia da Pesquisa: normas para apresentação de trabalhos: redação, formatação e editoração. Três de Maio: SETREM, 2007.
KOSKELA, Lasse. Test Drive: Pratical TDD and Acceptance TDD for Java Developers. New York: Manning Publications Co. ,2008.
LAKATOS, Eva M; MARCONI, Marina de A. Metodologia do Trabalho Científico: procedimentos básicos, pesquisa bibliográfica, projeto e relatório, publicações e trabalhos científicos. 4ª ed. São Paulo: Atlas, 1992.
LISBOA, Daniel M. GeneXus: Desarrollo Basado em el Conocimento. Uruguay: Grupo Magró, 2006.
MARTIN, Robert C. Código Limpo. Rio de Janeiro: Alta Books, 2009.
MYERS, Glenford J. The art of software testing. 2ª ed. John Wiley and Sons, 2004.
PAULA FILHO, Wilson de P. Engenharia de Software: fundamentos, métodos e padrões. 3ª ed. Rio de Janeiro: LTC, 2009.
PRESSMAN, Roger S. Engenharia de Software. 6ª ed. Rio de Janeiro: McGraw-Hill, 2006.
STEFFENS, Cristiano; GUNKEL, Ivan; SCHWENING, Cristiano. Estudo de TDD e aplicação de testes unitários automatizados em empresa de desenvolvimento de sistemas. Trabalho de Conclusão de Curso de Sistemas de Informação. SETREM: 2012.
TRUCCHIA, Francesco; ROMEI, Jacopo. Pro PHP Refactoring. New York: Apress, 2010.