Kalibro Metrics: a multi-repository and multi-language source code analysis tool

Carlos Morais[1], Paulo Meirelles[1], Vinícius Daros[1], Fabio Kon[1]
[1]Department of Computer Science
Institute of Mathematics and Statistics
University of São Paulo, Brazil
{morais, paulormm, vkdaros, fabio.kon}@ime.usp.br

Carlos Santos Jr.[1,2]
[2]Horizon Institute
University of Nottingham, United Kingdom
carlos.denner@nottingham.ac.uk

***Resumo*—Source code metrics are not new, but they have not yet been fully explored in software development. For example, most metrics do not have known thresholds to guide their use by non-experts. Most metrics tools show isolated numeric values, which are not easy to understand because the interpretation of these values depends on the implementation context. Kalibro Metrics goal is to improve the readability of source code metrics. It allows a source code metric user to create a configuration of thresholds associated with qualitative evaluation, including comments and recommendations. Using these configurations, Kalibro shows metric results in a friendly way, helping software engineers to spot design flaws to refactor, project managers to control source code quality, and software adopters and researchers to compare specific source code characteristics across free software projects. These configurations are shared among its users via the Kalibro Web Service and a source code tracking network to enhance metric results interpretation.**

***Index Terms*—Software metrics, source code analysis, thresholds configuration, Web Service integration, source code tracking network, Free Software.**

I. Introdução

O desenvolvimento de software usando métodos empíricos, como software livre e métodos ágeis, é uma realidade. A prática básica desses métodos são os testes automatizados, preocupando-se com as aplicações de testes em sistemas cada vez mais dinâmicos e reutilizáveis com propostas para aumento de qualidade e produtividade de software [1]. Adicionalmente, as áreas relacionadas a usabilidade vem sendo estudadas para serem aplicadas no contexto de desenvolvimento empírico de software. Criar software que seja útil e fácil de usar é fator importante para a evolução dos sistemas [2]. O Noosfero, um sistema para desenvolvimento de redes sociais na qual faz parte do nosso estudo, tem tido algumas iniciativas por parte dos desenvolvedores e usuários da plataforma para a melhoria da usabilidade da ferramenta.

II. Métodos Empíricos de Desenvolvimento de Software

O Empirismo baseia-se na aquisição de sabedoria através da percepção do mundo externo, ou então do exame da atividade da nossa mente, que abstrai a realidade que nos é exterior e as modifica internamente [**?**].

Software livre é uma filosofia que trata programas de computadores como fontes de conhecimento que devem ser compartilhados entre a comunidade. De acordo com [3], ativista fundador do movimento software livre, um software deve seguir quatro princípios:

1) Liberdade de execução para qualquer uso;
2) Liberdade de estudar o funcionamento de um software e de adaptá-lo às suas necessidades
3) Liberdade de redistribuir cópias;
4) Liberdade de melhorar o software e de tornar as modificações públicas de modo que a comunidade se beneficie da melhoria.

A utilização de métodos ágeis no desenvolvimento de software tem como características intrínsecas a flexibilidade e rapidez nas respostas a mudanças. A agilidade, para uma organização de desenvolvimento de software, é a habilidade de adotar e reagir rapidamente e apropriadamente a mudanças no seu ambiente e por exigências impostas pelos "clientes" [4].

Existe uma relação entre as práticas ágeis e o desenvolvimento baseado em software livre. [5] fala que o desenvolvimento baseado em software livre vem crescendo, porém a essência da comunidade ao redor do programa é de manter indivíduos que interajam de forma a produzir o que é mais importante. As ferramentas apenas possibilitam isso.

O desenvolvimento baseado em software livre e as práticas ágeis compartilham os mesmos valores, pois ambos são métodos empíricos, que buscam tomar decisões rápidas de acordo com a percepção do mundo externo.

III. Testes

Para [6] a automação de testes é uma prática ágil, eficaz e de baixo custo para melhorar a qualidade dos sistemas de software. No entanto utilizar testes automatizados como uma premissa básica do desenvolvimento é um fenômeno relativamente recente, com início em meados da década de 1990. Além do fato de ser uma técnica bastante utilizada pelas metodologias ágeis de desenvolvimento.

Os testes automatizados afetam diretamente a qualidade dos sistemas de software, portanto, agregam valor ao produto final, mesmo que os artefatos adicionais produzidos não sejam visíveis para os usuários finais dos sistemas [7]. Esses testes

podem ser divididos em diversos tipos, o que facilita a manutenção dos mesmos:

1) **Testes de unidade:** testes de correção responsável por testar os menores trechos de código de um sistema que possui um comportamento definido e nomeado [7]. Normalmente, esse teste é associado a funções para linguagens procedimentais e métodos em linguagens orientadas a objetos.
2) **Testes funcionais:** testes que tem como objetivo verificar a eficiência dos componentes de um sistema [8].
3) **Testes de aceitação:** são testes de correção e validação, idealmente especificados por clientes ou usuários finais do sistema para verificar se um módulo funciona como foi especificado [9]. Testes de aceitação devem utilizar linguagem próxima da natural para evitar problemas de interpretação e de ambiguidades [10].

Desenvolvimento dirigido por testes, TDD *(Test-Driven Development)*, é uma técnica de desenvolvimento de software que se dá pela repetição disciplinada de um ciclo curto de passos de implementação de testes e do sistema [11]. O ciclo curto de passos definidos por TDD cria uma dependência forte entre codificação e testes, o que favorece e facilita a criação de sistemas com alta testabilidade [7].

Desenvolvimento dirigido por comportamento *(BDD - Behavior Driven Development)* é uma prática que recomenda o mesmo ciclo de desenvolvimento de TDD, contudo, induzindo a utilização de uma linguagem ubíqua entre cliente e equipe de desenvolvimento, substituindo termos como assert, *assert, test case, test suite* por termos mais comuns ao cliente, como *should, context, specification* [7]. Embora seja principalmente uma ideia de como um processo de desenvolvimento de software deve ser gerenciado, a prática do BDD assume a utilização de ferramentas como suporte para o desenvolvimento de software [12]. O BDD utiliza essas ferramentas para que os testes tenham como ponto de partida o comportamento das funcionalidade do sistema.

Testes automatizados devem ser desenvolvidos com prioridade, buscando um rápido *feedback*, contribuindo assim com a melhoria do sistema. Para isso é necessário que os cenários de testes estejam bem definidos junto à equipe.

IV. Usabilidade

Ainda, usabilidade é definida como o fator que assegura que um produto ou serviço é fácil de usar, eficiente e agradável a partir do ponto de vista do usuário [13]. Adicionalmente, existem metas de usabilidade que servem para guiar o desenvolvimento de produtos fáceis de usar, eficientes e agradáveis [13]. São elas:

1) **Utilidade:** é a medida na qual o sistema propicia o tipo certo de funcionalidade, de maneira que os usuários possam realizar aquilo de que precisam ou desejam.
2) **Eficácia:** refere-se ao quanto um sistema é bom em fazer o que se espera dele.
3) **Eficiência:** refere-se à maneira como o sistema auxilia os usuários na realização de suas tarefas.
4) **Segurança:** implica em proteger o usuário de condições perigosas e situações indesejáveis.
5) **Facilidade de aprendizado:** refere-se a qual fácil é aprender a usar o sistema.
6) **Facilidade de lembrar como se usa:** refere-se à facilidade de lembrar como utiliza um sistema, depois de já ter aprendido como usar.

Para entender o que é usabilidade e como ela está inserida no ciclo de vida do desenvolvimento de software, precisamos compreender as relações que o termo tem com as diversas áreas que a envolve:

Design centrado no usuário é um campo de estudo que reúne metodologias de *design* nas quais o público álvo de um produto ou serviço influencia as diretrizes e requisitos do sistema [14]. Para [14], cunhador do termo, o *design* centrado no usuário é uma filosofia baseada nas necessidades e interesses dos usuário, com ênfase em fazer produtos usáveis e inteligíveis.

Toda a interação com um produto, serviço ou marca. O termo é usado frequentemente para sintetizar toda a experiência com um produto de software. Não engloba somente as funcionalidades e sim o quanto o aplicativo é agradável ao usuário [15]. O termo User Experience (UX) foi cunhado por Donald Norman na década de 80 para cobrir todos os aspectos da experiência da pessoa com o sistema. Acreditava que as definições de interface do usuário e usabilidade limitava o entendimento sobre o que o trabalho dele representava.

Um dos grandes problemas encontrados em softwares livres é a pouca atenção dada aos aspectos referentes a usabilidade e acessibilidade das aplicações. Acredita-se que um dos principais problemas que contribui para a falta de usabilidade de software livre é sua própria comunidade que apenas enfatiza na criação e, melhoria e teste do código fonte.

Segundo [13], uma das causas da baixa adoção de softwares livres em mercados de larga escala é a baixa qualidade dos estilos de interação implementados nas interfaces dos produtos. Uma grande maioria não se preocupa com bons elementos de interface com usuário (UI).

Entender o perfil do usuário é um dos principais pontos que devem ser levados em consideração pelos desenvolvedores de software em geral. Cada perfil de usuário tem suas particularidades e suas expectativas quanto a utilização do sistema. Quando falamos de usuários com experiência de uso de softwares semelhantes é preciso ter uma maior atenção na usabilidade para que possa ter uma boa aceitação e menor impacto com as mudanças.

Os métodos ágeis é uma abordagem de desenvolvimento que têm como princípios um conjunto de valores com foco no cliente e nas pessoas envolvidas, na entrega constante de software funcional, no desenvolvimento iterativo baseado em ciclos curtos de entrega e na aceitação da constante mudança dos requisitos ao longo do desenvolvimento.

Tanto os métodos ágeis e o IHC tem esses mesmos princípios, o que seria interessante a aplicação das técnicas utilizadas em IHC no contexto de uma abordagem ágil. Essa semelhança entre os valores proporciona um quadro favorável à integração

minimalista de práticas de IHC em ambientes ágeis. Podemos citar alguns desse valores como por exemplo: (i) ciclos curtos com entregas contínuas e incrementais, que favorecem a aplicação de técnicas de prototipagem; (ii) forte envolvimento do usuário que favorece a aplicação de princípios de projetos participativos e (iii) programação em pares onde em IHC geralmente a avaliação de usabilidade é feita em pares [16].

A integração entre os processos de usabilidade e métodos ágeis é esperada e possível visto que tanto os métodos ágeis como os processo de usabilidade em em comum características que colocam o foco do desenvolvimento nas necessidade e anseios dos usuários finais, na interação entre os *stakeholders* envolvidos e na qualidade final do produto a ser desenvolvido.

V. Estudo experimentao

O estudo deste trabalho, nesta primeira fase, foi baseado nos seguintes ambientes de rede social que utilizam a plataforma noosfero: o portal Participa.Br[1], para as observações sobre a usabilidade do Noosfero; a rede Comunidade.UnB[2] e o Portal UnB Gama[3], para as observações sobre os testes automatizados no Noosfero;

Foi planejado um estudo de usabilidade, cujo o objetivo é analisar a interação dos usuários com o portal Participa.Br a fim de avaliar a qualidade em uso do portal. A partir dos objetivos de medição estabelecidos, foram definidas questões sobre o que é preciso saber de forma a apoiá-la a entender se o objetivo específico foi alcançado. Assim, para cada questão foram definidas as métricas relacionadas na Tabela I:

Questões	Métricas	Diretrizes para interpretação
Q1. Qual o perfil do usuário que utiliza o Participa.Br?	Perfil	Análise de Dados Estatísticos, criação de personas, análise dos dados qualitativos.
Q2. Qual o grau de satisfação do usuário que utiliza o Participa.Br	Grau de satisfação do usuário	Escore da satisfação global pelo usuário (OVERALL)
Q3. Quantidade de tempo gasto para realizar as tarefas	Duração	Tempo gasto

Tabela I
Questões de Pesquisa

Foram levantadas algumas hipóteses para este estudo experimental no Participa.Br:

1) A média do grau de satisfação dos usuários que já utilizaram o portal seria maior do que quem nunca utilizou?
2) O grau de satisfação dos usuários que já tinha contato com o portal é diferente dos que nunca tiveram acesso?

Do ponto de vista de metodológico, elencamos algumas técnicas para identificamos o perfil dos usuários do Participa.Br:

1) **Dados Estatísticos** (*Google analytcs*, Piwiki, entre outros): Através dos dados estatísticos é possível identificar algumas informações sobre o perfil dos usuários que acessam o portal. Nas pesquisas quantitativas não são necessários o contato direto com o usuário. Esses dados estatísticos podem ser coletados de base de dados, redes sociais ou sistemas de análises de sites.
2) **Questionário de identificação de perfil dos usuários:** Para identificar o perfil dos usuários do Portal da Participação social é necessário realizar uma pesquisa qualitativa para levantamento das principais características contextuais dos usuários típicos, de modo a compreender quem são, qual o conhecimento e experiência com a internet e como utilizam para realizar seu trabalho acadêmico ou profissional. A realização dessa pesquisa será feita com os usuários do Portal da Participação Social. A análise do questionário servirá para entender o perfil dos usuários do Portal da participação social, através da investigação de seus interesses. Foi levantado também algumas questões referentes ao uso das funcionalidades do Participa.Br.
3) **Identificação de Personas:** Para a definição de usuários podemos utilizar a técnica de "Persona" que são personagens fictícios criados com base em dados reais. Os Personas atuam como representantes dos usuários reais e representam as necessidades de um grupo maior.

Elencamos algumas técnicas para avaliar a usabilidade do portal Participa.Br:

Técnica	Descrição
Observar Usuários	Um observador irá registrar o tempo gasto por cada participante de caso, avaliar a ferramenta e se necessitou de alguma ajuda
Perguntar aos usuários	Os questionários ASQ e PSSUQ de satisfação dos usuários serã as opiniões dos participantes.

Tabela II
Técnicas de avaliação para os testes com usuários

Os instrumentos de coletas de informações utilizados são dois questionários que são amplamente utilizados para medir a satisfação do usuário com produtos interativos e fornecem medidas padronizadas. São eles o *After-Scenario Questionnaire* (ASQ) [4] e o *Post-Study System Usabiliy Questionnaire* (PSSUQ). O ASQ é destinado ao uso em testes de usabilidade baseados em cenários. Possui três itens que abordam os seguintes componentes de usabilidade: (1) facilidade de conclusão da tarefa, (2) tempo necessário para completar uma tarefa e,(3) a adequação das instruções ou materiais de apoio fornecidos. O PSSUQ é aplicado após a conclusão de todos os cenários com o propósito de fornecer uma avaliação geral geral da usabilidade do sistema. pois permite uma avaliação de usabilidade mais ampla, podendo avaliar 4 fatores e usabilidade (satisfação geral, utilidade do sistema, qualidade da interface e qualidade da informação).

O estudo sobre testes teve seu enfoque na rede colaborativa baseada no noosfero desenvolvida para a Universidade de Brasília (UnB)[5]. Ao decorrer deste trabalho de graduação, foram desenvolvidos, juntamente com seus respectivos testes, alguns *plugins* que serão descritos nesta seção.

[1]Participa.Br
[2]comunidade.unb.br
[3]fga.unb.br

[4]ASQ: Proposto por Lewis
[5]unb.br

Como rede colaborativa dos membros da Universidade de Brasília, o Comunidade UnB necessita possuir restrição de acesso aos usuários, para que somente membros ativos da universidade tenham acesso ao conteúdo da rede colaborativa. Para que esta necessidade fosse satisfeita foi desenvolvido um *plugin* no noosfero, que efetuasse as restrições necessárias, utilizando o protocolo de autenticação da UnB, o LDAP (*Lightweight Directory Access Protocol*).

Com o auxílio de uma ferramenta de análise de código para *Ruby* chamada Rcov, foi obtida a taxa de cobertura de código do *plugin* desenvolvido, além de alguns dados sobre a execução dos testes funcionais e unitários que seguem abaixo:

- Quantidade de testes executados: **96 testes;**
- Quantidade de assertivas executadas: **111 assertivas;**
- Quantiadde de falhas obtidas: **0 falhas;**
- Tempo de execução dos testes: **7.8 segundos;**

Na imagem 1 existem dois gráficos de cobertura de código, o primeiro definido como *'total coverage'* representa a contagem realizada com as linhas em branco e os comentários do código, já o *'code coverage'* representa a contagem realizada sem as linhas em branco e os comentários do código.

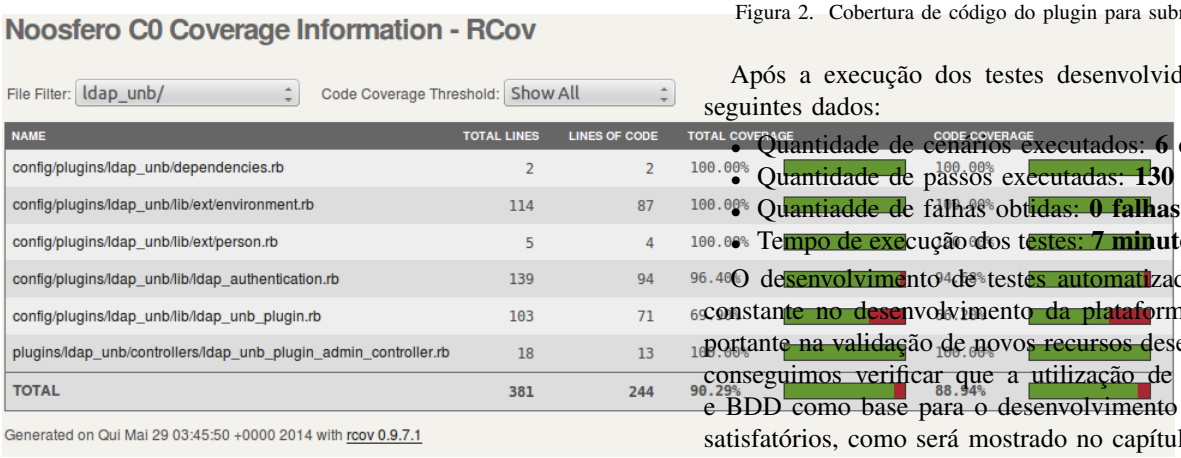

Figura 1. Cobertura de código do plugin LdapUnb

Este *plugin* também foi desenvolvido para a plataforma Noosfero, porém em uma aplicação diferente, o Portal UnB Gama. A ideia deste *plugin* surgiu da necessidade de existir um ambiente virtual em que os trabalhos de conclusão de curso pudessem ser submetidos aos professores e compartilhados com a comunidade acadêmica, buscando assim manter uma forma de versionamento dos trabalhos desenvolvidos e dispensando a necessidade de trabalhos de conclusão de curso impressos. Este *plugin* é responsável por criar uma atribuição de trabalhos, chamada de *work assignment*. Essa atribuição possui algumas funcionalidades específicas como possibilitar que os usuários envolvidos sejam notificados via *email* sobre a submissão de um certo trabalho. Para tal foi necessário instanciar um servidor de *email* para executar estas notificações, assim como criar uma página no Portal FGA[6] para que o usuário pudesse submeter seu trabalho.

[6]urlfga.unb.br

A ferramenta Rcov também foi utilizada para dimensionar a taxa de cobertura de código do *plugin* para envio de TCC, segue os dados sobre a execução dos testes funcionais e unitários:

- Quantidade de testes executados: **28 testes**;
- Quantidade de assertivas executadas: **84 assertivas**;
- Quantiadde de falhas obtidas: **0 falhas**;
- Tempo de execução dos testes: **10,5 segundos**;

Na imagem 2 está representado a cobertura de código, extraída da ferramenta Rcov:

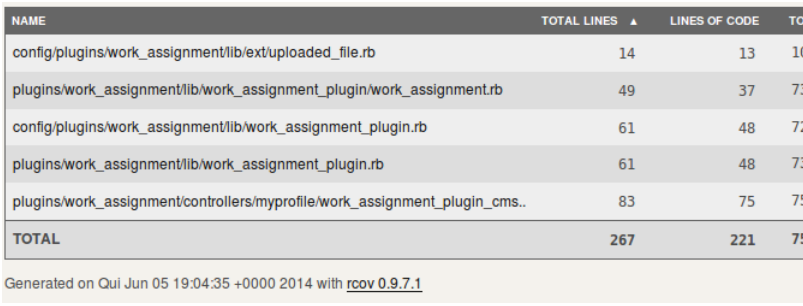

Figura 2. Cobertura de código do plugin para submissão de trabalho

Após a execução dos testes desenvolvidos, obtivemos os seguintes dados:

- Quantidade de cenários executados: **6 cenários**;
- Quantidade de passos executadas: **130 passos**;
- Quantiadde de falhas obtidas: **0 falhas**;
- Tempo de execução dos testes: **7 minutos e 18 segundos**;

O desenvolvimento de testes automatizados é uma prática constante no desenvolvimento da plataforma noosfero e importante na validação de novos recursos desenvolvidos. Assim conseguimos verificar que a utilização de práticas de TDD e BDD como base para o desenvolvimento trouxe resultados satisfatórios, como será mostrado no capítulo a seguir.

Com a proposta de algumas técnicas de avaliação da usabilidade para o Portal da Participação Social, verificamos que muitas delas precisam ser adaptadas para uma melhor adoção nos ambientes de software livre e em métodos ágeis. O teste de usabilidade proposto foi planejado pensando nas práticas clássicas de avaliação da usabilidade que é considerada por vários autores como uma das melhores maneiras de avaliar uma interface.

VI. Considerações finaiss

Durante esta primeira parte do trabalho de conclusão de curso, estudamos o desenvolvimento de software empírico e suas características, assim como a forma que esse desenvolvimento está ligado com testes automatizados e como as práticas de testes podem ser aplicadas ou não em um ambiente real de desenvolvimento que já se encontra estabelecido. Sobre o desenvolvimento de testes automatizados, verificamos que é possível aplicar grande parte das práticas de BDD e TDD no desenvolvimento de uma nova funcionalidade para a plataforma Noosfero, quando esse desenvolvimento está ainda

no levantamento da história, pois observamos dificuldades de desenvolver testes quando o desenvolvimento de uma nova funcionalidade já foi iniciado, o que aconteceu no plugin para o envio de TCC, resultando num desempenho menor dos testes executados.

Com as pesquisas realizadas sobre usabilidade de software, podemos notar que existe estudos na área onde foram criadas metodologias que unem tanto as abordagem ágeis com a abordagem centrada no usuário. É possível fazer a integração das abordagens, mas é necessário que tenha algumas adaptações.

ssim, considerando o que levantamos neste trabalho, chegamos a algumas hipóteses que serão respondidas na segunda fase deste trabalho.

- Como inserir os princípios de usabilidade dentro do processo desenvolvimento empírico de software?
- É possível alcançar melhores resultados em testes de usabilidade utilizando práticas do BDD e TDD, durante o desenvolvimento de software?

Nesse contexto, a ideia do estudo de caso inicial sobre projeto Participa.Br foi conhecer como funciona algumas técnicas de avaliação da usabilidade. Foi escolhido o Portal da Participação Social por ser um dos projetos apoiados pela faculdade. Propomos algumas técnicas para análise do perfil do usuário: como aplicação de questionário de perfil de uso, análise de dados estatísticos e criação de persona do usuário. Para analisar o grau de satisfação do usuário foi feito uma pesquisa com os principais questionários existentes e escolhemos o PSSUQ por ser um questionário que possui maior grau de confiabilidade e que retorna quatro fatores, sendo esses: Satisfação Geral, Qualidade da Interface, qualidade da informação e utilidade do sistema. Também foi escolhido o questionário ASQ que é aplicado depois de cada tarefa executada. Além disso ao aplicar o teste de usabilidade é preciso observar atentamente os passos que os usuário está realizando para concluir cada tarefa.

Aplicamos as técnicas de usabilidade pesquisadas durante o trabalho, em um processo baseado em BDD e TDD, a fim de verificar problemas de usabilidade, e satisfação e uso em um estudo de caso específico, no caso plataforma Noosfero. Para concluir este estudo, finalizaremos o processo de homologação as funcionalidades desenvolvidas (*plugins*) e as mesmas serão disponibilizadas para produção. Além disso, os questionários levantados (PSSUQ e ASQ) serão aplicados ao Participa.Br, para sabermos o grau de satisfação do usuário. Assim, partiremos para a segunda fase do trabalho, que será aplicar o estudo realizado no Portal do Software Público, a fim de verificar a influência de testes automatizados na usabilidade do sistema, buscando responder as hipóteses levantadas no início deste capítulo. Outro passo a ser realizado é verificar os padrões de design e usabilidade adotados pelo Noosfero, propor e implementar possíveis melhorias.

REFERÊNCIAS

[1] A. A. Vicente, "Definição e gerenciamento de métricas de teste no contexto de métodos ágeis," Master's thesis, USP - Universidade de São Paulo, 2010.

[2] A. P. Santos, "Aplicação de práticas de usabilidade ágil em software livre," 2012.

[3] R. M. Stallman, "Free software: Freedom and cooperation," 2001. [Online]. Available: http://www.gnu.org/events/rms-nyu-2001-transcript.txt

[4] S. Nerur, R. Mahapatra, and G. Mangalaraj, *Challenges of migrating to agile methodologies*, 2005.

[5] H. Corbucci, "Métodos ágeis e software livre: um estudo da relação entre estas duas comunidades," Ph.D. dissertation, Universidade de São Paulo, 2011.

[6] M. Potel and S. Cotter, *Inside Taligent Technology*. Taligent Press, 1995.

[7] P. C. Bernardo, "Padrões de testes automatizados," Master's thesis, Instituto de Matemática e Estatística – Universidade de São Paulo, 2011. [Online]. Available: http://www.teses.usp.br/teses/disponiveis/45/45134/tde-02042012-120707/

[8] L. Molinari, *Teste de Software. Produzindo Sistemas Melhores e Mais Confiáveis.*, 2003.

[9] R. C. Martin, "The test bus imperative: Architectures that support automated acceptance testing," 2005. [Online]. Available: http://www.martinfowler.com/ieeeSoftware/testBus.pdf

[10] R. Mugridge and W. dCunningham, *Fit for Developing Software: Framework for Integrated Tests*, 2005.

[11] L. Koskela, *Test Driven: Pratical TDD and Acceptance TDD for Java Developers*. Manning Publications, 2007.

[12] R. Haring, "Behavior driven development: Beter dan test driven development," 2011.

[13] J. Preece, Y. Rogers, and H. Sharp, *Design de Interação: Além do homem computador*, 2007.

[14] D. Norman, *O design do dia-a-dia*. Rocco, 2006. [Online]. Available: http://books.google.com.br/books?id=8zd8PgAACAAJ

[15] t. Lowdermilk, *Design Centrado no Usuário: Um guia para o desenvolvimento de aplicativos amigáveis*, 2013.

[16] D. F. Barbosa, E. S. Furtado, and A. S. Gomes, "Uma estratégia de apoio à institucionalização da usabilidade em ambientes de desenvolvimento ágil," in *Proceedings of the VIII Brazilian Symposium on Human Factors in Computing Systems*. Sociedade Brasileira de Computação, 2008, pp. 214–223.