Melhoria da Qualidade do Processo em Fábricas de Software Open Source através do Personal Software Process

Julio Maravitch Maurício Neto, Alexandre M. L. Vasconcelos, Hermano P. Moura

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7.851 – 50.732-970 – Recife – PE – Brasil

{jmmn, amlv, hermano}@cin.ufpe.br

Abstract. *The Open Source development practices have been the focus of many studies in Software Engineering. In this context, this article intends to identify the main aspects where Personal Software Process can be applied with the objective of attaining the improvement of the development process quality in Open Source environments.*

Resumo. *As práticas de desenvolvimento Open Source têm sido foco da atenção de diversos estudos da Engenharia de Sofware. Nesse contexto, este artigo procura identificar os principais aspectos onde o Personal Software Process pode ser aplicado com objetivo de buscar a melhoria da qualidade do processo de desenvolvimento em ambientes Open Source.*

1. Introdução

Recentemente, o desenvolvimento de softwares *Open Source(OS)* têm atraído uma considerável atenção. As motivações para esse crescente interesse são diversas, incluindo implicações na teoria econômica e direitos de propriedade intelectual [Lessig, 2002]; processos de inovação e criação de conhecimento [Hippel, 2003]; e implicações na prática da engenharia de software como disciplina [Feller, 2002a, b], que constitui o foco desse trabalho.

Formas de adaptar os métodos convencionais de engenharia de software a processos de desenvolvimento não-clássicos têm sido apontadas como uma das mais proeminentes questões de estudo da engenharia de software [Finkelstein, 2000]. A engenharia de software tradicional e a construção de sistemas através de fábricas de software podem aprender bastante com as práticas sociais e técnicas do modelo OS [Crowston, 2004]. A comunidade OS lida com times e práticas diversificados, apresentando pontos em comum com a engenharia de software proprietário especialmente quando os desenvolvedores estão trabalhando com desenvolvimento ágil de software. A combinação destes dois modelos tem o potencial para despontar como uma proeminente alternativa de modelo de desenvolvimento de software [Cavalcanti, 2005] [Fabriks, 2005] [Alvaro, 2004].

Embora as práticas OS tenham adquirido notável sucesso nos últimos anos [Mockus, 2002], existe um grande número de desafios relacionados à qualidade do processo a serem vencidos pelo modelo de desenvolvimento de software OS. Apesar desse fato, alguns projetos atingiram sucesso com um produto final de alta qualidade. Entretanto, até mesmo esses projetos maduros e bem sucedidos enfrentam alguns

problemas de qualidade [Michlmayr, 2004]. Em fábricas de software OS, onde existe interesse comercial, a preocupação com a qualidade do produto é uma necessidade fundamental.

O *Personal Software Process* (PSP) [Humprey, 1996] é um conjunto de guias e melhores práticas para incorporar disciplina ao processo de desenvolvimento de software. Ele provê um *framework* para controlar, gerenciar e melhorar a forma na qual um indivíduo desenvolve software. O presente trabalho tem como objetivo discutir o uso do PSP em fábricas de desenvolvimento OS, como alternativa de melhoria da qualidade do processo, e consequentemente, do produto de software.

Além desta seção introdutória, o trabalho está estruturado em mais cinco seções. A seção 2 fornece uma visão geral sobre o modelo de desenvolvimento OS. A seção 3 descreve a garantia da qualidade no modelo OS. A seção 4 apresenta o PSP e suas diversas fases. A seção 5 apresenta a descrição dos principais aspectos onde o PSP pode contribuir com a melhoria da qualidade de software OS. Por fim, a seção 6 apresenta as conclusões e os trabalhos futuros a serem realizados.

2. Modelo *Open Source* de Desenvolvimento

Stallman propôs uma revolucionária idéia em 1984 [Stallman, 1999] através da *Free Software Foundation*. Sua idéia foi subsequentemente confirmada em 1997 através da *Open source Definition* [Perens, 1999]. O conceito-chave de sua idéia é que deve existir acesso irrestrito aos códigos dos programas de computador: deve ser permitido a qualquer um usá-los, modificá-los e distribuir tais modificações sem custo associado.

A Filosofia básica do software OS é extremamente simples: quando se permite aos programadores trabalhar livremente no código-fonte de um programa, melhorias são realizadas porque o ambiente colaborativo ajuda a corrigir erros e permite a adaptação a diferentes necessidades e plataformas de hardware.

Em seu clássico ensaio, *The Cathedral and the Bazaar* [Raymond, 1999], Eric Raymond descreve o modelo de desenvolvimento de software utilizado por software OS. O modelo Bazaar estabelece que software OS, ao contrário dos softwares tradicionais (Cathedral), deve ser desenvolvido em um ambiente descentralizado sem regras predefinidas. Algumas características estão usualmente presentes no desenvolvimento de software utilizando o Bazaar [Raymond, 1999]: usuários são tratados como có-desenvolvedores, a rápida liberação de entregas, disponibilidade de diversas versões dos produtos, freqüente integração para evitar retrabalho, alto nível de modularização e uma estrutura de tomada de decisões dinâmica.

O modelo Bazaar permite que múltiplos contribuidores possam escrever, testar, ou remover erros do produto em paralelo, o que supostamente acelera o processo de evolução do software. Raymond observa que quanto mais pessoas estiverem olhando para o código, mais defeitos serão encontrados, o que diminui o período de maturação do software [Raymond 2001]. Desta forma, a participação de membros na comunidade representa um papel chave nesse modelo. Os perfis e as motivações que propiciam o comprometimento dos indivíduos também vem sendo objetivo de estudos [Gosh, 2002].

Apesar de o modelo *open source* ser tipicamente visto sob uma ótica não comercial, existem cada vez mais empresas sendo estruturadas a partir deste conceito e,

em curto período de tempo, tais empresas têm acumulado rendimentos consideráveis [Krishnamurthy, 2003]. *Softwares open source* oferecem alternativas baratas e flexíveis quando comparados às soluções comerciais e, deste modo, tornaram-se tão atrativos que até mesmo indústrias que requerem alto grau de qualidade do produto, como as de saúde, estão vendo oportunidades de inovação através do código aberto [Goth, 2005] [Wu, 2004].

3. Garantia da Qualidade no Modelo *Open Source*

A indústria de software vem se esforçando para encontrar formas de desenvolver produtos de software de qualidade. A abordagem de desenvolvimento OS tem ajudado a produzir rapidamente softwares confiáveis, de qualidade e a baixos custos [Mockus 2000].

No modelo tradicional de desenvolvimento de software, a garantia da qualidade é composta por um conjunto de atividades sistemáticas que provêem evidências sobre a habilidade do processo de software de gerar um produto que se adeqüe ao uso [Schulmeyer, 1999]. As atividades realizadas tradicionalmente por grupos de garantia da qualidade têm sido foco de diversas pesquisas [Fagan, 1986] [Porter, 1995] [Frank, 1998] [Rothermel, 1996] [Perry, 1994].

No âmbito do modelo OS, a garantia da qualidade também vem sendo estudada. Em seu trabalho, Zhao [Zhao, 2002]. descreve como a garantia da qualidade do software é realizada no modelo OS, como ela difere dos modelos tradicionais e que diferenças podem ser mapeadas em vantagens práticas na construção de software. Yang foca na identificação dos problemas da garantia da qualidade no modelo OS de forma a propor estratégias de melhoria no processo de desenvolvimento [Yang, 2006]. Groot propõe um sistema para observação da qualidade dos processos empregados na construção de software OS [Groot, 2006]. Através da identificação dos aspectos de qualidade de um determinado projeto, torna-se mais fácil para o usuário escolher uma solução baseada na qualidade do software. A indústria também demonstra atenção na definição de formas de medir a qualidade de software OS, alguns modelos de medição da maturidade de produtos de software vêm sendo utilizados [OpenBRR, 2005] [OSMM, 2005].

As principais diferenças na garantia da qualidade no modelo OS e no modelo tradicional estão relacionadas à disponibilidade do código-fonte e a transparência do processo de desenvolvimento [Groot, 2006]. Devido a estes fatores, a avaliação da qualidade de software OS é comumente mais transparente que nos software tradicionais, ao menos para observadores “externos” ou de terceira parte.

Em termos de processos tradicionais de desenvolvimento como o *Capability Maturity Model* [CMMI, 2006] e *ISO 9001* [Baker, 2001], a grande maioria dos projetos OS encontram-se nas fases iniciais *ad hoc.* Embora algumas áreas de processo (PA), definidas por esses processos sejam menos aplicáveis a software OS (ex: sub-contrato, requisitos, ou gerenciamento e definição), outras PAs utilizadas no modelo OS pertencem a altos níveis de maturidade (ex: gerenciamento de configuração, rastreamento de projeto) [Zhao, 2002].

A falta de documentação, a dependência a indivíduos, dificuldades em colaboração e no gerenciamento de entregas são apontados por Yang como os principais

problemas de qualidade no modelo OS [Yang, 2006]. A participação participantes no projeto não pode ser garantida, tornando-se um problema crítico devido à natureza voluntária do modelo OS [Michlmayr, 2003]. A maioria dos software OS são projetados e desenvolvidos por pequenos times ou um único desenvolvedor, ao invés de um grande grupo [Mockus, 2002]. O projeto torna-se então bastante dependente de alguns indivíduos e do grau de qualidade de seu trabalho.

4. *Personal Software Process*

Através do uso do PSP, os desenvolvedores planejam suas atividades e mensuram o esforço realizado em cada etapa do trabalho, enquanto ainda o realizam. Quando finalizado o trabalho, as medidas recolhidas são analizadas com o objetivo de melhorar a performance do desenvolvedor [Humprey, 1994]. O PSP é ensinado através de um curso [Humprey, 1996] no qual são realizados 10 exercícios de programação. Nesse curso, é demonstrado como aplicar os métodos de gerenciamento da qualidade e do processo quando na elaboração de programas. Após mais de uma década de sua concepção, existem alguns relatos positivos do seu uso [Ferguson, 1997], relatos de má aderência à indústria [Emam, 1996] [Morisio, 2000] e também estudos de sua aplicação na academia [Prechelt, 2000]. Os principais motivos da relatada má aderência do PSP à indústria são o grande período de treinamento necessário para sua introdução, de 150 a 200 horas, em média [Morisio, 2000]; e a dificuldade de armazenar dados. Esta, por sua vez, pode levar a problemas de qualidade [Johnson, 1999].

Diversos estudos apontam o uso de ferramentas de apoio como meio de tornar o PSP mais eficiente, além de facilitar a aderência do programador ao processo [O`Connor, 2001; Johnson 1998, Morisio 2000]. Nesse contexto, uma grande gama de ferramentas de suporte à utilização do PSP foram propostas e testadas, envolvendo também a automação da coleta de dados [Sison, 2005].

O PSP provê um processo que se sofistica progressivamente através de uma série de níveis que partem do PSP0 até o PSP3. Cada nível é baseado no anterior, sendo incrementado em termos de atividades a serem realizadas e/ou métricas a serem mensuradas. Cada nível do PSP possui um conjunto de formulários e instruções a serem seguidas. Estes formulários permitem a coleta de dados e a computação de métricas. Para cada programa existem dados estimados e reais, os quais ajudam na melhoria das estimativas.

O primeiro nível é o processo base, PSP0. O propósito do PSP0 é determinar o processo atual. Nesta fase, o indivíduo armazena informações sobre o tempo gasto em cada atividade e os defeitos injetados e removidos em um programa. O PSP0.1 é o próximo nível. A partir deste ponto o conceito de padrão de codificação é introduzido, assim como o de melhoria do processo. Aqui, o tamanho dos programas começa a ser mensurado.

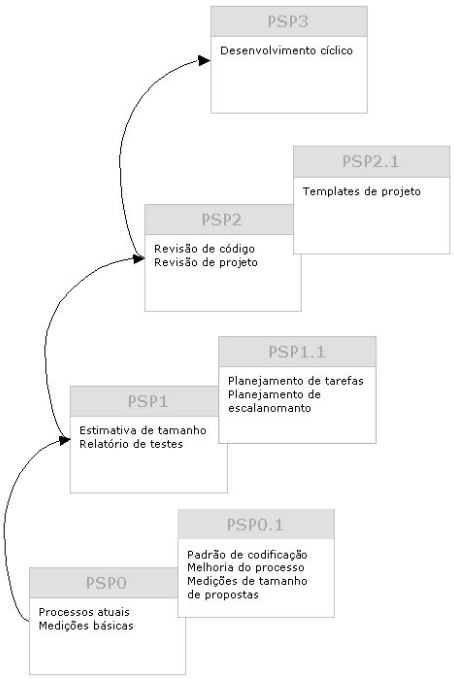

Figura 1. Modelo Incremental do PSP

O PSP1 e PSP1.1 se concentram no planejamento. No nível do PSP1, é introduzida a estimativa do tamanho do programa. O tamanho é estimado para cada projeto e seus limites de controle são calculados. O PSP1.1 agrega planos de atividades e cronograma. Neste ponto, as estimativas são comparadas aos dados reais para a melhoria das estimativas futuras.

Por sua vez, o PSP2 e PSP2.1 focam na qualidade do software desenvolvido. No nível do PSP2, revisões de código e de projeto são introduzidas. Com o objetivo de realizar as revisões de forma completa, listas de tópicos a serem verificados são produzidas com base nos defeitos injetados em todos os programas anteriores. O PSP2.1 inicia o uso de *templates* de projeto. Estes *templates* permitem ao desenvolvedor projetar o programa a partir de quatro pontos de vista diferentes: cenário operacional, especificação funcional, especificação de estados e especificação lógica.

Por fim, o PSP3 apresenta o conceito de desenvolvimento cíclico para grandes programas. Os formulários e procedimentos a seguir são os mesmos do nível PSP2.1, mas o programa é dividido em sub-partes.

5. Utilizando o PSP em Fábricas de software OS

Fábricas de software *open source*, ao contrário do que possa parecer, requerem muito mais organização e processos de gestão de pessoas do que fábricas tradicionais, devido ao fato de *open source* ter como uma de suas premissas fundamentais a colaboração de qualquer indivíduo que assim deseje [Belleza, 2006]. Desse modo, a própria configuração de uma fábrica *open source* é inconstante no tempo e no espaço, dado que tais indivíduos são pessoas que podem aderir ou sair do projeto muito mais frequentemente.

Ainda devido à característica de ser distribuída, uma fábrica de software *open source* pode encontrar problemas, principalmente no que se refere à coordenação de atividades que tendem a acontecer de maneira isolada e assíncrona [Belleza, 2006]. Por outro lado, o modelo distribuído permite aos membros da equipe trabalhar onde melhor lhe aprouver e, até certo ponto, organizarem suas próprias agendas, de modo que o trabalho conjunto possa ser bem-sucedido [Harris, 2006].

Em seu estudo, Gosh [Gosh, 2002] constatou que o desejo de um indivíduo em colaborar com uma comunidade OS pode ter como fundamento diversas razões: preferências por aspectos técnicos do software; convicções políticas, como o sonho de mudar a maneira cuja sociedade e economia lidam com software; aspectos sociais, como troca de informações e interesses com outros participantes da comunidade; o desejo por auto-realização; obtenção de lucro financeiro, entre outras. Entretanto, comparando os motivos para iniciar o desenvolvimento OS e os motivos para continuar com o mesmo, seus estudos revelam que a principal motivação inicial envolve a melhoria das habilidades pessoais, mas que com o passar do tempo e o amadurecimento da comunidade como um todo, convergem para aspectos comerciais e políticos. Além disso, sua pesquisa aponta a vontade de aprender e compartilhar conhecimento como elementos de destaque com relação às expectativas dos participantes da comunidade com relação aos outros.

Neste âmbito, o PSP mostra-se um forte aliado, pois permite a melhoria das habilidades pessoais do indivíduo de diversas formas [Humprey, 1994]. A definição, medição e rastreamento do esforço realizado possibilitam ter uma noção mais precisa da performance pessoal e identificar pontos fortes e fracos a serem trabalhados, objetivando melhores resultados. Além disso, ele propicia um ambiente no qual a troca de conhecimento flui de maneira natural. Em uma comunidade de desenvolvimento, quando cada membro possui um processo pessoal disciplinado, o grupo se comporta de forma diferente. Eles conhecem as habilidades uns dos outros e podem manter o foco em realizar o melhor trabalho e interagir com os outros de forma a ajudá-los [Humprey, 1994]. Além disso, em fábricas de software OS, indivíduos que apresentem um bom nível de colaboração na comunidade podem vir a ter seu trabalho remunerado. Tal fato, juntamente com os elementos supracitados, auxilia a manter motivado o indivíduo durante as diversas fases de anseios pessoais no desenvolvimento OS.

Uma vez que métodos de estimativas tradicionais não se mostram adequadas ao desenvolvimento distribuído, assíncrono e descontínuo realizado nas fábricas software OS, a capacidade de planejamento individual torna-se uma habilidade de suma importância [Cavalcanti, 2005]. A elaboração de boas estimativas de tempo por parte dos desenvolvedores, de forma *bottom-up*, possibilita a obtenção de dados mais acurados e colabora para evitar falhas no gerenciamento de entregas e coordenação de tarefas, apontados como alguns dos mais críticos problemas deste modelo de desenvolvimento [Yang, 2006][Belleza, 2006].

No PSP, medidas de tamanho e esforço são realizadas por cada indivíduo através do método PROBE (Proxy-Based Estimating) [Hayes, 1997]. A partir das estimativas individuais, o gerente de um projeto open source pode elaborar um cronograma de desenvolvimento mais realista e flexível do ponto de vista do desenvolvedor. Além disso, o PROBE fornece indicadores da porcentagem de trabalho já realizado. Através

desses indicadores é possível determinar o status do projeto, estimar seu grau de progresso e definir medidas de controle para garantia da entrega na data de conclusão prevista.

6. Conclusões e Trabalhos Futuros

Através desse estudo foi possível obter uma melhor visão do modelo de desenvolvimento OS, assim como das atividades de garantia da qualidade nele aplicadas. A partir das principais deficiências de qualidade do processo encontradas foi possível buscar alternativas de melhoria. Com o estudo do PSP foram encontrados alguns aspectos dos quais o modelo OS indicativamente pode se beneficiar. Destacam-se, a utilização do PSP como elemento motivador para a adesão e permanência dos indivíduos na comunidade de desenvolvimento e também o uso de estimativas feitas de forma *bottom-up*, pelos desenvolvedores, para elaboração de cronogramas.

Futuramente, devem ser feitos estudos de caso para avaliar empiricamente se há ganho com o uso no PSP em projetos de software OS. Tais estudos devem buscar comparar o número de adesões, desistências e permanências de indivíduos no decorrer do desenvolvimento de um projeto, assim como as razões pelas quais elas ocorreram. Também devem ser validadas as melhorias propostas na elaboração e controle do cronograma de construção do software utilizando a metodologia proposta pelo PSP, do ponto de vista do desenvolvedor, do gerente e do cliente.

Referências

Alvaro, A., Santos, T., Andrade, P., Vasconcelos, J. Albuquerque, J. and Meira, S. (2004) "*Lições Aprendidas na Criação de uma Fábrica de Software Open-Source*", 5º Workshop de Software Livre, WSL'2004, Porto Alegre, RS.

Baker, E. (2001) "*Wich way, SQA?*", IEEE Software 18(1), pp. 16-18

Belleza, C., Levi, F., Ochner, J., Maravitch, J., Vieira, H. and Marcuschi, R. (2006) "*Fases de Criação de uma Fábrica de Software Open Source Distribuída*", Universidade Federal de Pernambuco

Cavalcanti, A., Lucena, L., Lucena, R., Moraes, A., de Fernandes, D., Pereira, S., Albuquerque, J. and Meira, S. (2005) "*Towards an Open Source Software Factory*", In: 2nd Experimental Software Engineering Latin American Workshop, Uberlândia, MG, 2005.

Chrissis, M., Konrad, M., Shrum, S. (2006) "*CMMI: Guidelines for Process Integration and Product Improvement*", Addison-Wesley.

Crowston, K., Annabi, H., Howison, J. and Masango, C. (2004) "*Effective Work Practices for Software Engineering: Free/Libre Open Source Software Development*", WISER.

El Emam, K., Shostak, B. and Madhavji, N. (1996) "*Implementing Concepts from the Personal Software Process in an Industrial Setting*", In: Proceedings of the 4th International Conference of Software Process, Brighton, England

Fabriks (2005) "*An Experience of Modelling and Implementing an Open Source Software Factory Methodology*", SIMS2005 X Simpósio de Informática. SBC – Sociedade Brasileira de Computação, Uruguaiana – RS.

Fagan, M. (1986) "*Advances in Software Inspections*", IEEE Transactions on Software Engineering 12(7), pp. 744-751

Feller, J. and Fitzgerald, B. (2002a) *"Understanding Open Source software Development*", London: Addison-Wesley

Feller, J. and Fitzgerald, B (2002b) "*A further investigation of open source software: community, co-ordination, code quality and security issues*", Information Systems Journal, 12(1), pp. 3-7

Finkelstein, A. and Kramer, J. (2000) "*Software Engineering: A Roadmap*", ICSE.

Frank, P., Hamlet, R., Littlewood, B., Strigini, L. (1998) "*Evaluating Testing Methods by Delivered Reliability*", IEE Transactions on Software Engineering 24(8), pp. 586-601

Ferguson, P., Humprey, W., Khajenoori, S., Macke, S. and Matvya, A. (1997) "*Introducing the Personal Software Process: Three Industry Cases*", IEEE Computer 30(5), pp. 24-31

de Groot, A., Kugler, S., Adams, P.J. and Gousios, G. (2006) "*Call for Quality: Open Source Quality Observation"*, In: IFIP International Federation for Information Processing, Volume 203, Open Source Systems, eds. Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, G., (Boston: Springer), pp. 57-62

Goth, G. (2005) "Open *Source Business Models: Ready for Prime Time*", Disponível em:http://www.computer.org/portal/cms_docs_software/software/content/promo/promo4.pdf. Acessado em: 07/05/2006.

Gosh, R., Glott, R., Krieger, B. and Robles, G. (2002) "*Free/Libre and Open Source Software: Survey and Study*", Part 4: Survey of Developers, International Institute of Infonomics, University of Maastricht

Harris, B. (2006). "*Virtual man chooses Internet over office*", DesMoinesRegister.com.

Hayes, W. and Over, J. (1997) "*The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers*", CMU/SEI-97-TR-001.

von Hippel, E. and von Krogh, G. (2003) "*Open source software and the private-collective innovation model: Issues for organization science*". Organization Science, 14(2), pp. 209-223

Humprey, W. (1994) "*A Discipline for Software Engineering*", Addison-Wesley.

Humprey, W. (1996) "*Introduction to the Personal Software Process*", Addison-Wesley.

Johnson, P. and Disney, M. (1998) "*The Personal Software Process: A Cautionary Case Study*". IEEE Software, 15(6).

Johnson, P. and Disney, M. (1999) "*A Critical Analysis of PSP Data Quality: Results from a Case Study*", Empirical Software Engineering: An International Journal.

Krishnamurthy, S. (2003) "*An Analysis of Open Source Business Models*", University of Washington.

Lessig, L. (2002) "*The Future of Ideas: The Fate of the Commons in a Connected World*", New York: Vintage Books.

Michlmayr, M. and Hill, B. (2003) "*Quality and Reliance on Individuals in Free Software Projects*", In: Proceedings of the 3rd Workshop on Open Source Engineering, Portland, USA: ICSE, pp.105-109.

Michlmayr, M. (2004) "*Managing Volunteer Activity in Free Software Projects*", In: Proceedings of the 2004 USENIX Annual Technical Conference, FREENIX Track, Boston, USA, pp. 93-102

Mockus, A., Fielding, R. and Herbsleb, J. (2000) "*A Case Study of Open Source Software Development: the Apache Server*", In: The 22nd International Conference on Software Engineering, pp. 263-272.

Mockus, A., Fielding, R. and Herbsleb, J, (2002) "*Two Case Study of Open Source Software Development: Apache and Mozilla*", ACM Transactions on Software Engineering and Methodology, Vol. 11, No. 3, pp. 309-346

Morisio M. (2000) "*Applying the PSP in industry*", IEEE Software, 17(6), pp. 90-95

O'Connor, R., Duncan, H., Coleman, G., Morisio, M., McGowan, C., Mercier, C. and Wang, Y., "*Improving Professional Software Skills in Industry - A Training Experiment"*, Technical Report CA-0201, Dublin City University, 2001

OpenBRR (2005) "*Modelo de Levantamento para Avaliação de Preparo para Negócios*", Disponível em: http://www.openbrr.org, Acessado em: 18/02/2007.

OSMM (2005) "*Making the Open Source Ready for the Enterprise*: The Open Source Maturity Model", Disponível em: http://www.navicasoft.com/Newsletters/OSMMWhitepaper.pdf, Acessado em: 18/02/2007.

Perens, B. (1999) "*The open source definition. in Open Sources: Voices from the Open Source Revolution"* C. Dibona, S. Ockman, and M. Stone, Eds., O'Reilly, Sebastopol, Calif., 171–188.

Perry, D., Staudenmayer, P., Votta, L. (1994) "*People, Organizations, And Process Improvment*", IEEE Software 11(4), pp. 36-45

Prechelt, L. and Unger, B. "*An Experiment Measuring the Effect of Personal Software Process (PSP) Training*", IEEE Transactions on Software Engineering

Porter, A., Votta, L. (1995) "*Comparing Detection Methods for Software Requirements Inspections: A Replication Using Professional Subjects*", Empirical Software Engineering: An International Journal 3(4), pp. 355-379

Raymond, E. (1999) "*The Cathedral and the Bazaar*". 1st. O'Reilly & Associates, Inc.

Raymond, E. (2001) "*The Cathedral and the Bazaar. Musgins on Linux and Open Source by an Accidental Revolutionary*", revised edn, Sebastopol, CA: O'Reilly & Associates, Inc.

Rothermel, G., Harrold, M. (1996) "*Analyzing Regression Test Selecion Techniques*", IEEE Transaction on Software Engineering 22(8), pp. 529-551

Schulmeyer, G., McManus, J. (1999) "*Handbook of Software Quality Assurance*", Prentice hall

Sison, R., Diaz, D., Lam, E., Navarro, D. and Navarro, J. (2005) "*Personal Software Process (PSP) Assistant*", In: Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC'05)

Stallman, R. (1999) "*The GNU Operating System and the Free Software Movement*", In: Open Sources: Voices from the Open Source Revolution, O'Reilly, Sebastopol, Calif., 53–70.

Wu, S. (2004) "The Impact of Open Source Software", Disponível em: http://www2.cs.uh.edu/~schwu/nOSS_TermPaper1.pdf. Acessado em: 07/05/2006.

Yang, C. (2006) "*Problems in Quality Assurance under Open Source Development Model*", Freie Universitat, Berlin

Zhao, L. and Elbaum, S. (2003) "*Quality Assurance under the Open Source Development Model*", The Journal of Systems and Software, volume 66, pp. 65-75