

Caracterização de Evolução de projetos de Software Livre através da identificação de atributos de Qualidade de produto

Antonio Soares de Azevedo Terceiro
terceiro@colibre.com.br

1 de dezembro de 2006

1 Introdução

“Software Livre” é uma questão de liberdade, não de preço. Para entender o conceito, você deve pensar em “liberdade de expressão”, não em “cerveja grátis”.

“Software livre” se refere à liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software. Mais precisamente, ele se refere a quatro tipos de liberdade, para os usuários do software:

- A liberdade de executar o programa, para qualquer propósito (liberdade no. 0)
- A liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades (liberdade no. 1). Acesso ao código-fonte é um pré-requisito para esta liberdade.
- A liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo (liberdade no. 2).
- A liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie (liberdade no. 3). Acesso ao código-fonte é um pré-requisito para esta liberdade.

[...]

“Software Livre” Não significa “não-comercial”. Um programa livre deve estar disponível para uso comercial, desenvolvimento comercial, e distribuição comercial. O desenvolvimento comercial de software livre não é incomum; tais softwares livres comerciais são muito importantes.

— Projeto GNU: Definição de Software Livre[oqs00]

O Software Livre hoje está presente em uma grande parte da infra-estrutura computacional em funcionamento. O avanço do Software Livre em estações de trabalho *desktop* também é notável nos últimos anos. Grandes corporações como IBM e Sun têm investido em projetos de Software Livre e inclusive passaram a disponibilizar alguns produtos seus como Software Livre. Diversos projetos de Software Livre desenvolvidos em grande parte por voluntários competem de igual pra igual com softwares não-livres similares.

Esses acontecimentos levantam questionamentos sobre o modelo de produção do Software Livre. Como funciona o gerenciamento de projetos distribuídos onde não há relação formal entre os participantes? Quais são as diferenças em produtos gerados pelo modelo de Software Livre

e de Software Proprietário? O modelo do Software Livre de fato apresenta melhores resultados que o proprietário? Porquê?

Neste trabalho, é proposto um estudo da evolução de projetos de Software Livre. Serão adotadas métricas para indicar o grau de sucesso dos projetos, e através da identificação de atributos de software nos projetos bem e mal-sucedidos, será feito um estudo do impacto positivo ou negativo desses atributos na evolução do projeto.

2 Revisão Bibliográfica

2.1 Qualidade de Software

A disciplina de qualidade de software é um conjunto planejado e sistemático de atividades para garantir que software seja construído com qualidade. Ela consiste de garantia de qualidade de software, controle de qualidade de software, e engenharia de qualidade de software. Como atributo, qualidade de software é (1) o grau ao qual um sistema, componente ou processo atinge os requisitos especificados. (2) O grau ao qual um sistema, componente ou processo atinge as necessidades e expectativas dos clientes ou usuários.

— IEEE 610.12: *IEEE Standard Glossary of Software Engineering Terminology*, (tradução livre)

“Atingir os requisitos” pode ser um objetivo muito ambíguo, principalmente se não houver uma descrição precisa o suficiente desses requisitos. Para que seja possível aferir a qualidade, é preciso que se tenha especificações de requisitos sem ambigüidades, para que durante o desenvolvimento a aderência a esses requisitos possa ser medida. A não-aderência aos requisitos, conhecida como defeito, indica a ausência de qualidade. [Kan02]

Essa medição da qualidade é feita em termos do que se chama atributos de qualidade. A área de pesquisa em Qualidade de Software trabalha tanto atributos de qualidade do processo quanto de qualidade do produto. Neste trabalho, serão utilizados principalmente atributos de qualidade do produto. Estes estão agrupados em dois grupos [DTB05]:

- **Atributos internos.** Os atributos que podem ser aferidos através do código-fonte do software: acoplamento, coesão, tamanho em linhas de código, tamanho funcional, e métricas em geral. [HS96]
- **Atributos externos.** Os atributos que normalmente são percebidos pelos usuários: defeitos (*bugs*)¹, usabilidade de interface, desempenho, satisfação do usuário, etc.

2.2 Evolução de Software

Evolução de Software é um processo contínuo, pelo qual se dá a transformação gradual de um software, idealmente envolvendo a sua melhoria em termos de qualidade. Espera-se que a cada nova versão, um software possua novas funcionalidades, que os defeitos presentes numa versão anterior tenham sido corrigidos, e ele torne-se mais propenso a mudanças com o tempo.

Lehman [LB85] define sistemas *E-type* como softwares que são utilizados ativamente para resolver algum problema de um domínio do mundo real. Em sistemas *E-type*, o principal atributo de qualidade considerado é a satisfação do usuário.

Isso inclui a maioria do software que usamos no cotidiano. Intuitivamente, percebe-se que na medida da passagem do tempo, sistemas desse tipo tendem a carecer de mudanças de forma que possam ser adaptados para uma realidade que também muda constantemente. De fato, Lehman também definiu 8 leis, conhecidas com as leis de Lehman, que descrevem a evolução de sistemas

¹Os defeitos costumam ser medidos através de uma densidade de defeitos, a quantidade de defeitos por cada, digamos, mil linhas de código, ou por cada unidade funcional qualquer.

E-type [LRWP97]. As leis I, II, VI e VII, (veja tabela 1) são de especial interesse no contexto desse trabalho.

I	Mudança contínua	Sistemas <i>E-type</i> devem ser continuamente adaptados, ou tornam-se progressivamente insatisfatórios.
II	Complexidade crescente	À medida que um sistema <i>E-type</i> evolui, sua complexidade aumenta a não ser que seja realizado trabalho para mantê-la ou reduzi-la.
VI	Crescimento contínuo	O conteúdo funcional de um sistema <i>E-type</i> deve ser aumentado continuamente para manter a satisfação dos usuários durante o seu ciclo de vida.
VII	Qualidade decrescente	A qualidade de um sistema <i>E-type</i> vai aparentar estar diminuindo a não ser que sejam rigorosamente mantidos e adaptados para mudanças em seu ambiente operacional.

Tabela 1: Uma parte das leis de Lehman [LRWP97].

Essas leis mostram a importância do estudo e pesquisa em evolução: sendo inevitável para praticamente todo software, há de se lidar de uma forma natural com as mudanças. Esse tema é, então, extremamente trabalhado pela comunidade da área. [BBD00, MFRP06].

Uma proposta recente para lidar com mudanças como parte do processo de desenvolvimento são as metodologias ágeis. O principal exemplo de metodologia ágil atualmente é a Programação Extrema (Extreme Programming) [Bec99], que incorpora diversas técnicas já amplamente utilizadas por projetos de Software Livre, como propriedade coletiva do código, auto-atribuição dos desenvolvedores para tarefas, desenvolvimento orientado a testes, etc.

Refatoração é uma das técnicas componentes da Programação Extrema, e tem por si só uma importância crucial. Fowler et al [FBB⁺99] catalogaram situações comuns de artefatos no código que causam baixa manutenibilidade, extensibilidade, reusabilidade, etc, conhecidas por “*bad smells*”. Refatorações são, portanto, transformações de código com “*bad smells*” em código funcionalmente equivalente mas que implementa a funcionalidade desejada de forma a facilitar manutenção, reuso e extensão.

2.3 Software Livre e Engenharia de Software

O Software Livre vem ganhando atenção crescente da comunidade de Engenharia de Software. Com um modelo de desenvolvimento totalmente diferente do convencional, cujos resultados contradizem a intuição, o Software Livre tem se tornado um objeto de estudo cada vez mais frequente.

Mockus et al [MFH02] analisaram os projetos Apache [apab] e Mozilla [moz], considerando dados sobre quantidade de desenvolvedores e colaboradores, os papéis desenvolvidos por estes, densidade de defeitos (em defeitos/KLOC² e defeitos/Kdelta³), entre outros. Uma conclusão interessantes a que os autores chegaram é de que ambos projetos de Software Livre, quando comparados a projetos de software proprietário⁴ C, D e E, apresentaram menor densidade de defeitos.

Trong e Bieman [DTB05] repetiram o estudo de caso de Mockus et al, aplicando-o ao projeto FreeBSD [fre]. Seus resultados levaram-nos a revisar a hipótese de Mockus et al de que projetos de Software Livre apresentam menor densidade de defeitos que projetos de software proprietário:

²KLOC: milhares de linhas de código

³Kdelta: milhares de alterações, mensurado a partir do repositório das ferramentas de controle de versão utilizadas pelos projetos.

⁴os autores do referido artigo, assim como de diversos outros analisados, utilizam os termos “commercial products” e “commercial projects” em oposição a “open source products” e “open source projects”, com o que o presente autor não concorda: por princípio, tanto a definição de Software Livre [oqs00] quanto a definição de Código Aberto [os] não excluem a possibilidade de exploração comercial do software. Ao contrário: ambas exigem a possibilidade de utilização do software para qualquer fim e por qualquer um.

os autores afirmam que no caso em que o projeto de Software possuir um mecanismo de separar uma versão estável da versão atualmente em desenvolvimento, então a densidade de defeitos das versões estáveis lançadas equivale à de produtos proprietários após o lançamento.

É importante salientar que ambos os estudos, tanto o de Mockus et al quanto o de Trong e Bieman, consideram como premissa que os projetos analisados são “bem-sucedidos”. Uma crítica que os próprios Trong e Bieman fazem aos resultados alcançados é a ausência de um grupo de controle, ou seja, de projetos considerados “mal-sucedidos”. Por outro lado eles também lançam a questão: o que determina o sucesso — ou insucesso — de um projeto de Software Livre? Crowston [CAH03] discute medidas extraídas da literatura, considerações sobre o processo de desenvolvimento de Software Livre e uma análise da opinião de desenvolvedores de Software Livre, que podem ser usadas para aferir o grau de sucesso de projetos de Software Livre.

Outro aspecto que tem impacto na discussão sobre qualidade de projetos de Software Livre é o fato de que grande parte dos desenvolvedores é voluntário. Michlmayr e Hill [MH03] apontam que apesar de ser dito que software é melhor desenvolvido em times pequenos, depuração (testes e identificação de defeitos) é altamente paralelizável. Por outro lado, dizem eles, “depende de um único desenvolvedor cria um ponto de falha que levanta um número de preocupações sérias sobre qualidade e confiabilidade”. Os autores, usando o projeto Debian [deb] como exemplo, apontam a manutenção em times de voluntários, o que diminui o risco da dependência em um único desenvolvedor, como uma saída para lidar com essa questão. Os autores notam também que por outro lado a manutenção em times cria a necessidade de uma estrutura adicional de comunicação entre os desenvolvedores.

Robles et al [RGBMA06] estudaram a evolução das versões estáveis do projeto Debian [deb]. O Debian é um sistema operacional livre, desenvolvido por um grupo de mais de 1000 desenvolvedores e mais alguns milhares de colaboradores. Nesse estudo, Robles et al utilizaram o Software Livre para apontar um outro aspecto da Evolução de Software: ao invés de analisar a evolução de um projeto isoladamente, analisou-se a manutenção de uma compilação com milhares de pacotes de software ⁵. Os resultados incluem: i) o tamanho da distribuição dobra a cada versão estável lançada; ii) por outro lado, o tamanho médio dos pacotes se mantém constante, fato que é explicado pelos autores como consequência da introdução de novos pacotes, ainda jovens, na distribuição; iii) linguagens básicas como C perdem importância relativa com relação à quantidade de código na distribuição (embora continuem aumentando em termos absolutos), enquanto outras linguagens como Perl e Python vêm crescendo com o tempo, tanto em termos relativos quanto absolutos.

Denker e Ducasse [DD06] relatam o processo de evolução de Squeak, uma implementação livre de Smalltalk. O processo de desenvolvimento, com participação de diversos atores com interesses e motivações distintas, traz desafios à comunidade de Squeak. Os autores sugerem que características do modelo de desenvolvimento, e mesmo das linguagens de programação atuais, dificultam uma evolução mais tranquila.

Linux [lin] é um outro projeto que obtém bastante atenção da comunidade de Engenharia de Software. Godfrey e Tu mostraram que o Linux tem crescido linearmente durante muitos anos [GT00]. Schach et al [SJW⁺02] vão além: além de mostrarem o crescimento linear do número de linhas de código, eles apontam uma ameaça à manutenibilidade do Linux: o acoplamento entre funções no código cresce de forma exponencial.

Também tem se trabalhado a questão de análise do processo de software no contexto de Software Livre. Reis e Fortes [RdMF02] publicaram uma análise do processo de desenvolvimento utilizado pelo projeto Mozilla [moz], bem como as ferramentas utilizadas para gerenciar um projeto de tal porte. Reis realizou, ainda, um estudo com diversos projetos de Software Livre, e sistematizou uma caracterização do processo de software para projetos de Software Livre. [Rei03] Bauer e Pizka [BP03] apresentam vários princípios que norteiam o desenvolvimento de software livre, e como eles ajudam a gerenciar os projetos com todas as adversidades inerentes ao seu

⁵Tais compilações são conhecidas como distribuições. Consistem em um sistema operacional básico mais uma quantidade de pacotes, que são componentes que contêm softwares relacionados e podem ser instalados e removidos do sistema através de uma ferramenta de gerência de pacotes fornecida pelo sistema operacional.

contexto: equipes distribuídas, necessidade de formas extras de comunicação, etc.

3 Objetivos

Esta seção descreve os objetivos do trabalho.

3.1 Objetivo Geral

Compreender a evolução de projetos de Software Livre e identificar fatores que influenciam no sucesso ou no insucesso desses projetos.

3.2 Objetivos específicos

- Estudar uma amostra de projetos de Software Livre em termos de atributos de qualidade internos e externos, bem como em termos de outros atributos que possam ser determinados a partir do seu produto (código-fonte) ou que estejam disponíveis de alguma forma pré-existente (por exemplo, como metadados de repositórios de software livre).
- Através da análise de diversas versões desses projetos, comparar as métricas de uma versão para outra e caracterizar a evolução desses projetos.
- Determinar um conjunto de métricas e critérios para avaliar projetos em bem ou mal-sucedidos.
- Identificar características comuns entres os projetos considerados bem-sucedidos.
- Verificar o impacto da introdução dessas características em outros projetos, em termos de melhorias mensuráveis a partir de seus produtos.
- Discutir os atributos relacionados a projetos mal-sucedidos, e verificar se eles estão estabelecidos na comunidade de desenvolvedores de Software Livre, de forma a identificar possíveis ameaças ao modelo de desenvolvimento.

Alguns atributos que podem ser trabalhados são, entre outros:

- Acoplamento
- Coesão
- Quantidade de linhas de código (LOC)
- Densidade de Defeitos
- Utiliza uma arquitetura de software conhecida? (Sim ou Não).
- Utiliza alguma técnica de separação avançada de interesses? (Sim ou Não).
- Número de desenvolvedores.
- Utiliza testes automatizados? (Sim ou Não).

4 Motivação

Software Livre [oqs00] diz respeito à liberdade dos usuários de executar, estudar, modificar e redistribuir um software. O Software Livre, na última década, ganhou uma importância bastante considerável dentro do contexto da indústria de desenvolvimento de software.

Do ponto de vista estratégico, essa importância se evidencia de diversas formas. Companhias de grande porte vêm fazendo investimentos em projetos de Software Livre: A IBM [ibm] investe em diversos projetos, como diversos dos projetos da Fundação Apache [apaa], ou mesmo o Linux [lin], entre outros. A Sun Microsystems [sun] patrocina o desenvolvimento do OpenOffice.org [ooo] após ter liberado o código do seu StartOffice sob uma licença livre, e muito recentemente também liberou a sua implementação da linguagem Java como Software Livre. [sun06].

Do ponto de vista técnico, softwares livres formam hoje a grande parte da infra-estrutura de Tecnologia da Informação em contextos públicos e privados em todas as áreas de atuação. Os processos pelos quais os Software Livres evoluem são de importância crucial para todo um ecossistema que se formou em torno deles.

Analisado sob a luz das teorias tradicionais da Engenharia de Software [Som01], o Software Livre é um fenômeno sem paralelo. Quebra-se o paradigma de equipes locais com desenvolvimento centralizado e processos de desenvolvimento altamente formais. O Software Livre evolui com equipe altamente fragmentadas, onde grande parte dos desenvolvedores trabalha voluntariamente.

O campo de pesquisa *Desenvolvimento de Software Global*⁶ se dedica a estudar os princípios de funcionamento de processos de software distribuídos e descentralizados, incluindo estudos sobre Software Livre. [Spi06a] Além disso, diversos eventos e periódicos internacionais já incorporam a Engenharia de Software no contexto do Software Livre como um dos temas de interesse. [spi06b, sim03, iee02, isj01, wos]

Os trabalhos referenciados na seção 2 possuem limitações no que diz respeito à necessidade de panorama mais abrangente da evolução de projetos de Software Livre. Em particular, muitos dos trabalhos realizam estudos de caso pouco abrangentes [MFH02, DTB05, GT00, SJW+02, DD06].

Robles et al [RGBMA06] realiza um estudo abrangente no que tange a número de projetos, porém trabalha poucos atributos (número de linhas de código e linguagem em que são escritas), o que dificulta uma compreensão da evolução individual de cada projeto.

Desta forma, este trabalho busca estudar de forma mais aprofundada a evolução de projetos de Software Livre levando em conta diversas métricas de qualidade bem como outros atributos dos softwares analisados, além de tentar fornecer evidências que possam ser extrapoladas para um contexto mais amplo.

Serão definidos critérios para avaliar quão bem-sucedido é um projeto, e a partir da identificação de atributos comuns aos projetos considerados bem sucedidos, será possível:

1. Compreender melhor como o modelo de desenvolvimento do Software Livre influencia a evolução de um projeto.
2. Identificar os atributos que se relacionam com o sucesso dos projetos.
3. Com base nos atributos identificados como relacionados ao sucesso de projetos, propor métodos, técnicas e ferramentas de desenvolvimento que possam reproduzir, até certo ponto⁷, o funcionamento dos projetos considerados bem-sucedidos.
4. Discutir potenciais fraquezas do modelo do Software Livre, a partir de atributos relacionados a projetos mal-sucedidos.

⁶Global Software Development – GSD.

⁷Em projetos de Software Livre é bastante importante considerar que uma parte considerável dos desenvolvedores é formada por desenvolvedores independentes voluntários. Desta forma, o sucesso dos projetos também depende muito de fatores característicos desse tipo de relação com o projeto. Estudar a participação de voluntários em projetos de desenvolvimento também é um tópico a ser abordado nesse trabalho.

A escolha de projetos de Software Livre como objeto de estudo justifica-se pela existência de grandes repositórios de projetos publicamente disponíveis. Estes repositórios podem ser os repositórios de uso geral, onde projetos independentes são gestados [sou, sav, ali, ber]; ou os repositórios mantidos por projetos de sistemas operacionais livres, que além de serem uma compilação de uma grande quantidade de pacotes de software, trabalham sobre eles para que se integrem num sistema operacional coerente [deb, gen, fed]. Outra possibilidade é a utilização de repositórios de projetos de grande porte como os ambientes *desktop* GNOME [gno] e KDE [kde], que possuem diversos sub-projetos interdependentes.

5 Metodologia

A seguir estão listadas as etapas inicialmente previstas para o desenvolvimento do trabalho [BSH86].

1. Disciplinas e revisão bibliográfica.

2. Planejamento do experimento.

Nesta etapa será realizado um estudo de preparação do experimento para determinar escopo do estudo, métodos a ser utilizados para a execução do experimento, modelos estatísticos a serem utilizados, etc.

Em especial, algumas atividades serão realizadas durante esta etapa:

- Determinar os atributos a serem considerados na caracterização dos projetos de Software Livre.
- Determinar um tamanho da amostra viável e adequada para o estudo.

3. Operação do experimento.

Esta etapa corresponde à realização do experimento, consistindo de: um período de **preparação**, onde possivelmente será realizado um experimento piloto, com uma amostra reduzida, de forma a validar a fase anterior de planejamento do experimento. um período de **execução**, no qual o experimento final será conduzido. um período de **análise** dos resultados, no qual modelos estatísticos e testes serão realizados sobre os dados obtidos.

4. Interpretação dos resultados obtidos.

Nesta etapa será realizado um estudo que procurará extrapolar os resultados obtidos para aplicação em outros contextos.

Em especial, espera-se:

- Identificar atributos de software que estejam relacionados ao sucesso de projetos.
- Elencar esses atributos de forma que possam ser implantados por projetos de software com características semelhantes.

5. Introdução dos atributos identificados como relacionados a sucesso de projetos em projetos existentes e avaliação dos resultados.

Em especial, uma idéia é estudar o impacto da introdução de práticas identificadas nos projetos bem-sucedidos em outros projetos considerados menos bem-sucedidos e avaliar se essa introdução leva a uma melhoria.

6. Divulgação dos resultados.

7. Redação da Tese.

6 Proposta de cronograma

A tabela 2 apresenta uma proposta de cronograma compreendendo as etapas descritas na seção 5.

Atividade	2007		2008		2009		2010	
	1º sem.	2º sem.	1º sem.	2º sem.	1º sem.	2º sem.	1º sem.	2º sem.
1	•	•						
2			•	•				
3				•	•			
4					•	•		
5						•	•	
6				•	•	•	•	•
7				•	•	•	•	•

Tabela 2: Proposta de Cronograma de atividades.

Referências

- [ali] Alioth. Disponível em <http://alioth.debian.org/>. Último acesso em 28 de Novembro de 2006.
- [apaa] Apache software foundation. Disponível em <http://www.apache.org/>. Último acesso em 22 de Novembro de 2006.
- [apab] The Apache HTTP Server Project. Disponível em <http://httpd.apache.org/>. Último acesso em 30 de Novembro de 2006.
- [BBD00] Elizabeth Burd, Steven Bradley, and John Davey. Studying the process of software change: An analysis of software evolution. In *WCRE '00: Proceedings of the Seventh Working Conference on Reverse Engineering (WCRE'00)*, page 232, Washington, DC, USA, 2000. IEEE Computer Society.
- [Bec99] Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, Cambridge, MA, USA, 1 edition, October 1999.
- [ber] BerliOS – The Open Source Mediator. Disponível em <http://www.berlios.de/>. Último acesso em 28 de Novembro de 2006.
- [BP03] A. Bauer and M. Pizka. The contribution of free software to software evolution. In *IEEE International Workshop on Principles of Software Evolution (IWPSE03)*, 2003.
- [BSH86] V R Basili, R W Selby, and D H Hutchens. Experimentation in software engineering. *IEEE Trans. Softw. Eng.*, 12(7):733–743, 1986.
- [CAH03] Kevin Crowston, Hala Annabi, and James Howison. Defining open source software project success. In *Proc. of International Conference on Information Systems (ICIS 2003)*, 2003.
- [DD06] Marcus Denker and Stéphane Ducasse. Software evolution from the field: an experience report from the Squeak maintainers. In *ERCIM workshop on Software Evolution*, 2006.
- [deb] Debian – Universal Operating System. Disponível em <http://www.debian.org/>. Último acesso em 28 de Novembro de 2006.

- [DTB05] Trung T. Dinh-Trong and James M. Bieman. The FreeBSD Project: A Replication Case of Open Source Development. *IEEE Transactions on Software Engineering*, 31, June 2005.
- [FBB+99] Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, June 1999.
- [fed] Fedora Project. Disponível em <http://fedora.redhat.com/>. Último acesso em 28 de Novembro de 2006.
- [fre] The FreeBSD Project. Disponível em <http://www.freebsd.org/>. Último acesso em 30 de Novembro de 2006.
- [gen] Gentoo linux. Disponível em <http://www.gentoo.org/>. Último acesso em 28 de Novembro de 2006.
- [gno] GNOME: The Free Software Desktop Project. Disponível em <http://www.gnome.org/>. Último acesso em 29 de Novembro de 2006.
- [GT00] Michael W. Godfrey and Qiang Tu. Evolution in open source software: A case study. In *IEEE International Conference on Software Maintenance*, pages 131–142, 2000.
- [HS96] Brian Henderson-Sellers. *Object-oriented metrics: measures of complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.
- [ibm] Internet Business Machines. Disponível em <http://www.ibm.com/>. Último acesso em 22 de Novembro de 2006.
- [iee02] Open source software engineering – special issue of iee proceedings - software, February 2002.
- [isj01] Double Special Issue of ISJ on Open Source – Open Source Software: Investigating the Software Engineering, Psychosocial and Economic Issues – Information Systems Journal, October 2001.
- [Kan02] Stephen H. Kan. *Metrics and Models in Software Quality Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [kde] K Desktop Environment – Conquer your Desktop! Disponível em <http://www.kde.org/>. Último acesso em 29 de Novembro de 2006.
- [LB85] Manny Lehman and Les Belady. *Program Evolution: Processes of Software Change*. London Academic Press, London, 1985.
- [lin] The linux kernel archives. Disponível em <http://www.kernel.org/>. Último acesso em 22 de Novembro de 2006.
- [LRWP97] M.M Lehman, J.F. Ramil, P.D. Wernick, and D.E. Perry. Metrics and laws of software evolution-the nineties view. In *Proceedings of the 4th International Symposium on Software Metrics*, 1997.
- [MFH02] Audris Mockus, Roy T Fielding, and James D Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Trans. Software Engineering and Methodology*, 11(3), 2002.
- [MFRP06] Nazim H. Madhavji, Juan Fernandez-Ramil, and Dewayne Perry. *Software Evolution and Feedback: Theory and Practice*. John Wiley & Sons, 2006.

- [MH03] Martin Michlmayr and Benjamin Mako Hill. Quality and the reliance on individuals in free software projects. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, pages 105–109, Portland, OR, USA, 2003.
- [moz] Mozilla Project. Disponível em <http://www.mozilla.org/>. Último acesso em 30 de Novembro de 2006.
- [ooo] Openoffice.org. Disponível em <http://www.openoffice.org/>. Último acesso em 22 de Novembro de 2006.
- [oqs00] O que é software livre?, 2000. Disponível em <http://www.gnu.org/philosophy/free-sw.pt.html>. Último acesso em 22 de Novembro de 2006.
- [os] The Open Source Definition. Disponível em <http://www.opensource.org/docs/definition.php>. Último acesso em 30 de Novembro de 2006.
- [RdMF02] Christian Robottom Reis and Renata Pontin de Mattos Fortes. An Overview of the Software Engineering Process and Tools in the Mozilla Project. In *Proceedings of the Open Source Software Development Workshop*, pages 155–175,, Newcastle, UK, February 2002.
- [Rei03] Christian Robottom Reis. Caracterizacao de um Processo de Software para Projetos de Software Livre. Master’s thesis, Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, February 2003.
- [RGBMA06] Gregorio Robles, Jesus M. Gonzalez-Barahona, Martin Michlmayr, and Juan Jose Amor. Mining large software compilations over time: Another perspective of software evolution. In *Proceedings of the International Workshop on Mining Software Repositories (MSR 2006)*, Shanghai, China, 2006.
- [sav] Savannah. Disponível em <http://savannah.nongnu.org/>. Último acesso em 28 de Novembro de 2006.
- [sim03] Libre Software: Implications for Organisations – Special issue of Systemes d’Information et Management, 2003.
- [SJW+02] Stephen R. Schach, Bo Jin, David R. Wright, Gillian Z. Heller, and A. Jefferson Offutt. Maintainability of the linux kernel. *IEE Proceedings - Software*, 149(1):18–23, 2002.
- [Som01] Ian Sommerville. *Software Engineering (6th Edition)*. Pearson Addison Wesley, 2001.
- [sou] SourceForge.net. Disponível em <http://sourceforge.net/>. Último acesso em 28 de Novembro de 2006.
- [Spi06a] Diomidis Spinellis. *Code Quality: The Open Source Perspective*. Addison Wesley, 2006.
- [spi06b] Free/Open Source Software Processes – Special issue of Software Process - Improvement and Practice, January 2006.
- [sun] Sun microsystems. Disponível em <http://www.sun.com/>. Último acesso em 22 de Novembro de 2006.
- [sun06] Another freedom for java technology, 2006. Disponível em <http://www.sun.com/software/opensource/java/>. Último acesso em 22 de Novembro de 2006.
- [wos] Open Source Application Spaces: The 5th Workshop on Open Source Software Engineering. Disponível em <http://opensource.ucc.ie/icse2005/>. Último acesso em 24 de Novembro de 2006.