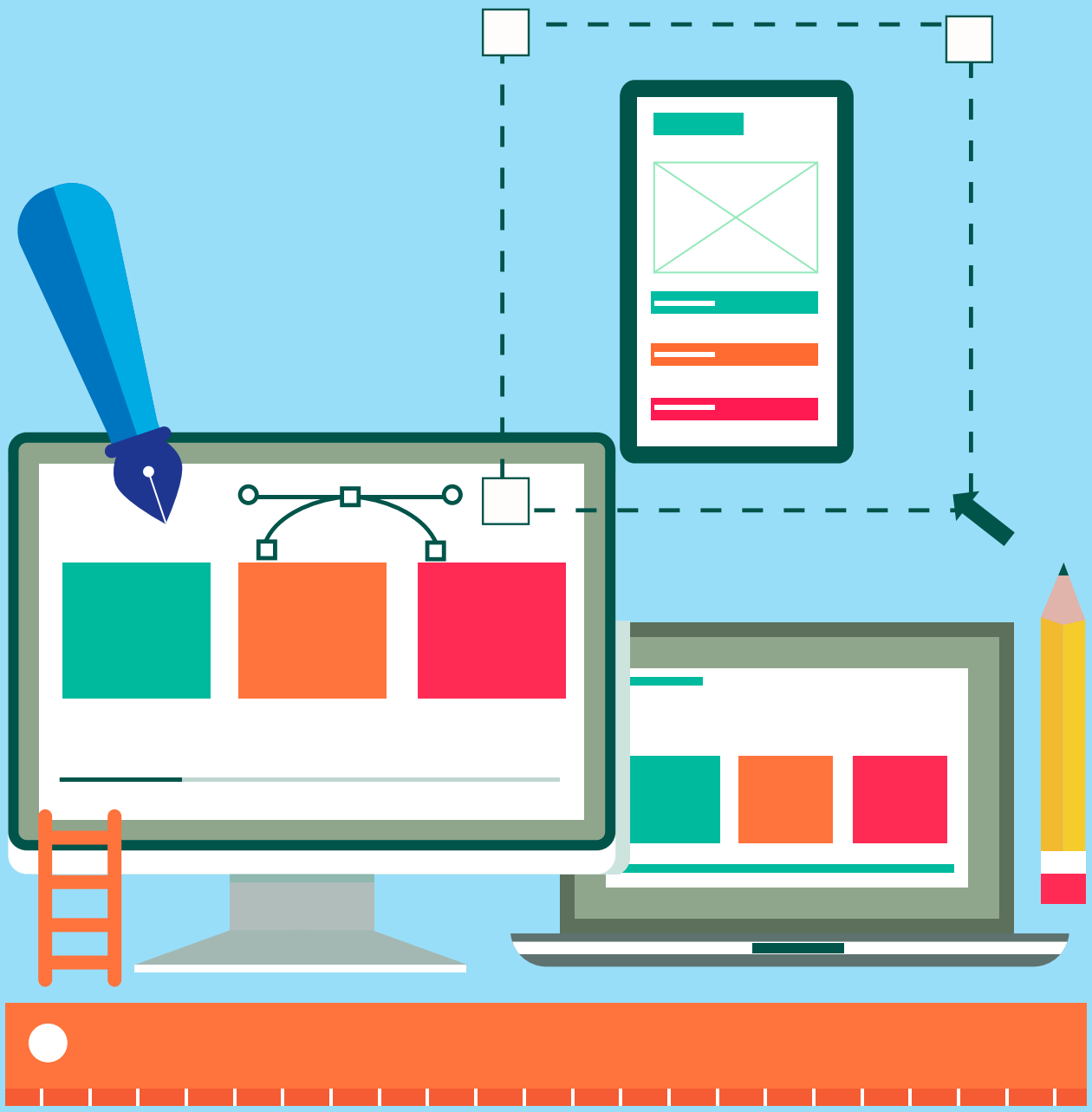
27
02 / 2015

Computação Brasil

Revista da
Sociedade Brasileira
de Computação



SOFTWARE LIVRE

A importância do código aberto na computação do futuro.

SBC
Sociedade Brasileira
de Computação

35 ANOS

Paulo Roberto Freire Cunha
Presidente da Sociedade Brasileira
de Computação

SOFTWARE DO FUTURO

NESTA NOVA EDIÇÃO DA REVISTA COMPUTAÇÃO BRASIL, DAMOS DESTAQUE AO SOFTWARE LIVRE, UM DOS TEMAS MAIS RELEVANTES DA ATUALIDADE NO SEGMENTO DE TECNOLOGIA DE INFORMAÇÃO E COMUNICAÇÃO (TICs).

O Software Livre é hoje um assunto importantíssimo para toda a sociedade, ultrapassando a questão ideológica e tornando-se um ecossistema complexo e de interesse global, que inclui pesquisa científica, educação, tecnologia, segurança, licença de uso e políticas públicas.

Um ponto fundamental é o impacto que ele gera no mercado, atuando hoje como um facilitador para milhares de startups em todo o mundo. Por isso, tem um papel essencial num país como o Brasil, onde o sucesso do empreendedorismo e da inovação tecnológica está diretamente ligado à redução de custos, à criatividade e à flexibilidade.

Para destacar o assunto, a revista traz diversos especialistas que mostrarão as mais recentes análises e pesquisas sobre Software Livre, com o propósito de estimular o leitor a fazer uma reflexão sobre o futuro do desenvolvimento de plataformas com código aberto e suas aplicações em diferentes frentes.

Aproveitando este momento, convidamos os leitores a participarem da próxima edição do Congresso da Sociedade Brasileira de Computação (CSBC), que neste ano acontece em Recife (PE), entre os dias 20 e 23 de julho. É importante destacar que estamos no final da gestão da atual diretoria da SBC e que, em breve, falaremos sobre todos os projetos implantados durante o período em que tivemos a honra de representar a instituição junto à sociedade.

COMO SE ASSOCIAR

Se você deseja renovar a anuidade ou se associar à SBC, confira o valor anual:

Estudante Graduação Básico: R$ 12,00
Estudante Sócio ACM: R$ 42,00
Estudante: R$ 50,00
Efetivo Sócio ACM: R$ 110,00
Efetivo/Fundador: R$ 125,00
Institucional: R$ 760,00
Assinante Institucional C: R$ 1.240,00
Assinante Institucional B: R$ 2.360,00
Assinante Institucional A: R$ 4.220,00

A anuidade da SBC vale pelo ano fiscal (janeiro a dezembro). Sócios da SBMicro têm desconto. Adquira as publicações editadas pela SBC por meio do site www.sbc.org.br.

27
02 / 2015

Computação Brasil

Revista da Sociedade Brasileira de Computação

Caixa Postal 15012
CEP: 91.501-970 – Porto Alegre/RS
Av. Bento Gonçalves, 9.500 - Setor 4 – Prédio 43412 – Sala 219
Bairro Agronomia - CEP: 91.509-900 - Porto Alegre/RS
Fone: (51) 3308.6835 | Fax: (51) 3308.7142
E-mail: comunicacao@sbc.org.br

Projeto Editorial e Execução:
Giornale Comunicação Empresarial
Fone: (51) 3378.7100 - www.giornale.com.br

Índice

JUNHO
23 a 27

XLV DSN
45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2015)
Rio de Janeiro (RJ) • www.ft.unicamp.br/dsn2015

JUNHO
30 A 3

II ELA-ES
II Escola Latino-Americana de Engenharia de Software (ELA-ES 2015)
Porto Alegre (RS) • www.inf.ufrgs.br/elaes2015/

JULHO
20 a 23

CSBC 2015
XXXV Congresso da Sociedade Brasileira de Computação 2015
Recife (PE) • www.cin.ufpe.br/~csbc2015/

AGOSTO
17 a 21

XIV SBQS
XIV Simpósio Brasileiro de Qualidade de Software (SBQS 2015)
Manaus (AM) • sbqs2015.com.br/

AGOSTO
24 a 26

ERAD-RJ

I Escola Regional de Alto Desempenho do Estado do Rio de Janeiro (ERAD-RJ 2015)

Petrópolis (RJ) • eradrj2015.lncc.br

AGOSTO
26 a 29

XXVIII SIBGRAPI

XXVIII Conference on Graphics, Patterns and Images (SIBGRAPI 2015)

Salvador (BA) • sibgrapi2015.dcc.ufba.br/index.html

AGOSTO
27 a 29

IV ENCOSIS

IV Encontro Regional de Computação e Sistemas de Informação (ENCOSIS 2015)

Manaus (AM) • www.encosis.com.br

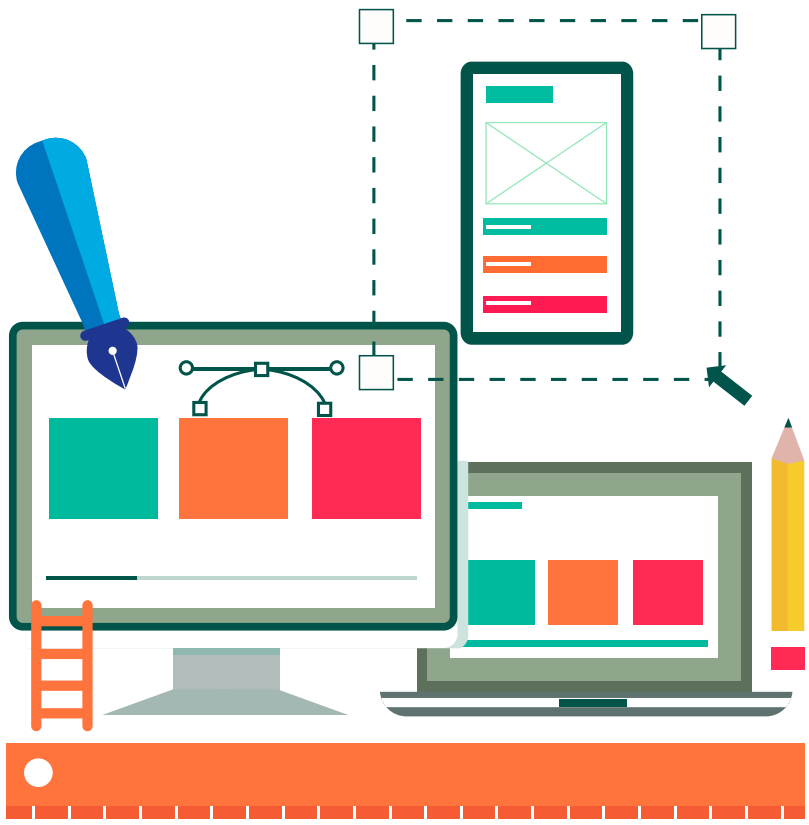
SETEMBRO
21 a 26

VI CBSoft

VI Congresso Brasileiro de Software: Teoria e Prática (CBSoft 2015)

Belo Horizonte (MG) • www.cbsoft2015.dcc.ufmg.br/

AS VÁRIAS FACES DO DO SOFTWARE LIVRE

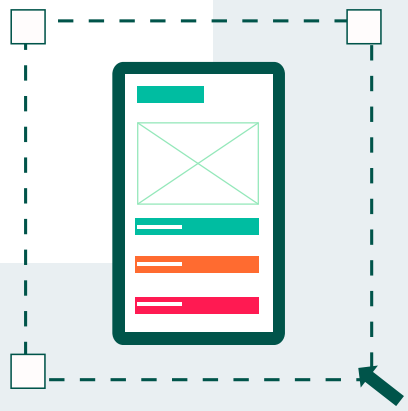


por Nelson Lago e Fabio Kon

QUEM CONHECE UM POUCO sobre o mercado de software sabe da relevância do Software Livre (também chamado de *open source*, ou seja, de código aberto) em diversas áreas de aplicação. Servidores web, compiladores e interpretadores de linguagens populares, bibliotecas, gestores de conteúdo, navegadores e muito mais são hoje territórios em que o Software Livre compete confortavelmente, e muitas vezes com ampla vantagem, com produtos não livres. É claro hoje na indústria de TI que um software livre pode ser tecnicamente excelente e, ao mesmo tempo, oferecer os benefícios de acesso ao código-fonte, independência de fornecedor, custo e risco reduzido etc. De fato, a oferta de um produto sob licença livre está se tornando um diferencial positivo no mercado, como se vê no crescente número de produtos de empresas que tanto são desenvolvidos desde o início como software livre quanto são transformados em Software Livre *a posteriori*.

Essa pujança comercial tem tido frutos positivos também no que diz respeito ao empreendedorismo e à inovação tecnológica, já que o Software Livre tem sido uma grande mola propulsora que tem alavancado milhares de startups em todo o mundo. Graças a ele, hoje em dia é possível montar um sofisticado ambiente de produção de software e entrega de serviços digitais na Internet por meio da nuvem, a um custo baixíssimo e com grande flexibilidade e agilidade.

No entanto, o ecossistema do Software Livre envolve um amplo universo para além do simples "mercado". Como funciona seu processo de desenvolvimento e o que podemos aprender com ele? Qual seu papel no



governo e sua relevância para a cidadania? Como ele pode influenciar o ensino da Computação? Essas são algumas das questões que nortearam a produção deste número especial da Revista Computação Brasil.

Quanto ao processo e às tecnologias de desenvolvimento, Terceiro, Chavez e Meirelles apresentam as técnicas e ferramentas de controle de qualidade comumente usadas em projetos livres, enquanto Gerosa, Wiese, Oliva e Aniche discutem os caminhos abertos pela pesquisa com mineração de repositórios de software e Meirelles, Maia e Terceiro introduzem o novo Portal do Software Público Brasileiro. Mas o Software Livre não existiria sem as suas comunidades; seus aspectos sociais são discutidos por Steinmacher e Gerosa, com um levantamento sobre a entrada de novos participantes em comunidades de desenvolvimento, e por Denner, que sublinha a importância das licenças de software na inserção de profissionais e empresas nessas comunidades. O Software Livre é também um veículo para a cidadania; a relação entre ele e privacidade é enfocada por Speicys, enquanto Aranha expõe seu papel no caminho para as soluções envolvendo sistemas críticos para o cidadão, especificamente a urna eletrônica brasileira. Já a importância social, cultural e econômica do Software Livre leva Amadeu à discussão sobre sua presença central nos possíveis mecanismos de direcionamento nacional rumo

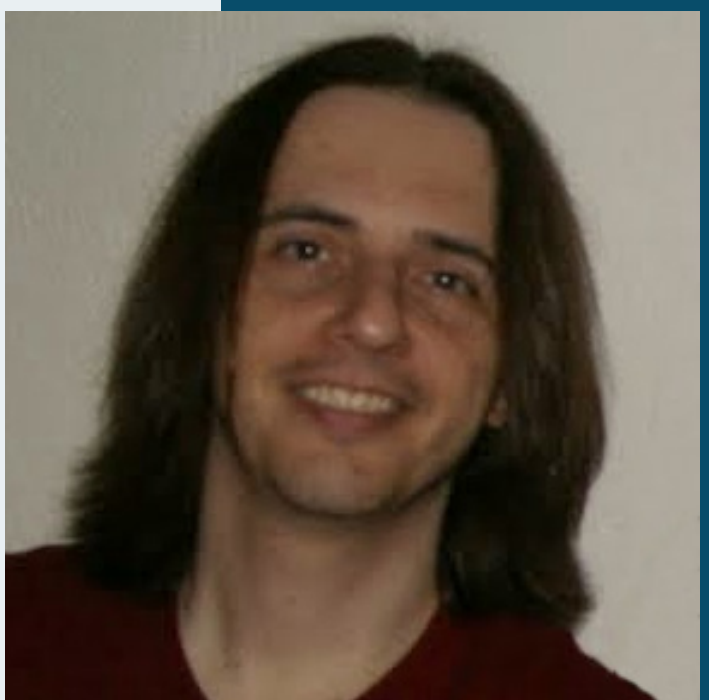
O ecossistema do Software Livre envolve um amplo universo para além do simples mercado.

à excelência e independência tecnológica em termos tanto econômicos quanto políticos. Finalmente, dado que a expansão da computação depende de mais e melhores profissionais da área, Barbosa, Arimoto, Isotani e Maldonado trazem ideias sobre o Software Livre no ensino da Computação.

O Software Livre inspirou e faz parte de um movimento mais amplo pelo compartilhamento aberto do conhecimento. Consequentemente, os artigos deste número da Revista Computação Brasil é disponibilizado sob a licença Creative Commons Internacional com atribuição (CC BY 4.0).

Software, cada vez mais, têm se tornado central na nossa sociedade. Compreender sua dinâmica e, em especial, o posicionamento do Software Livre dentro dela pode abrir o caminho para programas de melhor qualidade com menor custo e, ao mesmo tempo, para uma sociedade melhor. Além disso, no Brasil, o comprometimento do governo com o Software Livre e a sua crescente disseminação abrem as portas para uma maior independência tecnológica e mesmo política do país. É o momento de explorar essas oportunidades.

Boa leitura! ●



NELSON LAGO | Mestre em Ciência da Computação e Gerente Técnico do Centro de Competência em Software Livre do IME-USP. Ministrou diversos cursos e palestras sobre software livre, abordando tanto aspectos técnicos quanto conceituais. Participou do processo de criação da ONG "LinuxSP", onde ofereceu, juntamente com outros voluntários, apoio técnico ao projeto dos telecentros da prefeitura de São Paulo.

FABIO KON | Professor Titular de Ciência da Computação do IME-USP, Editor-Chefe do SpringerOpen Journal of Internet Services and Applications e Coordenador Adjunto de Pesquisa para Inovação da FAPESP. Realiza pesquisas em Sistemas Distribuídos, Engenharia de Software, Empreendedorismo Digital e Software Livre.

PRÁTICAS DE CONTROLE DE QUALIDADE EM PROJETOS DE SOFTWARE LIVRE

por **Antonio Terceiro, Paulo Meirelles e Christina Chavez**

O DESENVOLVIMENTO DESCENTRALIZADO, ABERTO E COM HIERARQUIA FLEXÍVEL TÍPICO DE PROJETOS DE SOFTWARE LIVRE TEM DEMONSTRADO O POTENCIAL DO MODELO PARA A CRIAÇÃO DE SOFTWARE DE ALTA QUALIDADE. AINDA ASSIM, AS COMUNIDADES ENVOLVIDAS TAMBÉM INVESTEM EM FERRAMENTAS, POLÍTICAS E PROCEDIMENTOS PARA AUXILIAR NO CONTROLE DE QUALIDADE DESSES PROJETOS.

QUEM AINDA NÃO TEM contato cotidiano com projetos de Software Livre provavelmente se questiona quanto à qualidade desses projetos. Como é possível desenvolvedores distribuídos, sem vínculos contratuais entre si e talvez até voluntários, produzirem software de qualidade?

Projetos de software livre são bastante heterogêneos sob vários aspectos, e a qualidade é um deles: um projeto de software livre pode ter um processo de controle de qualidade (normalmente abreviado para QA, do inglês *Quality Assurance*) definido e bastante eficiente, ou pode não demonstrar nenhuma preocupação com o assunto.

O contexto da maioria dos projetos de software livre também é diferente de projetos privados, o que faz com que atividades de QA tenham características um pouco diferentes.

A incerteza na alocação de recursos para QA não é exclusiva do software livre. Em projetos privados, frequentemente atividades de QA são preteridas pela gestão em função de orçamento, prazos ou priorização. Em projetos de software livre, a abundância ou a escassez de atividades de QA pode estar relacionada a diferentes fatores, como tamanho do projeto (projetos maiores necessitam mais de QA), popularidade do projeto (mais participantes tornam mais fácil implementar QA) e criticidade do projeto para organizações formais (empresas ou governos podem alocar recursos próprios para QA em projetos de seu interesse).

O contexto da maioria dos projetos de software livre também é diferente de projetos privados, o que faz com que atividades de QA tenham características um pouco diferentes. Num projeto privado, a equipe de QA pode ter autoridade, delegada pela gerência, para demandar trabalho da equipe de desenvolvimento. Num projeto de software livre, a equipe de QA dificilmente vai poder dizer aos desenvolvedores o que fazer, limitando-se a relatar os problemas que poderão ou não ser resolvidos. Numa empresa, é comum haver

uma equipe dedicada exclusivamente para QA, enquanto que na maioria dos projetos de software livre as pessoas envolvidas com QA provavelmente também desempenham outros papéis no projeto, algo também comum em equipes que adotam métodos ágeis, por exemplo.

Diversas práticas de QA são comuns em projetos de software livre, em especial nos bem-sucedidos. Tais projetos têm êxito *porque* adotam práticas de QA, que tornam possível a participação de colaboradores distribuídos geograficamente. Sua adoção também viabiliza o crescimento sustentável desses projetos, que nascem com um pequeno número de colaboradores e podem se transformar em comunidades com centenas ou milhares de desenvolvedores.

Revisão de código é uma prática bastante difundida na comunidade de software livre. Comunidades tradicionais como a do (kernel) *Linux* e do (servidor web) *Apache HTTPD* difundiram esta prática: nessas comunidades, toda mudança a ser realizada no código-fonte precisa ser revisada por outros desenvolvedores. No caso do Linux, a revisão é feita pelo mantenedor responsável pelo subsistema afetado antes de a mudança ser incorporada. No Apache HTTPD, ocorrem tanto a revisão *a priori* para colaboradores externos ao projeto quanto *a posteriori* para seus membros oficiais.

Com a popularização do *GitHub* e de outras plataformas de desenvolvimento colaborativo "social", esta prática se difundiu bastante, não só em projetos de software livre como também em projetos privados. No GitHub, colaboradores podem criar cópias do repositório (*forks*) de qualquer projeto, realizar modificações e enviar solicitações (*pull requests*) para que as mudanças sejam incorporadas à versão oficial. Os mantenedores do projeto ou outros desenvolvedores podem revisar as mudanças, fazer sugestões e críticas e, eventualmente, incorporá-las.

Testes automatizados são bastante comuns em projetos de software livre, e dificilmente encontraremos um projeto bem-sucedido sem um bom conjunto de testes automatizados. Devido às diversas ferramentas existentes, a prática de incluir testes automatizados em projetos se torna mais comum a cada dia. Uma vez escritos, os testes automatizados funcionam como um mecanismo de segurança durante o processo de manutenção: qualquer mudança que faça os testes falharem produz automaticamente um sinal de alerta para a equipe.

Cada nova versão de um módulo publicado no CPAN (repositório de módulos Perl) é automaticamente testada em diferentes versões da linguagem e sistemas operacionais.

Em projetos com testes automatizados, é comum que os mantenedores solicitem que as contribuições com novas funcionalidades ou conserto de defeitos venham acompanhadas dos testes automatizados correspondentes. Além disso, os testes são utilizados durante o processo de revisão de código: contribuições que fazem os testes falharem sem uma boa justificativa provavelmente não serão aceitas.

Integração contínua é também uma prática importante, e consiste na utilização de um serviço que constantemente executa os testes de um projeto e fornece relatórios com os resultados.

A comunidade *Perl* é um bom exemplo de como a integração contínua pode ser levada a sério. Cada nova versão de um módulo publicado no *CPAN* (repositório de módulos Perl) é automaticamente testada em diferentes versões da linguagem e sistemas operacionais por um cluster distribuído mantido pela comunidade, chamado *CPAN Testers*. Um ponto interessante é que essa bateria de testes é realizada sem necessidade de ação explícita do mantenedor de um módulo, ou seja, os testes associados a um módulo serão executados automaticamente.

Um sistema chamado *Travis CI* (cujos componentes também são software livre) vem se popularizando ao fornecer um serviço gratuito de integração contínua para projetos de Software Livre hospedados no GitHub. Depois de configurado, ele executará os testes do projeto sempre que uma nova mudança ocorrer e para cada novo pull request, adicionando um comentário com os resultados dos testes. O Travis CI possui também uma versão comercial para projetos privados.

Análises estática e dinâmica de código também são bastante utilizadas em projetos de software livre, com o objetivo de identificar potenciais defeitos e garantir conformidade com padrões.

A comunidade *Python*, por exemplo, produziu uma especificação de convenções de codificação, chamada *PEP-8*, que serve como um guia para tornar programas em Python mais legíveis e uniformes entre si. A PEP-8 facilita, entre outras coisas, a colaboração de desenvolvedores em múltiplos projetos com o menor esforço adicional possível. Existem diversas ferramentas que checam código-fonte em relação à PEP-8, entre elas pep8, pyflakes e flake8.

Ferramentas de análise dinâmica são também ubíquas em projetos de software livre: *gdb* (*debugger*), *valgrind* (analisador de desempenho, uso de memória e de condições de contorno) e *strace* (analisador de chamadas de sistema), apenas para citar algumas, são bastante populares entre os desenvolvedores.

Na comunidade *Debian*, que desenvolve um sistema operacional (ou "distribuição Linux") de mesmo nome, existe um documento de política técnica chamado de *Debian Policy*. Esse documento descreve regras e recomendações de como pacotes devem ser organizados e se comportar, de forma que seja possível uma

comunidade com mais de 1.000 desenvolvedores geograficamente distribuídos manter mais de 20.000 pacotes consistentes entre si e integrados ao sistema. Para garantir a qualidade dos pacotes Debian, checando também, mas não apenas a sua conformidade com a Debian Policy, são amplamente usadas tanto ferramentas de análise estática como de análise dinâmica desenvolvidas especialmente para o projeto.

Conclusões: Neste artigo, fizemos uma breve apresentação de algumas das principais práticas de controle de qualidade em software, apresentando uma pequena parte do que é usado em projetos de software livre.

Para estudantes e profissionais, participar de atividades de manutenção e QA em projetos de software livre, certamente, é uma experiência bastante enriquecedora. Desenvolvedores com histórico de colaboração em projetos de software livre têm bastante facilidade para encontrar bons trabalhos em muitas organizações no Brasil e — principalmente — em outros países (trabalhando remotamente ou presencialmente).

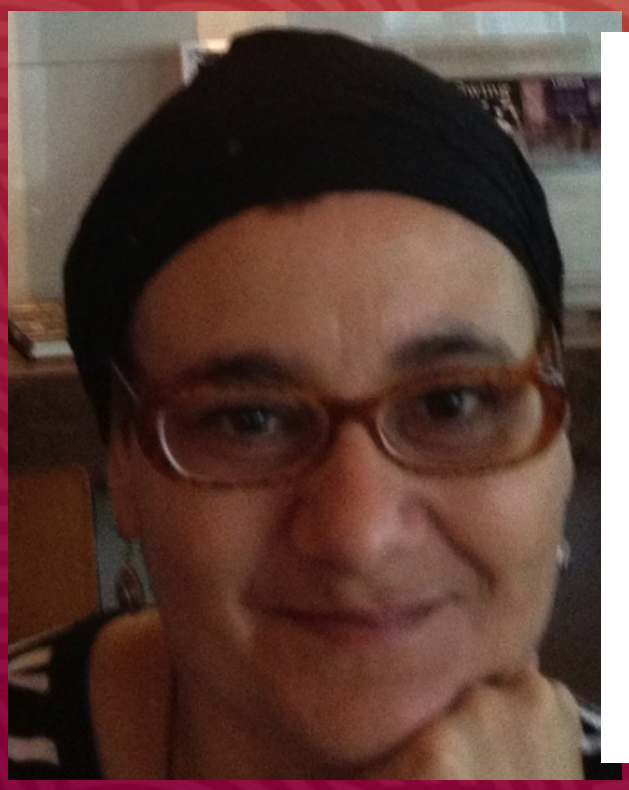
Não só esses profissionais estarão se aperfeiçoando como também poderão trazer práticas e ferramentas de QA validados em projetos de software livre.

Para organizações cujo negócio depende de projetos de software livre, dedicar parte do tempo de seus profissionais para atividades de QA é uma forma de garantir a sustentabilidade desses projetos, e, por consequência, a sua própria.

No caso de organizações desenvolvedoras de software em geral, incentivar seus profissionais a participar de projetos de software livre durante o expediente traz diversos benefícios. Não só esses profissionais estarão se aperfeiçoando como também poderão trazer práticas e ferramentas de QA validadas em projetos de software livre, ou seja, “no mundo real”, para uso em seus projetos privados. ●

ANTONIO TERCEIRO | Sócio fundador da Cooperativa de Tecnologias Livres (COLIVRE) e bolsista do Laboratório Avançado de Produção, Pesquisa e Inovação em Software da Universidade de Brasília (LAPPIS/UNB), atuando como líder técnico no projeto de reformulação do Portal do Software Público Brasileiro. Possui Doutorado em Ciência da Computação pela Universidade Federal da Bahia (2012). É membro do projeto Debian e um dos principais desenvolvedores do projeto Noosfero.

PAULO MEIRELLES | Professor do Bacharelado em Engenharia de Software da UnB. Possui Doutorado em Ciência da Computação pelo IME-USP (2013); é pesquisador-colaborador do Centro de Competência em Software Livre (CCSL) e do Núcleo de Apoio às Pesquisas em Software Livre (NAP-SOL) da USP; coordena a Evolução do Portal do Software Público Brasileiro, no LAPPIS/UnB; e é consultor do projeto Participa.Br.



CHRISTINA CHAVEZ | Professora Associada do Departamento de Ciência da Computação da Universidade Federal da Bahia (UFBA) e Coordenadora do Programa de Pós-Graduação em Ciência da Computação (PGCOMP-UFBA). Possui Doutorado em Informática pela Pontifícia Universidade Católica do Rio de Janeiro (2004). Tem experiência na área de Ciência da Computação, com ênfase em Engenharia de Software, atuando principalmente nos seguintes temas: arquitetura de software (documentação, recuperação e avaliação), evolução de software e educação em engenharia de software.

MINERAÇÃO DE REPOSITÓRIOS DE SOFTWARE LIVRE

por **Marco Aurélio Gerosa, Igor Scaliante Wiese, Gustavo Ansaldi Oliva e Maurício Finavaro Aniche**

A MINERAÇÃO DE REPOSITÓRIOS DE SOFTWARE OFERECE UM AMPLO LEQUE DE INFORMAÇÕES SOBRE PROJETOS DE DESENVOLVIMENTO E TEM SE TORNADO UMA IMPORTANTE ÁREA DE PESQUISA, CONTRIBUINDO PARA A MELHORIA DAS ATIVIDADES QUE ENVOLVEM A CONSTRUÇÃO, MANUTENÇÃO E EVOLUÇÃO DE SISTEMAS DE SOFTWARE.

POR QUE UM GRANDE SUPERMERCADO resolve abrir uma filial na cidade X em vez da cidade Y? As grandes empresas tomam decisões analisando dados históricos coletados de diversas fontes. Projetos de software também têm à disposição uma grande quantidade de dados produzidos durante as atividades de desenvolvimento e de apoio: milhares ou milhões de linhas de código são escritas, defeitos são relatados e resolvidos, discussões técnicas acontecem nas listas de e-mail e gerenciadores de requisitos etc. Analisar tais dados aumenta o entendimento sobre o projeto, os desenvolvedores e os processos que governam a evolução e manutenção do software e possibilita que decisões sejam tomadas de forma mais fundamentada. As análises também possibilitam que aprendamos mais sobre a Engenharia de Software em si.

Nesse contexto, a mineração de repositórios de software é uma área de pesquisa voltada para a recuperação, interligação e análise dos dados históricos produzidos durante o desenvolvimento de software e que estão armazenados em repositórios. Por exemplo, suponha que você é um desenvolvedor e fez uma mudança para uma determinada parte de um sistema. O que mais você tem que mudar? Com base na ideia de que os artefatos que mudaram juntos no passado tendem a mudar juntos no futuro, pesquisadores têm construído ferramentas que auxiliam na propagação de mudanças. Além disso, usa-se também a mineração para entender como o software evoluiu ao longo do tempo, compreender os efeitos da interação social entre os

desenvolvedores no processo de desenvolvimento e prever defeitos e esforço. Há ainda uma série de outras aplicações. Você pode encontrar uma lista mais extensa na principal conferência da área (msrconf.org). A mineração de repositórios de software é um campo de pesquisa relativamente novo e tem atraído o interesse de diversos pesquisadores. Hoje, uma grande parcela dos estudos empíricos em Engenharia de Software envolve algum tipo de mineração de repositórios.

A mineração de repositórios de software é um campo de pesquisa relativamente novo e tem atraído o interesse de diversos pesquisadores.

Projetos de Software Livre e Mineração de Repositórios

Grande parte do sucesso da área de mineração de repositórios deve-se à abundante disponibilização de dados viabilizada pelo movimento de software livre. Projetos de grande visibilidade, como o Linux, a Eclipse IDE e vários outros da Apache Software Foundation e da Free Software Foundation, assim como, mais recentemente, projetos hospedados no GitHub, são frequentemente escolhidos para realização de estudos científicos. O acesso aos dados normalmente é realizado por meio de ferramentas oferecidas por grupos de pesquisa, tais como o Metrics Grimoire (http://metricsgrimoire.github.io/) para coleta de dados de tarefas e modificações de artefatos de diferentes repositórios, ou por meio do uso de APIs oferecidas pelos próprios repositórios, como é o caso do GitHubAPI (https://developer.github.com/v3/). O próprio GitHub oferece acesso a dados de milhões de projetos abertos por meio de um banco de dados BigQuery do Google para facilitar análises de forma aberta (https://www.githubarchive.org/). Vale a pena visitar esse sítio e conferir os ganhadores das competições que têm sido promovidas pelo GitHub e ver bons exemplos de aplicações de mineração de repositórios.

Software Livre tem, portanto, se tornado uma importante fonte de informação e vem contribuindo para a evolução dessa recente linha de pesquisa. Afinal, se os pesquisadores precisam de muitos dados e projetos para conseguir explicar fenômenos da Engenharia de Software, o ecossistema de código aberto se torna uma grande oportunidade para viabilizar essa exploração.

Desafios da Mineração de Repositórios

Ahmed Hassan, no artigo "The Road Ahead for Mining Software Repositories", listou uma série de estudos e desafios enfrentados por pesquisadores de mineração de repositórios. Dentre os desafios, destacam-se a grande quantidade de dados não estruturados e de fontes heterogêneas de informação. Outro ponto de destaque está relacionado à escala. Alinhado ao fenômeno de BigData, estudos atuais têm demandado escalabilidade das técnicas de coleta e análise dos dados para grande quantidade de informações. Outro desafio é facilitar a reprodução das pesquisas realizadas para aplicação em contextos/projetos diferentes.

O desafio é transformar pesquisas empíricas em soluções aplicáveis no dia a dia.

No que diz respeito à aplicação, o assunto tem sido foco de discussões na comunidade. O desafio é transformar pesquisas empíricas em soluções aplicáveis no dia a dia dos engenheiros de software, auxiliando o processo de desenvolvimento do software e no contínuo aumento de qualidade.

Conclusão

Nos últimos anos, a importância da mineração de repositórios tem aumentado com a necessidade das empresas de tornarem-

-se mais orientadas a dados. Gerentes de projetos de software têm interesse em transformar os dados que antes eram somente armazenados nos diferentes repositórios de software em informações relevantes para fundamentar seu processo de tomada de decisão. Do ponto de vista acadêmico, a mineração de repositórios provê dados empíricos para avaliar o uso de técnicas e tecnologias do ponto de vista histórico e vêm sendo cada vez mais usada pelos principais grupos de pesquisa em Engenharia de Software. Dados de projetos de software livre vêm sendo largamente utilizados nesses estudos.

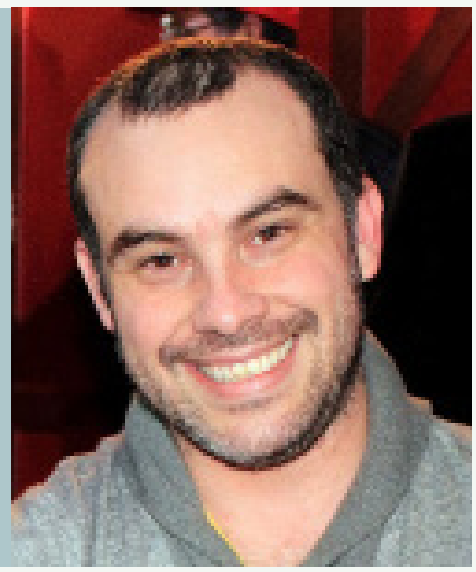
A mineração de repositórios de código veio para ficar e vai ajudar gerentes e desenvolvedores a tomarem decisões no dia a dia.

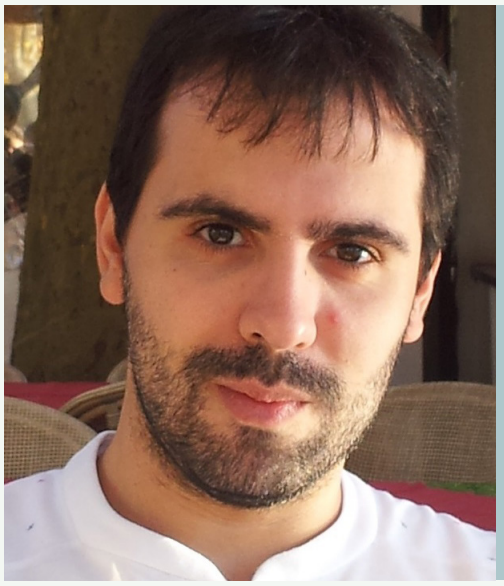
O grupo de pesquisa LAPESSC (http://lapessc.ime.usp.br) do Departamento de Ciência da Computação do IME/USP realiza pesquisas de mineração de repositórios focando em estudos de evolução, manutenção e design de software, dependências de artefatos, métricas de software, análise da colaboração de desenvolvedores e software social. Atualmente o grupo vem desenvolvendo uma ferramenta chamada MetricMiner (http://www.metricminer.org.br), cuja ideia é facilitar a vida de pesquisadores que precisam extrair e ligar informações oriundas de diversos repositórios de código.

Fique de olho: a mineração de repositórios de código veio para ficar e vai ajudar gerentes e desenvolvedores a tomarem decisões no dia a dia e pesquisadores a entenderem melhor a Engenharia de Software, acabando com muitos de seus mitos. ●

MARCO AURÉLIO GEROSA | Professor associado no Departamento de Ciência da Computação da Universidade de São Paulo. Sua pesquisa está na interseção entre Engenharia de Software e Sistemas Colaborativos, focando em engenharia de software experimental, mineração de repositórios de software, evolução de software e dimensões sociais do desenvolvimento de software. Recebe bolsa de produtividade do CNPq e coordena projetos de software livre que já receberam diversos prêmios. Para mais informações, visite www.ime.usp.br/~gerosa.

IGOR SCALIANTE WIESE | Doutorando pela Universidade de São Paulo (IME-USP) sob supervisão do professor Marco Gerosa. É professor do Departamento de Computação da Universidade Tecnológica Federal do Paraná - campus Campo Mourão. Cursou Doutorado-sanduíche na University of California - Irvine sob a orientação do professor David Redmiles. Atua na área de Engenharia de Software, em especial nos temas de mineração de repositórios de software, métricas de software e dependências de software.





GUSTAVO ANSALDI OLIVA | Doutorando pela Universidade de São Paulo (IME-USP) sob supervisão do professor Marco Gerosa. Sua principal pesquisa é em Engenharia de Software, com foco na recuperação, análise e visualização de dependências de software. Gustavo já recebeu bolsas da HP Brasil e da Comissão Europeia. Na indústria, Gustavo trabalhou na IBM Brasil como consultor e desenvolvedor de software por mais de 3 anos. Recentemente, cursou Doutorado-sanduíche na Queen's University sob a supervisão do professor Ahmed Hassan.

MAURÍCIO FINAVARO ANICHE | Doutorando pela Universidade de São Paulo (IME-USP) sob supervisão do professor Marco Gerosa. É também instrutor e desenvolvedor pela Caelum, renomada instituição de ensino brasileira. Atua na área de engenharia de software e suas principais linhas de pesquisa são Test-Driven Development e Métricas de Código. É criador do MetricMiner, ferramenta de código aberto, que ajuda pesquisadores a minerar repositórios.

SOFTWARE PÚBLICO BRASILEIRO:

DE PORTAL PARA PLATAFORMA INTEGRADA DE COLABORAÇÃO

por Paulo Meirelles, Antonio Terceiro e Rodrigo Maia

O SOFTWARE LIVRE SE CARACTERIZA PELA COLABORAÇÃO ENTRE EQUIPES E USUÁRIOS HETEROGÊNEOS; INSPIRADA NESSE MODELO, ESTÁ SENDO CRIADA UMA PLATAFORMA COLABORATIVA PARA A EVOLUÇÃO DO PORTAL DO SOFTWARE PÚBLICO BRASILEIRO, COM MAIS TRANSPARÊNCIA E EFICIÊNCIA NO DESENVOLVIMENTO DE PROJETOS DE SOFTWARE DO GOVERNO.

O GOVERNO FEDERAL VEM NOS ÚLTIMOS ANOS melhorando seus processos de desenvolvimento e adoção de software. Em 2003, a recomendação da adoção de software livre passou a ser uma política, incentivada com a criação do *Guia Livre*. Em 2007, a Secretaria de Logística e Tecnologia da Informação (SLTI) do Ministério do Planejamento, Orçamento e Gestão lançou o portal do *Software Público Brasileiro (SPB)* - um sistema web para o compartilhamento de projetos de software no Governo.

Por um lado, a *Instrução Normativa 04/2012* indica que os gestores devem consultar as soluções existentes no portal do SPB antes de realizar uma contratação de software. Por outro, a evolução técnica do portal do SPB foi comprometida, desde 2009, ao não acompanhar a evolução do seu *framework* base, o *OpenACS*. Não houve o lançamento de novas versões do portal desde então.

Uma nova plataforma para o SPB está sendo desenvolvida pela Universidade de Brasília, através dos seus Laboratórios LAPPIS e MídiaLab em parceria com o Centro de Competência em Software Livre da Universidade de São Paulo (CCSL-USP). A nova plataforma é baseada na integração de ambientes colaborativos, sistemas de controle de versão e de monitoramento da qualidade do código-fonte, e está sendo desenvolvida por uma equipe heterogênea composta por alunos, professores e profissionais, aplicando métodos ágeis e práticas de desenvolvimento distribuído de software.

Evolução para uma plataforma integrada de colaboração

O CONCEITO DE SOFTWARE PÚBLICO se diferencia do de Software Livre em alguns aspectos, destacando-se a atribuição de bem público ao software. Isso significa que o Governo, especificamente o Ministério de Planejamento, assume algumas responsabilidades que garantem ao usuário do software, em especial os órgãos públicos, condições adequadas de uso. Embora haja diferenças entre o que é um software livre e um software público brasileiro, há princípios comuns como a tendência da descentralização na tomada de decisões, do compartilhamento de informações e da retroalimentação. Por isso, a nova plataforma para o SPB foi pensada para contemplar ferramentas que promovam a colaboração e a interação nas comunidades (por gestores, usuários e desenvolvedores) dos projetos, conforme as práticas usadas nas comunidades de software livre. Isso inclui listas de e-mail, fóruns de discussão, *issue trackers*, sistemas de controle de versão e ambientes de rede social.

Para integrar as ferramentas e prover a autenticação única nos serviços da plataforma, um sistema web chamado Colab, que funciona como *proxy* reverso para os ambientes, está sendo evoluído. Em resumo, o Colab oferece a integração de busca, autenticação e apresentação, provendo um único ambiente ao usuário que tem em seu perfil algumas métricas de contribuições (e-mails para listas, inserções em *wikis*, cadastros de *issue* e *commits* nos repositórios).

O Colab foi desenvolvido para o Interlegis (programa do Senado Federal). Por padrão, funciona integrado com o ser-

vidor de listas de e-mail *GNU Mailman* e utiliza o *Apache Lucene Solr* para a indexação dos conteúdos para as buscas. A partir de 2014, as ferramentas GitLab e Noosfero foram integradas ao Colab para compor o novo SPB.

O GitLab é uma plataforma de desenvolvimento colaborativo social integrada ao sistema de controle de versão Git. É o ambiente mais técnico: os repositórios dos projetos do SPB, com páginas *wiki*, *issue tracker* e mecanismos de controle de versão de código estão nele. O Noosfero é uma plataforma para rede social e de economia solidária que contém funcionalidades de gerenciamento de conteúdos (CMS), além de permitir a configuração das páginas de usuários e de comunidades de forma flexível. É o ambiente de maior interação com o usuário do SPB, desde os cadastros até o acesso às páginas dos projetos para *download*, leitura de documentação e contato com os responsáveis.

Em 2015, os perfis das diferentes ferramentas estão sendo integrados de modo que o usuário gerencie seus dados em um único lugar.

A integração dessas ferramentas não está totalmente completa, pois demanda a solução de questões complexas de arquitetura de software. O que foi desenvolvido em 2014 está funcional e já supera o antigo portal do SPB em muitos aspectos. Em 2015, os perfis das diferentes ferramentas estão sendo integrados de modo que o usuário gerencie seus dados em um único lugar. Os controles de acesso e a gestão de permissões também estão evoluindo. O mecanismo de coleta de dados e busca está sendo refatorado para acessar os conteúdos das novas ferramentas integradas ao Colab. Além disso, o Mezuro, um sistema para o monitoramento de métricas de código-fonte, será acoplado ao Colab para fornecer acompa-

Evolução da experiência do usuário

nhamento da qualidade do código dos projetos.

A INTEGRAÇÃO DOS AMBIENTES COLABORATIVOS vai além dos aspectos funcionais. Oferecer à população uma experiência unificada desses ambientes é fundamental para estimular o uso da plataforma, uma vez que reduz a percepção de complexidade.

Assim, a arquitetura da informação está sendo redesenhada para proporcionar uma navegação transparente e que atenda aos diversos tipos de usuário. Os modelos de interação de cada ferramenta estão sendo harmonizados, diminuindo a curva de aprendizado. Ao mesmo tempo, um novo estilo visual está sendo criado para apresentar essa experiência unificada e para atender às diretrizes de Identidade Padrão de Comunicação Digital do Governo Federal.

Os usuários fazem parte do processo. Em 2014, foi aplicado um questionário para avaliar a satisfação das pessoas com o portal antigo e identificar problemas de experiência do usuário. Em 2015, estão previstas pelo menos quatro atividades de validação da nova plataforma com usuários e cidadãos interessados.

Considerações finais

A NOVA PLATAFORMA DO SPB FOI LANÇADA para homologação em dezembro de 2014 e está em uso por algumas comunidades em beta.softwarepublico.gov.br. Todas as ferramentas são software livre e o que está sendo desenvolvido pelas equipes da UnB e USP é publicado em repositórios abertos, disponíveis na versão beta do próprio SPB. Mais

importante do que isso, as melhorias necessárias nas ferramentas utilizadas estão sendo contribuídas de volta para as respectivas comunidades. Isso não só é o certo a se fazer do ponto de vista da comunidade de software livre como vai possibilitar a redução de custos de manutenção no futuro para os cofres públicos e a evolução continuada da plataforma em sinergia com outras organizações que fazem uso das mesmas ferramentas.

Disponibilizar um conjunto de ferramentas e melhorar a experiência do usuário no ambiente é parte desse processo de reformulação do SPB. Aspectos culturais da colaboração em rede para um efetivo uso do que é fornecido na plataforma necessitam ser amadurecidos pelo MP junto às comunidades do SPB. Além disso, a demanda por maior impacto do software público na oferta de software, na adoção das soluções disponibilizadas e na atração de colaboradores e usuários requer intervenção. Um estudo para propostas de licenciamento e seus impactos para o SPB, bem como para sanar as contradições presentes na *Instrução Normativa 01/2011* (que dispõe sobre os procedimentos do SPB), também será conduzido pela UnB para complementar o que está sendo desenvolvido do ponto de vista tecnológico. ●

PAULO MEIRELLES | Professor do Bacharelado em Engenharia de Software da UnB. Possui Doutorado em Ciência da Computação pelo IME-USP (2013); é pesquisador-colaborador do Centro de Competência em Software Livre (CCSL) e do Núcleo de Apoio às Pesquisas em Software Livre (NAP-SOL) da USP; coordena a Evolução do Portal do Software Público Brasileiro, no LAPPIS/UnB; e é consultor do projeto Participa.Br.

ANTONIO TERCEIRO | Sócio fundador da Cooperativa de Tecnologias Livres (COLIVRE) e bolsista do Laboratório Avançado de Produção, Pesquisa e Inovação em Software da Universidade de Brasília (LAPPIS/UnB), atuando como líder técnico no projeto de reformulação do Portal do Software Público Brasileiro. Possui Doutorado em Ciência da Computação pela Universidade Federal da Bahia (2012). É membro do projeto Debian e um dos principais desenvolvedores do projeto Noosfero.

RODRIGO MAIA | É consultor sênior em Design de Experiência de Usuário e bolsista do Laboratório Avançado de Produção, Pesquisa e Inovação em Software da Universidade de Brasília (LAPPIS/UnB), onde lidera as atividades de Design no projeto de reformulação do Portal do Software Público Brasileiro. Possui especialização em Tecnologias Digitais na Prática do Design e da Arte pela Concordia University de Montreal/Canadá (2012).

DEIXEM-ME AJUDAR!

NOVATOS EM PROJETOS DE SOFTWARE LIVRE ENFRENTAM MUITAS BARREIRAS

por Igor Steinmacher e Marco Aurélio Gerosa

IMPORTANTES PROJETOS DE SOFTWARE LIVRE OFERECEM COMUNIDADES ABERTAS, EM QUE É POSSÍVEL CONTRIBUIR, POR EXEMPLO, POR MEIO DO ENVIO DE NOVAS FUNCIONALIDADES OU CORREÇÕES DE DEFEITOS. ENTRETANTO, PARA CONTRIBUIR, HÁ MAIS BARREIRAS DO QUE VOCÊ POSSA IMAGINAR.

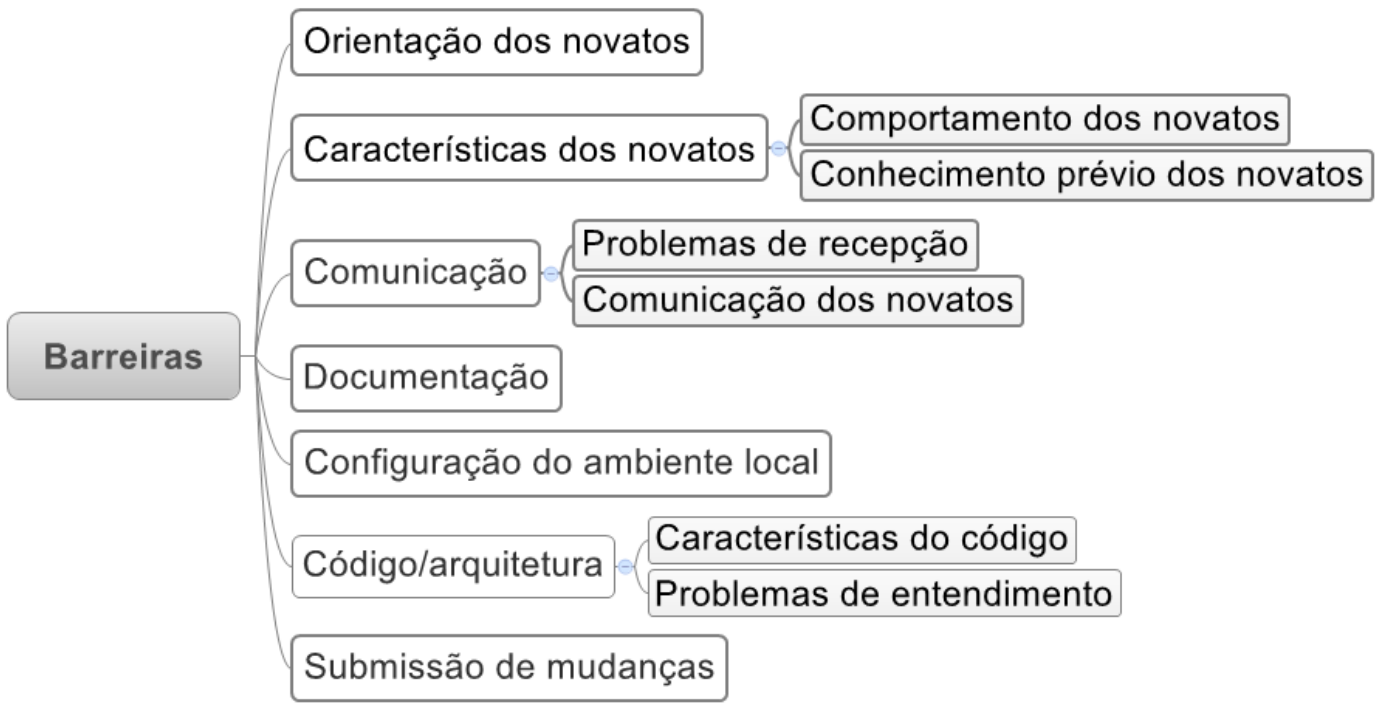
MUITOS PROJETOS DE SOFTWARE LIVRE necessitam do apoio constante de novatos para a sobrevivência, sucesso e continuidade do projeto. Estudos recentes apontam que novatos constituem uma fonte de inovação e são necessários para substituir membros mais antigos que deixam o projeto. Entretanto, novos desenvolvedores enfrentam diversas barreiras para começar a participar, como problemas de recepção, erros de configuração do ambiente de trabalho, curva de aprendizado e documentação incompleta ou inexistente.

Em geral, espera-se que os novatos de projetos de software livre aprendam os aspectos técnicos e sociais dos projetos sozinhos. Sabe-se ainda que os desenvolvedores novatos precisam de habilidades técnicas específicas e que, em geral, finalizar a primeira tarefa em um projeto de software livre é um processo longo com múltiplos passos. Enfrentando barreiras durante todo esse processo, os novatos perdem motivação e, em alguns casos, desistem de contribuir. Como Karl Fogel diz em seu livro *Producing Open Source Software: How to Run a Successful Free Software Project: "Se o projeto não causa uma boa primeira impressão, recém-chegados raramente dão a ele uma segunda chance"*. Essas barreiras para enviar a primeira contribuição afetam não só os interessados em se tornarem desenvolvedores do projeto, mas também aqueles que desejam enviar uma única contribuição, como, por exemplo, estudantes em cursos de computação que necessitam contribuir com projetos como parte de seus cursos ou mesmo profissionais que encontram defeitos ou querem personalizar um produto.

Com um melhor entendimento dessas barreiras, os pesquisadores e as comunidades podem investir seus esforços em criar e melhorar ferramentas e processos.

Para auxiliar os novatos a realizarem sua primeira contribuição, é necessário, primeiramente, que as barreiras enfrentadas nesse período sejam identificadas e entendidas. Com um melhor entendimento dessas barreiras, os pesquisadores e as comunidades podem investir seus esforços em criar e melhorar ferramentas e processos a fim de receber mais contribuições. Em um estudo para identificar as barreiras

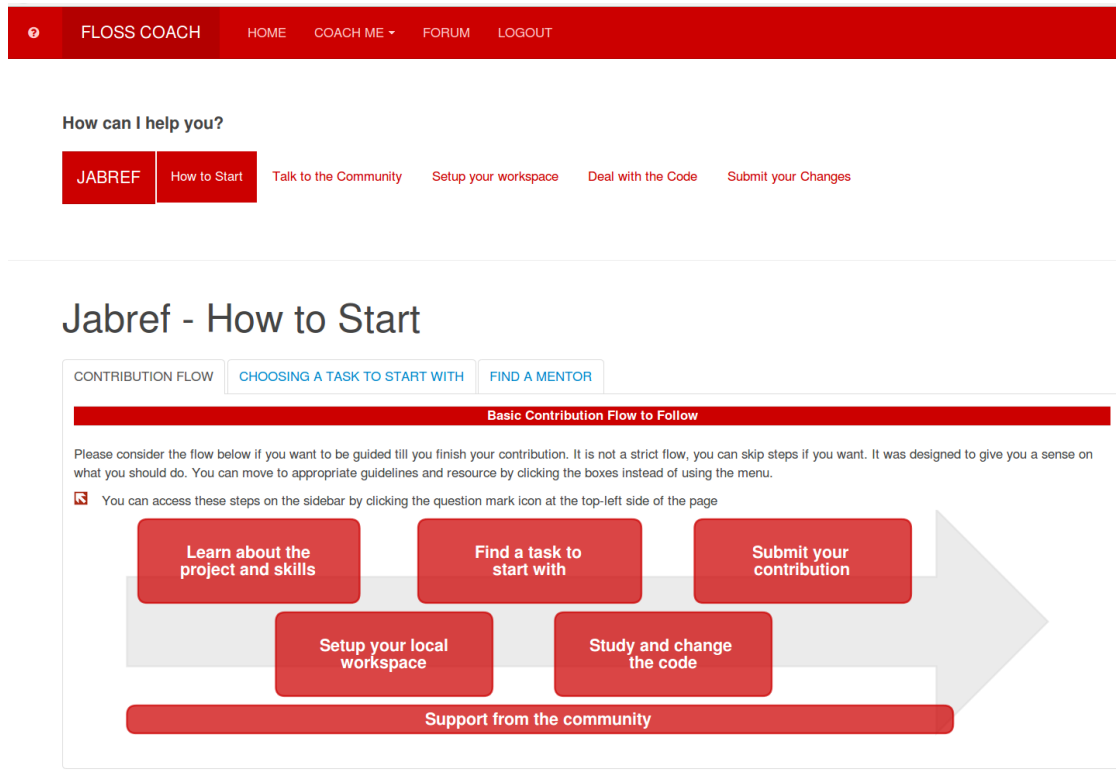
Figura 1 Categorias e subcategorias de barreiras



enfrentadas por novatos, coletamos dados tanto por meio de revisão sistemática da literatura quanto através de entrevistas com membros experientes, novatos que desistiram e novatos que contribuíram para projetos de software livre. A partir da análise desses dados, 58 barreiras foram organizadas em um modelo com sete diferentes categorias: Orientação dos novatos; Características dos novatos; Comunicação; Documentação; Configuração do ambiente local; Código/Arquitetura; Submissão de mudanças. As categorias e suas subcategorias são apresentadas na Figura 1.

Com base nas barreiras identificadas e nas entrevistas realizadas, foi construído o FLOSScoach (www.flosscoach.com), um portal para apoiar os primeiros passos de novatos em projetos de software livre. O portal foi estruturado de maneira a refletir as categorias identificadas no modelo de barreiras criado. Informações já fornecidas e estratégias já existentes nos projetos foram utilizadas e organizadas dentro do portal. Ali, o novato encontra informações sobre as habilidades necessárias para desenvolver para os projetos, o passo a passo para a contribuição, a localização de recursos como repositório de código

Figura 2 Tela da ferramenta FLOSScoach apresentando o fluxo de contribuição sugerido



fonte, gerenciador de tarefas e lista de e-mails, uma lista de tarefas adequadas para novatos (quando fornecidas pelo projeto) e dicas de como se portar frente à comunidade. A interface do portal é ilustrada na Figura 2.

O portal FLOSScoach é mantido por pesquisadores da USP e está disponível para acesso público. Atualmente o portal conta com informações para um pequeno conjunto de projetos de software livre. As informações disponíveis são específicas para cada um dos projetos. Futuramente, o objetivo é minerar automaticamente um número maior de projetos para beneficiar mais novatos e projetos de software livre.

Estudos preliminares mostraram que o FLOSScoach auxilia os novatos, guiando-os em seus primeiros passos e tornando-os mais confiantes para contribuir com os projetos. Além disso, o portal foi considerado útil e de fácil utilização pelos novatos. Apesar de a ferramenta ter auxiliado em algumas categorias de barreiras, os desafios relacionados à redução das barreiras técnicas (por exemplo, problemas de configuração de ambiente local e entendimento do

código) seguem em aberto.

Desse estudo também foram derivadas algumas recomendações aos novatos que desejem contribuir com projetos de Software Livre:

Busque pelos recursos utilizados pelo projeto

Encontre a página oficial, o repositório de código fonte, gerenciador de tarefas e meios de comunicação disponíveis (listas de e-mail, canais de IRC, fóruns).

Seja proativo

Tente resolver os problemas antes de perguntar à comunidade. Buscar alternativas auxilia no aprendizado e demonstrar que tentou antes de perguntar mostra interesse e facilita o primeiro contato.

Não tenha medo da comunidade

Tenha a comunidade como um porto seguro. Sempre que não conseguir resolver os problemas por si ou utilizando os recursos fornecidos pela comunidade, fale com os outros membros utilizando os meios fornecidos.

Envie mensagens simples e claras

Algumas dicas de como enviar uma mensagem: seja gentil, apresente-se mencionando suas habilidades e objetivos, faça suas perguntas de maneira clara e objetiva e mostre que tentou resolver o problema antes de recorrer à comunidade.

Não compre brigas

Quando receber uma mensagem que considere rude ou mal-educada, desvie das respostas e utilize-se do que for importante para você. Se a resposta não resolver seu problema, desculpe-se pelo mal-entendido e refaça sua pergunta educadamente.

Prepare-se para enfrentar os problemas de configuração do ambiente local

Configurar o ambiente local (baixar código, configurar a aplicação e construí-la localmente) é uma atividade dolorosa e demorada. Siga os tutoriais fornecidos e sempre tire dúvidas com a comunidade.

Utilize uma máquina virtual

Para evitar problemas com dependências, incompatibilidade com aplicações já existentes em sua máquina e versões antigas de sistema operacional, utilize uma máquina virtual com um sistema operacional recente instalado.

Encontre uma tarefa fácil

Descubra se o projeto etiqueta as tarefas para identificar aquelas que são adequadas a novatos ("easy hacks", "easy task", "good for newcomers" etc.)

Sempre mantenha a comunidade informada de suas decisões

Se escolher uma tarefa, envie um comentário dizendo que está trabalhando nela; se desistir, faça o mesmo; se conseguiu resolver um problema e precisa que alguém revise o código, informe a comunidade. Tornando seus passos públicos, você está contribuindo para evitar retrabalho.

A identificação e organização das barreiras e a concepção do portal e das orientações para novatos é o primeiro passo para auxiliar a entrada de novatos em projetos de Software Livre. Além dessas ações, espera-se que as comunidades adéquem seus processos, práticas e ferramentas a fim de criarem ambientes mais apropriados para os novatos, por meio da redução das barreiras identificadas. ●

IGOR STEINMACHER | Professor de Ciência da Computação da Universidade Tecnológica Federal do Paraná (UTFPR). Atualmente cursa Doutorado em Ciência da Computação no IME/USP, sendo sua pesquisa vinculada ao Centro de Competência em Software Livre (CCSL). Tem se dedicado a entender aspectos sociais das comunidades de software livre. Seus principais tópicos de interesse são sistemas colaborativos e engenharia de software, com foco principal na entrada de novatos em projetos de software livre.

MARCO AURÉLIO GEROSA | Professor Associado e Livre Docente do Departamento de Ciência da Computação do IME/USP e coordenador do Núcleo de Apoio à Pesquisa em Ambientes Colaborativos na Web (NAWEB). Atua nas áreas de Engenharia de Software e Sistemas Colaborativos e suas interseções. Alguns de seus tópicos de interesse atuais são: mineração de repositórios de software e sistemas colaborativos na Web. Já publicou mais de 150 artigos científicos e sua produção tem fator de impacto H 21.

SOFTWARE LIVRE, MAS NÃO TÃO LEVE E SOLTO

por Carlos Denner Santos Jr.

O SOFTWARE LIVRE SE ORIGINA NO ESFORÇO DE ATIVISTAS PARA SE TORNAR UM FENÔMENO DE SUCESSO TÉCNICO E MERCADOLÓGICO. ESSA TRAJETÓRIA DE PROFISSIONALIZAÇÃO E DIFUSÃO VEIO ACOMPANHADA DO ENVOLVIMENTO DE EMPRESAS E GOVERNOS NOS PROCESSOS DE DESENVOLVIMENTO E COMERCIALIZAÇÃO, O QUE TORNOU O SIGNIFICADO ESPECÍFICO DO QUE É PERMITIDO FAZER COM UM SOFTWARE LIVRE DEPENDENTE DE SUA LICENÇA.

A produção de software livre foi inicialmente concebida como uma alternativa ao modelo proprietário tradicional, onde o uso do aplicativo era vendido sob os termos de uma licença. Isso apesar de Eric Raymond dizer em seu ensaio clássico, A Catedral e o Bazar[1], que originalmente software era desenvolvido de forma colaborativa e não comercial, aberta, fazendo assim um chamado de retorno às origens direcionado aos desenvolvedores, que podiam se organizar de forma geograficamente dispersa e voluntária pela Internet. Desde "o chamado", software caracterizado como livre tem sido produzido por organizações importantes no mercado de software mundial e se tornado uma convivência não tão conveniente mas obrigatória para os aplicativos proprietários concorrentes.

Os termos que regulam essas permissões variam de acordo com a licença específica que o autor do aplicativo adotou.

Basicamente, um software pode ser considerado livre quando sua licença permitir, gratuitamente, sua execução para qualquer propósito, o acesso ao código-fonte modificável e a redistribuição do aplicativo, contendo ou não aperfeiçoamentos incorporados[2]. No entanto, os termos que regulam essas permissões variam de acordo com a licença específica que o autor do aplicativo adotou. No começo, referia-se de forma metonímica a software livre como aquele licenciado sob a General Public License (GPL[3]). Isso era suficiente para os usuários e desenvolvedores sem interesses comerciais, buscando apenas uma solução de qualidade gratuita ou cooperar nas tarefas de desenvolvimento, pois todas as permissões requeridas estavam ali contempladas. Assim, a GPL se espalhou de forma viral, obtendo os benefícios de ter sido a opção padrão para os liberadores de software mais pioneiros e ativistas.

Mas a consagração do software livre como alternativa viável a aplicativos dominantes, como é o caso do sistema operacional Linux/Android em contraste ao Windows/iOS, e o consequente envolvimento de grandes corporações com preocupações

mercadológicas e judiciais elevaram a questão do licenciamento a um outro patamar de importância e complexidade. As empresas e seus departamento jurídicos, além dos leitores atentos, rapidamente perceberam que as permissões de uso, modificação e redistribuição necessárias para um software ser chamado de livre eram reguladas por sua licença. Um software GPL, por exemplo, pode ser modificado e redistribuído sim, mas sua redistribuição deve vir acompanhada do código-fonte referente à modificação realizada, o que desagrada àqueles que buscam vantagem competitiva baseada no direito exclusivo a essa modificação.

De forma oportunista, inclusive testando a capacidade jurídica-litigiosa de fazer cumprir os termos das licenças de software livre dos autores de aplicativos, empresas famosas (e possivelmente pessoas desconhecidas) violaram obrigações como a de redistribuição com o código-fonte do trabalho derivado de originais sob a GPL, apropriando-se de propriedade pública. O contrário também ocorreu, com o uso indevido de código-fonte proprietário em software livre[4] , fazendo surgir iniciativas de auditoria e reformulação do kernel do Linux[5].

Um software GPL, por exemplo, pode ser modificado e redistribuído sim, mas sua redistribuição deve vir acompanhada do código-fonte referente à modificação realizada.

Dentre os casos de empresas que foram identificadas infringindo os termos da GPL, talvez o mais famoso seja o da Free Software Foundation (FSF) contra a Cisco e a Broadcom. Segundo a revista Forbes[6] , a Linksys, umas das empresas da Cisco na época, produzia e comercializava um roteador que utilizava Linux, sistema operacional licenciado sob a GPL. Ao vender roteadores controlados por Linux, a Linksys redistribuía um software GPL no mercado, mas sem disponibilizar em conjunto o código-fonte correspondente. A descoberta de que a Linksys fazia isso motivou a FSF a tomar iniciativas legais de

fazer cumprir aquilo que a GPL exigia, pressionando a Cisco a disponibilizar o código-fonte utilizado em seus roteadores, o que foi atendido posteriormente.

Essas iniciativas institucionais de fazer cumprir os termos de licenças de software já foram realizadas centenas de vezes hoje em dia, sendo uma preocupação central das organizações promotoras de software livre, como a FSF e a Open Source Iniciative (OSI[7]). Esse cenário demonstra o alto envolvimento de atores importantes no mercado de tecnologia da informação em software livre, e ajuda a entender a diversidade de licenças livres que foram criadas visando atender melhor os interesses desse ecossistema cada vez mais variado e integrado à competição mercadológica, às cortes judiciais e ao cotidiano da população digitalmente ativa.

Atualmente, dezenas de licenças livres estão disponíveis para os autores de software[8] . Para facilitar o entendimento dessas licenças, tem sido comum o agrupamento delas por grau de restrição[9]. São duas as restrições principais em uso em licenças livres: a primeira é a que exige que trabalhos derivados daquele código-fonte sejam licenciados com os mesmos termos da licença original. Essa restrição é a que deu origem ao termo copyleft, sendo uma característica que, segundo seus defensores, contribui para a sustentabilidade do modelo livre de desenvolvimento de software. A segunda restrição é a que proíbe a combinação durante a compilação de códigos-fonte com licenças que imponham restrições adicionais, o que culmina por gerar incompatibilidades entre algumas licenças livres.

Licenças que possuem as duas restrições são chamadas de recíprocas totais, pois exigem que tudo ao redor do software esteja licenciado da mesma maneira. O exemplo mais popular desse tipo de licença é a GPL[10], sendo então uma representante do grupo mais restritivo de licenciamento de software livre. Mas existem licenças apenas com a restrição referente a trabalhos derivados, não tendo a segunda restrição descrita, as recíprocas parciais (e.g.,

LGPL). Além dessas, há também as que não fazem nenhuma dessas restrições, as chamadas licenças acadêmicas ou comerciais (e.g., MIT). A decisão de qual licença adotar é de responsabilidade dos detentores dos direitos autorais do aplicativo, e os impactos dessa decisão no sucesso do projeto em atrair colaboradores, individuais ou organizacionais, para compartilhar os custos de desenvolvimento são conhecidos[11].

Sabendo das restrições impostas pelas licenças livres e da variedade de interesses e atores envolvidos nos ecossistemas desses projetos, não é difícil imaginar que a escolha da licença funciona como incentivo, ou o contrário, para essas pessoas, físicas ou jurídicas, que se envolvem e colaboram, em maior ou menor grau, no desenvolvimento de um determinado software livre. A atratividade do projeto é influenciada, além de outras coisas, pelas restrições da licença de seu software[12]. Isso porque o que se pode fazer com um software livre, especialmente em termos comerciais, depende de sua licença. O software é livre, mas você não.

NOTAS:

1 http://www.dominiopublico.gov.br/pesquisa/DetalheObraForm.do?select_action&co_obra=8679
2 http://www.sciencedirect.com/science/article/pii/S0963868712000340
3 http://www.gnu.org/copyleft/gpl.html
4 A SCO processou a IBM por utilizar código-fonte do Unix no kernel do Linux.
5 http://www.gnu.org/distros/free-distros.html
6 http://www.forbes.com/2003/10/14/cz_dl_1014linksys.html
7 http://opensource.org/
8 http://www.gnu.org/licenses/license-list.html
9 http://ccsl.ime.usp.br/files/relatorio-licencas.pdf
10 Na verdade, hoje em dia, a GPL representa uma família de licenças, pois várias versões e aprimoramentos dessa licença já foram desenvolvidos (GPLv2, GPLv3, etc.).
11 http://dl.acm.org/citation.cfm?id=1958950
12 http://www.scielo.br/pdf/rae/v50n4/07.pdf

CARLOS DENNER | É professor da Universidade de Brasília e trabalha na interseção entre administração estratégica, sistemas de informação e computação. Estuda a dinâmica de competitividade entre organizações na sociedade digital, o desenvolvimento de software (livre), as questões de propriedade intelectual e a aceitação e/ou rejeição de tecnologias e inovações.

PRIVACIDADE, SOFTWARE LIVRE E A ERA PÓS-PC

por Roberto Speicys Cardoso

DURANTE MUITOS ANOS O SOFTWARE LIVRE FOI UMA DAS PRINCIPAIS FERRAMENTAS PARA PROTEGER A PRIVACIDADE DOS USUÁRIOS DE COMPUTADOR. A RECENTE EVOLUÇÃO TECNOLÓGICA, ENTRETANTO, CORROEU MUITAS DESSAS PROTEÇÕES. É CHEGADA A HORA DE A COMUNIDADE CIENTÍFICA E A SOCIEDADE SE EMPENHAREM E LEVAREM O SOFTWARE LIVRE A UM OUTRO NÍVEL.

As revelações sobre o aparato de vigilância norte-americano feitas por Julian Assange e Edward Snowden mostraram que os modelos de adversário utilizados pela pesquisa em privacidade até muito recentemente eram exageradamente otimistas: não só o governo norte-americano vigia em massa as comunicações de cidadãos americanos e de outros países, como alguns dos mecanismos (como os "grampos" no interior dos datacenters de grandes empresas de tecnologia como o Yahoo e o Google) surpreenderam até mesmo a tradicionalmente cética comunidade de pesquisa. Ainda que as informações disponíveis hoje em dia só permitam afirmar que um tal aparato de vigilância é operado pelo governo dos Estados Unidos, é razoável supor que outros países utilizem mecanismos de vigilância similares em menor escala.

É um fato, portanto, que as comunicações de milhares de cidadãos são monitoradas sem um motivo razoável ou autorização específica da justiça, em violação a diversas leis, dentre as quais o Marco Civil Brasileiro, e à Declaração Universal dos Direitos Humanos. Para entendermos como evitar que essas invasões continuem a acontecer, entretanto, é necessário compreendermos as razões que nos trouxeram até aqui. E elas são principalmente duas: (i) a volta de uma arquitetura de sistemas centralizada acessada por terminais relativamente "burros"; e (ii) um modelo de negócios cada vez mais popular baseado na erosão da privacidade de seus usuários.

Na era do PC, a arquitetura de sistemas mais popular era extremamente distribuída: cada usuário possuía um computador relativamente poderoso, capaz de executar localmente as aplicações necessárias para um uso profissional ou de lazer, eventualmente

trocando dados com outros computadores em uma rede local ou através da Internet (Figura 1). Essa arquitetura possuía diversas características que favoreciam a proteção da privacidade: a maior parte dos dados do usuário era armazenada em sua própria máquina e a comunicação com a rede era opcional; o software não livre era muito comum, mas o movimento pelo software livre se empenhava em produzir as ferramentas para eliminar essa limitação, garantindo que as informações fossem utilizadas para a finalidade específica do software em execução e não coletadas para outros fins.

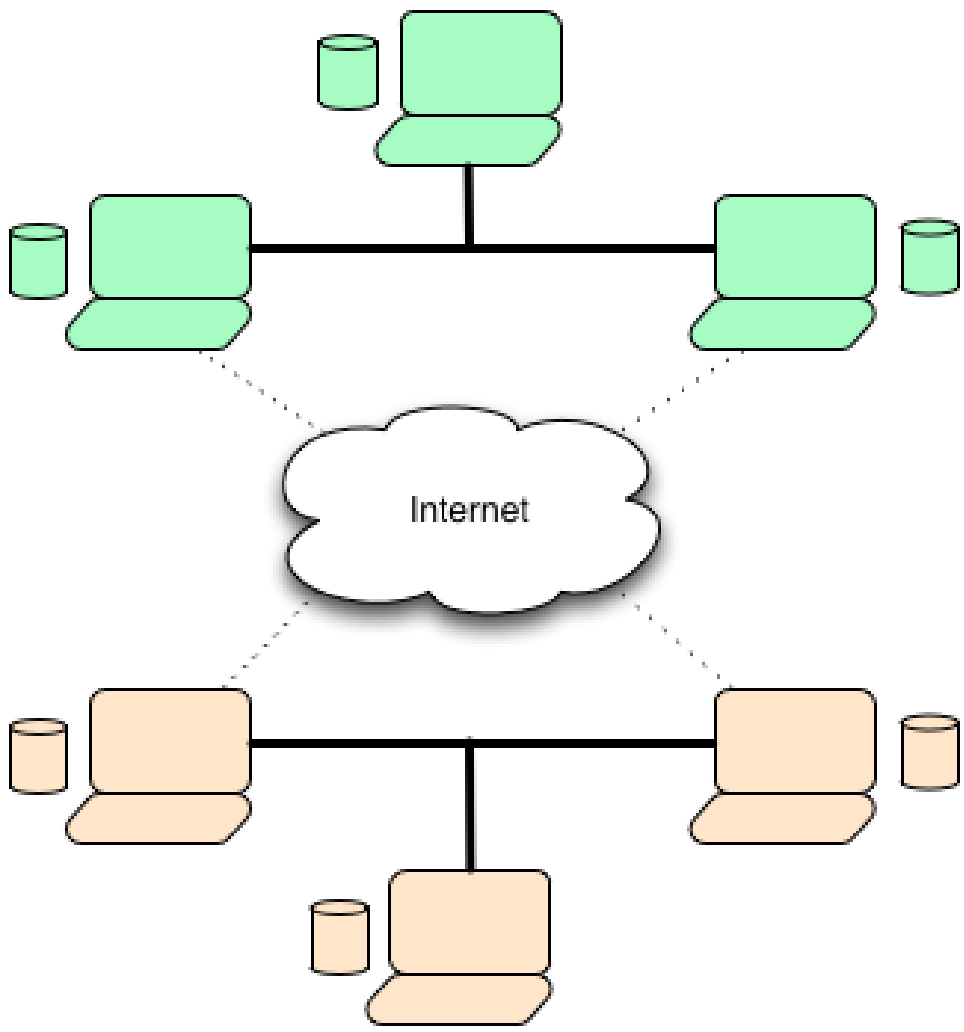


Figura 1: Arquitetura típica de um sistema de informação na era do PC

Aos poucos, funções que antes eram executadas pelos computadores pessoais passaram a ser executadas por servidores na Internet, um processo acelerado com o surgimento da computação em nuvem (*cloud computing*). As reduções de custo e a flexibilidade no uso dos recursos proporcionadas pela computação em nuvem

popularizaram arquiteturas que armazenam dados e que executam software remotamente (Figura 2). Essa mudança, entretanto, criou novas ameaças à privacidade dos usuários. De fato, é impossível saber se e quando os dados são acessados, uma vez que eles estão armazenados em servidores administrados por terceiros. Da mesma forma, é impossível verificar o software em execução na nuvem, já que não temos acesso ao seu código-fonte.

Paralelamente, a computação móvel também se desenvolveu e o acesso aos serviços da nuvem migrou dos computadores pessoais para os *smartphones* e *tablets*. Essa migração não apenas manteve inalterados os problemas já apontados como os agravou, pois o uso de software livre nos dispositivos móveis é complicado por três fatores: (i) os fabricantes de smartphones produzem hardware e software extremamente dependentes um do outro, (ii) os *smartphones* se conectam à rede de telefonia através de uma camada adicional de software restrito e (iii) o grande número de dispositivos e sensores existentes nos *smartphones* operados por drivers restritos.

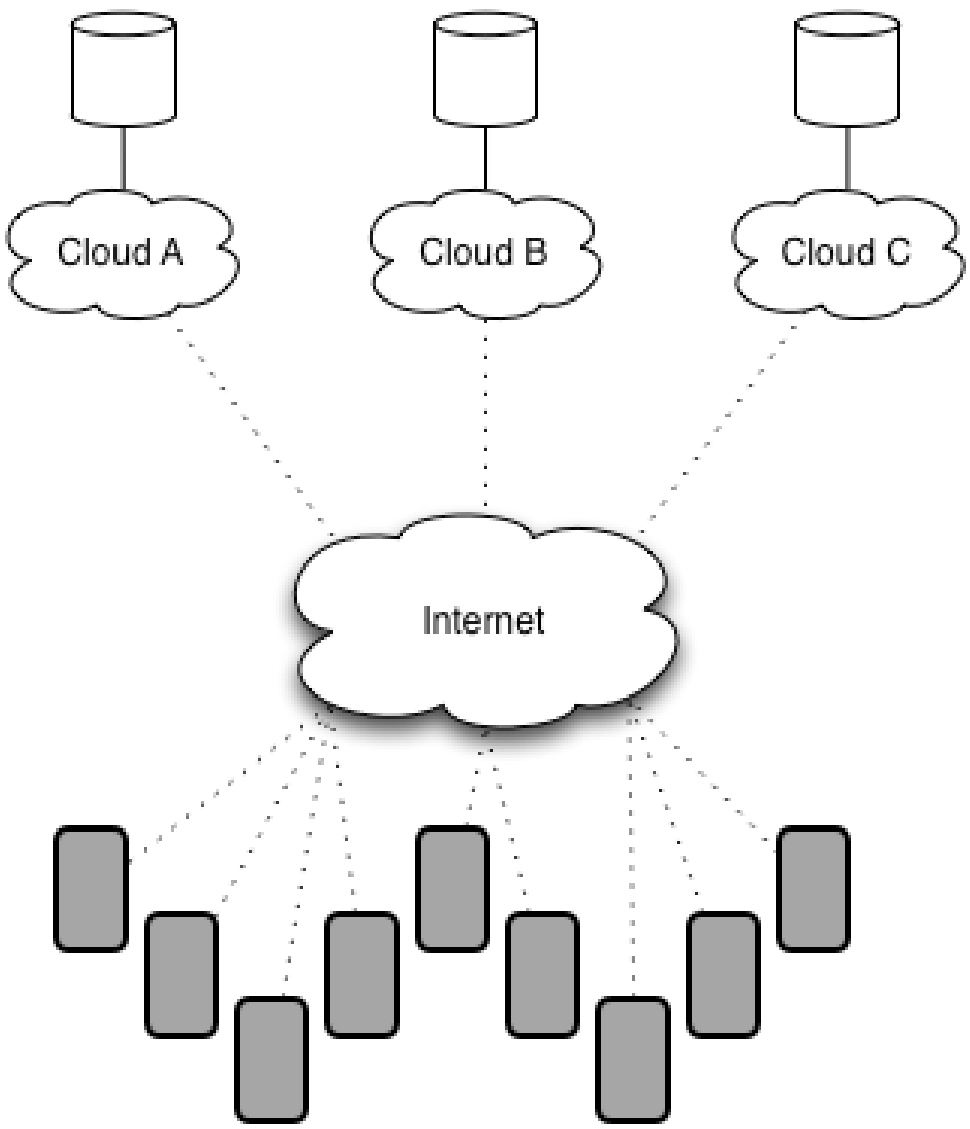


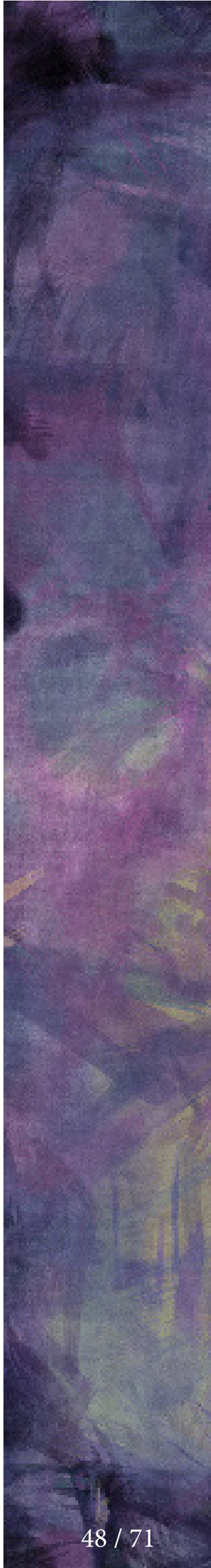
Figura 2: Arquitetura típica de um sistema de informação na era pós-PC

As principais plataformas móveis hoje são restritas, com exceção do Android que possui boa parte de seu código aberto, mas, ainda assim, tem muitas limitações. Existem dois problemas com o sistema Android que colocam em risco a privacidade: (i) ele depende de uma série de *drivers* e bibliotecas restritos, em particular do Google Services, para onde o Google vem migrando uma série de APIs essenciais como as APIs de geolocalização; e (ii) é muito difícil instalar uma versão personalizada do sistema Android no seu smartphone: se, de um lado, é provável que o seu computador pessoal possa executar uma versão do GNU/Linux hoje, de outro, essa flexibilidade é muito mais limitada no caso dos celulares.

Em resumo, hoje em dia utilizamos serviços na Internet sem saber exatamente o que eles fazem, a partir de dispositivos executando software que não podemos verificar e que acessam nossos dados em circunstâncias obscuras. Não à toa, diversos ataques da NSA à privacidade dos cidadãos se aproveitam exatamente dessas falhas: grampos em datacenters são menos eficientes se cada pessoa administra seu próprio servidor de e-mail; a coleta de dados pessoais a partir de aplicações é menos eficiente se elas possuem código aberto e são executadas em uma plataforma livre.

A falta de clareza no armazenamento e no uso de nossas informações permitiu a popularização de um modelo de negócios baseado na coleta indireta de dados pessoais e sua comercialização para terceiros.

A falta de clareza no armazenamento e no uso de nossas informações permitiu a popularização de um modelo de negócios baseado na coleta indireta de dados pessoais e sua comercialização para terceiros, em particular para a divulgação hiper-segmentada de anúncios. Não é coincidência que as vozes que hoje defendem publicamente que o direito à privacidade não é uma prioridade pertencem exatamente àqueles dependentes desse modelo em seus negócios. A questão não é saber se essas empresas são ou não confiáveis (ainda que,



pelo histórico de problemas, elas não sejam): a mera existência de serviços centralizados que concentram enormes quantidades de dados pessoais facilita invasões de privacidade em massa.

Nós, membros da comunidade de computação, temos a responsabilidade de defender a privacidade nos sistemas que desenvolvemos. Temos que utilizar criptografia sempre que dados forem transmitidos ou armazenados, pois os bons algoritmos de criptografia são revisados pela comunidade e resistem a ataques. Temos que conceber arquiteturas de sistemas que sejam extremamente distribuídas sempre que possível, pois arquiteturas distribuídas dificultam o acesso indevido a grandes quantidades de informações pessoais. Temos que estimular e contribuir com projetos como o Replicant, que buscam alternativas realmente livres para o sistema operacional dos *smartphones,* como o F-Droid, que organiza aplicações livres para *smartphones,* e como o Cozy Cloud, que armazena e sincroniza dados pessoais sem passar por servidores de terceiros. Só assim conseguiremos reequilibrar a disputa pelos dados pessoais e ter de volta nossa privacidade. ●

ROBERTO SPEICYS CARDOSO | Doutor em privacidade em sistemas pervasivos pela Universidade Pierre et Marie Curie - Paris VI. Ele é fundador das empresas Ambientic na França, que desenvolve middleware multiplataforma para aplicações móveis, e da Scipopulis no Brasil, que desenvolve soluções móveis e de análise de dados para melhoria da mobilidade urbana em grandes cidades de países em desenvolvimento.

SOFTWARE LIVRE E SEGURANÇA ELEITORAL

Vantagens e limitações com a adoção de tecnologia livre nas eleições eletrônicas

por Diego de Freitas Aranha

APÓS AS PRIMEIRAS ELEIÇÕES COMPLETAMENTE ELETRÔNICAS NO ANO 2000, O BRASIL PERMANECE COMO O ÚNICO PAÍS DO MUNDO A ADOTAR MÁQUINAS DE VOTAÇÃO PURAMENTE ELETRÔNICAS, SEM POSSIBILIDADE DE RECONTAGEM OU VERIFICAÇÃO INDEPENDENTE DOS RESULTADOS, COM SEVERAS LIMITAÇÕES DE TRANSPARÊNCIA.

APÓS A INTRODUÇÃO DAS URNAS ELETRÔNICAS em 1996 e realização das primeiras eleições completamente eletrônicas no ano 2000, o Brasil permanece como o único país do mundo a adotar máquinas de votação puramente eletrônicas, sem possibilidade de recontagem ou verificação independente dos resultados, com severas limitações de transparência. O cenário se torna mais crítico após observadas graves vulnerabilidades no sistema, com impacto no caráter secreto do voto e integridade dos resultados. Trata-se de consequências diretas de processo imaturo de desenvolvimento de software, falta de treinamento formal da equipe de desenvolvimento e ausência de auditoria externa. Há várias formas de se aprimorar o sistema atual, com variados ganhos em transparência e custos de implementação, mas antes de discuti-las é importante analisar, sob um ponto de vista crítico, o modelo atual de auditoria de software e dos resultados da eleição.

A *RESOLUÇÃO 23.397/2013* DO TRIBUNAL SUPERIOR ELEITORAL (TSE) determina que apenas fiscais formalmente indicados por partidos políticos, Ordem dos Advogados do Brasil (OAB) e Ministério Público (MP), "possuem acesso antecipado aos programas de computador desenvolvidos pelo Tri-

Observa-se, portanto, que a fiscalização é realizada apenas por fiscais partidários, representando entidades com conflito de interesse direto com o resultado.

bunal Superior Eleitoral ou sob sua encomenda a serem utilizados nas eleições, para fins de fiscalização e auditoria, em ambiente específico e controlado pelo Tribunal Superior Eleitoral". Os programas incluem o software de votação, sistemas de preparação do equipamento e demais componentes de software que interagem, direta ou indiretamente, com as urnas eletrônicas e totalização dos seus resultados.

Enquanto na teoria parece um mecanismo suficiente e adequado de auditoria, na prática o resultado é outro. Primeiramente, há um termo de confidencialidade mandatório que limita a divulgação das informações coletadas pelos fiscais. Além de limitar a disseminação de informação técnica sobre o sistema de votação, o termo certamente não é capaz de impedir a divulgação interna, em um partido, de vulnerabilidades encontradas por um fiscal malicioso. Também não há registro anterior de indicação de fiscais por parte da OAB ou MP, entidades que deveriam representar a sociedade no processo, mas se omitem possivelmente por questões políticas. Observa-se, portanto, que a fiscalização é realizada apenas por fiscais partidários, representando entidades com conflito de interesse direto com o resultado. Há também outras limitações de natureza técnica, como a janela limitada de seis meses para fiscalização dos programas, mesmo com alterações ainda em curso; a restrição do ambiente controlado pelo TSE, limitando as ferramentas de auditoria; e a alta complexidade causada por um conjunto de programas com volume superior a 10 milhões de linhas de código[1]. Há maior transparência no processo de totalização de resultados, com a comparação entre Boletins de Urna físicos e eletrônicos, mas obstáculos logísticos também dificultam sua

1
Esta estimativa inclui apenas aplicativos e sistema operacional utilizados na urna eletrônica, mas como o TSE realiza intervenções diretas no código externo, há necessidade de se auditá-lo praticamente em sua totalidade.

2
O projeto Você Fiscal representa uma tentativa em curso de realizar essa conferência distribuída e por amostragem: www.vocefiscal.org

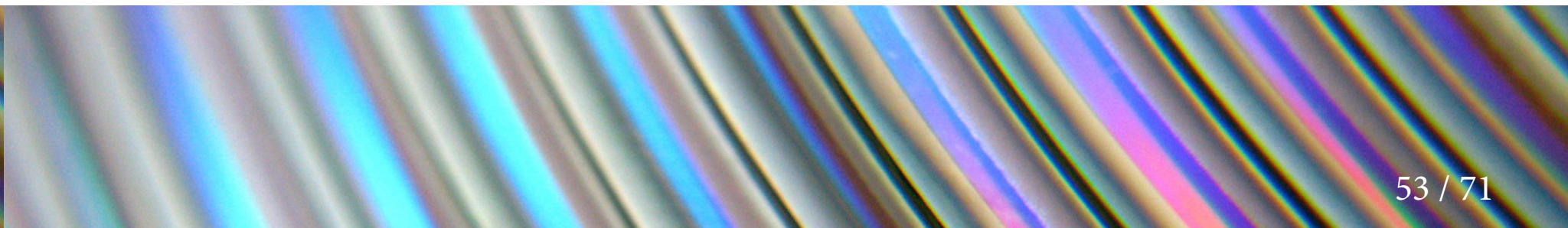
realização[2]. Resta torcer para que partidos concorrentes se anulem em eventual atuação maliciosa, ignorando o fato elementar de que eleições são financiadas também com recursos públicos e servem, antes de tudo, aos próprios eleitores. É surpreendente que eleições puramente eletrônicas ainda utilizem tecnologia que não seja diretamente auditável pela sociedade.

Um novo modelo

A ADOÇÃO DE TECNOLOGIAS VERDADEIRAMENTE livres, tanto no software quanto no hardware das urnas eletrônicas, pode ampliar substancialmente a transparência do sistema. Assim como a adoção pelo TSE do sistema operacional GNU/Linux em 2008, em um movimento louvável, foi defendida sob o argumento da segurança e independência de tecnologias proprietárias, as mesmas propriedades se aplicam ao software desenvolvido pelo Tribunal. Além de permitir trivialmente auditorias independentes da tecnologia eleitoral, verificação de alegações de natureza técnica do Tribunal e contribuições por parte da sociedade, a disponibilização do sistema sob licenças livres possui outra grande vantagem, ligeiramente mais sutil. Quando confrontados com a possibilidade de verificação independente da qualidade do sistema e seus mecanismos de segurança, uma equipe de desenvolvimento que preza por sua reputação e credibilidade não possui alternativa além de aprimorar as suas práticas e projeto, resultando em um sistema mais seguro e confiável. Desde que haja interesse da comunidade técnica para auditar o sistema, a pressão externa se torna um incentivo formidável para incremento de qualidade.

Desde que haja interesse da comunidade técnica para auditar o sistema, a pressão externa se torna um incentivo formidável para incremento de qualidade.

Nos Testes Públicos de Segurança de 2012, foram observa-

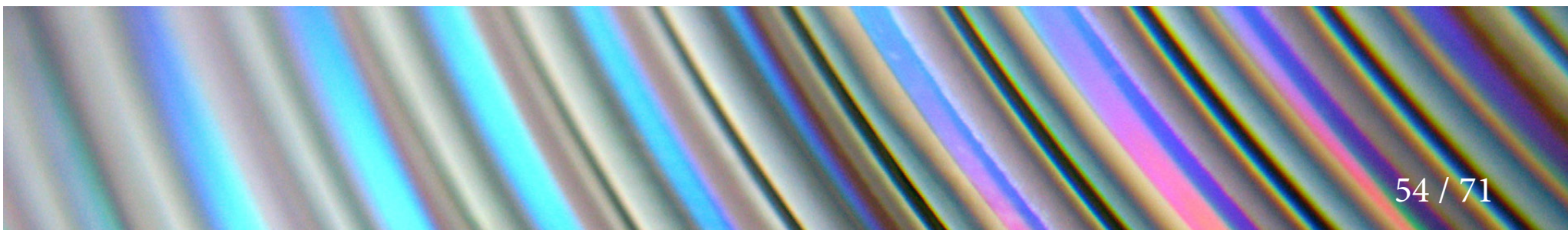


das, diretamente no código-fonte, a utilização de algoritmos criptográficos obsoletos, a presença e compartilhamento massivo de chaves criptográficas, a proteção inadequada do sigilo do voto e verificação insuficiente de integridade do software de votação[3]. Esta última é realizada pelo próprio equipamento, de forma passiva, abrindo a possibilidade de adulteração indetectável do software e seus produtos. Vale lembrar que se trata de uma base de código com então 16 anos de história e responsabilidade crítica na manutenção do regime democrático no país. É improvável que um sem-número de vulnerabilidades e erros de projeto, descobertos na ocasião em apenas 5 horas de investigação, permaneçam tanto tempo em um sistema auditável por qualquer profissional qualificado do governo, academia ou indústria.

3
Aranha, D. F.; Karam, M. M.; Miranda, A; Scarel, F. Software Vulnerabilities in the Brazilian Voting Machine. In: Dimitrios Zissis, Dimitrios Lekkas. (Org.) Design, Development, and Use of Secure Electronic Voting Systems. IGI Global, p. 149-175, 2014

Problema resolvido?

INFELIZMENTE, A SIMPLES DISPONIBILIDADE de software e hardware sob licenças livres é requisito necessário, mas não suficiente para garantir eleições inteiramente seguras e auditáveis. Mesmo que a tecnologia seja minuciosamente examinada por fiscais independentes, não há garantia técnica de que o mesmo sistema foi utilizado no dia da eleição. A utilização de compilação determinística, permitindo a criação de versões reprodutíveis dos programas em formato binário, e assinaturas digitais para verificação ativa da integridade do software não impedem que atuação maliciosa interna comprometa instalações individuais do sistema. O processo de compilação e geração de mídias de instalação para um sistema dessa magnitude é complexo e pode ser difícil de reproduzir externamente, dadas as inúmeras dependências. A verificação ativa de integridade também exige a presença de fiscais na abertura das seções eleitorais,



software adicional para validação das assinaturas digitais e acesso direto ao conteúdo do equipamento. Finalmente, a fiscalização das eleições não deveria ser limitada a profissionais que dominam conhecimento técnico, mas estendida a toda a sociedade. Por esse motivo, o caminho para eleições seguras e transparentes não envolve apenas a adoção de tecnologia auditável e sua livre disponibilização, mas também a implementação de um registro físico e anônimo do voto, que junto com procedimentos adequados para auditoria e recontagem distribuam a todos os eleitores a possibilidade de verificar o registro correto de seus votos, a exemplo do resto do mundo. Nada mais natural, visto que são os eleitores os maiores interessados no processo democrático.

DIEGO DE FREITAS ARANHA | É professor da Universidade Estadual de Campinas, com experiência na área de Criptografia e Segurança Computacional. Coordenou a primeira equipe de investigadores independentes capaz de detectar e explorar vulnerabilidades no software da urna eletrônica em testes controlados organizados pelo Tribunal Superior Eleitoral.

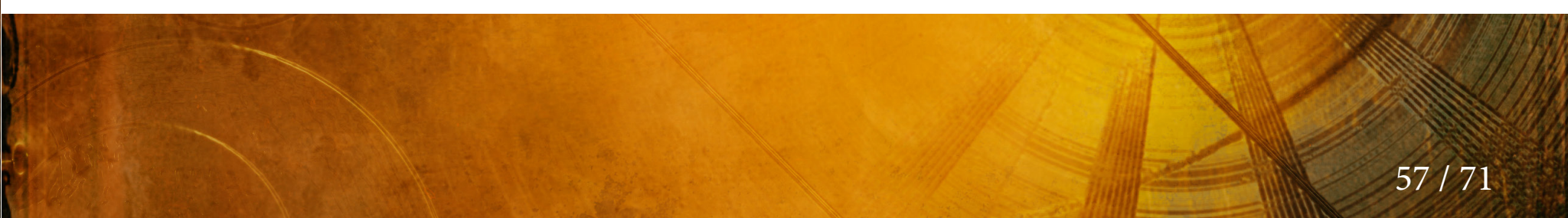
DEZ TESES SOBRE SOFTWARE LIVRE NO BRASIL PÓS-SNOWDEN

por Sérgio Amadeu da Silveira

DADO O PAPEL DECISIVO DO SOFTWARE NA CONSTRUÇÃO DA SOCIEDADE E DA CULTURA, A DEPENDÊNCIA INTERNACIONAL BRASILEIRA NA ÁREA TEM EFEITOS SIGNIFICATIVOS PARA O PAÍS. CIÊNCIA ABERTA, SOFTWARE LIVRE E DESENVOLVIMENTO COLABORATIVO ESTÃO NA BASE DE DIVERSAS AÇÕES QUE O BRASIL DEVE TOMAR PARA REVERTER ESSA DEPENDÊNCIA.

O pesquisador Alexander Galloway escreveu, em seu livro Protocol: how control exists after decentralization, que o computador digital é uma "máquina abstrata capaz de realizar o trabalho de qualquer outra máquina (desde que ele possa ser descrito logicamente)". Essa metamáquina está presente em um crescente número de atividades do nosso cotidiano. Computadores ou máquinas de processar informações exigem softwares. Por isso, Lev Manovich afirmou (http://lab.softwarestudies.com/2008/11/softbook.html) que "nossa sociedade contemporânea pode ser caracterizada como uma sociedade de software e nossa cultura pode ser justamente chamada de uma cultura de software — porque o software, hoje, desempenha um papel central na formação de ambas e muitas das suas estruturas imateriais que formam a 'cultura' ".

Consequentemente, o poder econômico na sociedade informacional está cada vez mais concentrado em países capazes de desenvolver softwares e tecnologias de armazenamento, processamento e distribuição de informações. A maioria dessas tecnologias foram criadas nos Estados Unidos, mais precisamente na Califórnia, onde processos sinergéticos foram possíveis pela concentração de cientistas, pesquisadores e hackers. Grandes corporações se ergueram e disseminaram essas tecnologias da inteligência pelo planeta. O Estado norte-americano, incentivador da inovação tecnológica como elemento de sua superioridade estratégica, aproveitando o caráter cibernético das redes e dispositivos digitais, organizou processos de vigilância e espionagem massiva, recentemente denunciados pelo ex-agente da NSA, Edward Snowden.

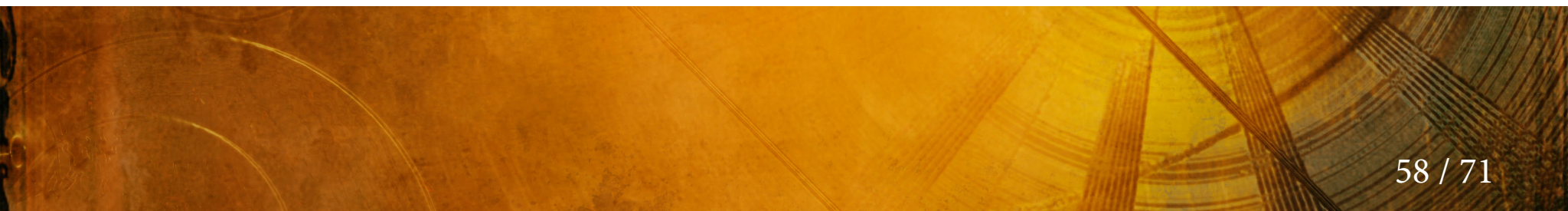


Neste cenário, reúno aqui dez teses sobre caminhos que o Brasil deveria trilhar para nos reposicionarmos como desenvolvedores de tecnologias. A tese central passa pela defesa da ciência aberta, do software livre e das tecnologias desenvolvidas colaborativamente. O caminho do conhecimento fechado, das patentes e das tecnologias proprietárias está minado, bloqueando um ritmo adequado e necessário de desenvolvimento e inovação. Aceitar o modelo proprietário hegemônico de produção tecnológica como natural baseia-se na crença ingênua na linearidade histórica e desconsidera a deterioração dos termos de troca entre países. Por exemplo, em 2012, somente a IBM registrou 5.866 patentes e a Microsoft 3.121, ou seja, apenas duas empresas norte-americanas registraram mais patentes do que todas as empresas e universidades brasileiras.

SEGUEM DEZ PROPOSIÇÕES PARA A NOSSA REFLEXÃO:

1. O desenvolvimento de tecnologias da informação é estratégico

UM DOS PRINCIPAIS ARGUMENTOS CONTRÁRIOS à adoção de software livre nos governos é que a gestão pública deve adquirir produtos de grandes empresas com larga experiência internacional, pois o importante é a atividade final do governo. Ocorre que o desenvolvimento de tecnologias da informação adequadas e necessárias é uma atividade estratégica para o país. Foi resolvendo problemas concretos que os Estados Unidos criaram sua capacidade tecnológica. O sociólogo Manuel Castells escreveu, em seu livro A *sociedade em rede*, que "as elites aprendem fazendo e com isso modificam as aplicações da tecnologia, enquanto a maior parte das pessoas aprende usando e, assim, permanecem dentro dos limites do pacote da tecnologia".

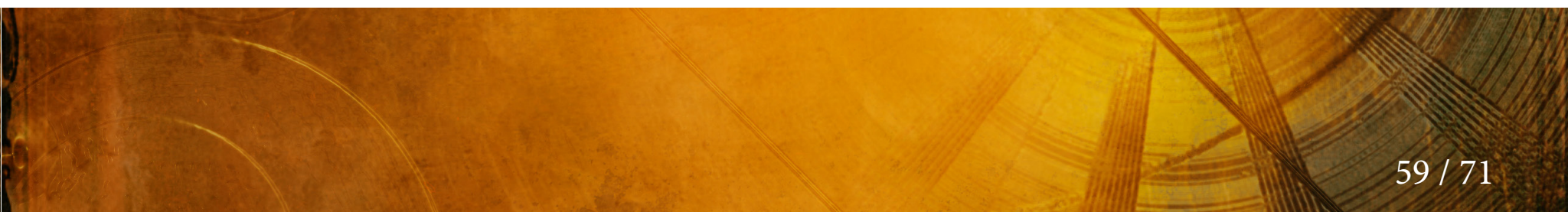


2. Parte do poder de compra do governo deve ser voltado ao desenvolvimento das capacidades estratégicas do país

O GASTO ANUAL DO GOVERNO FEDERAL COM LICENÇAS de software proprietários que possuem similares livres ultrapassa a soma de R$ 1 bilhão apenas na administração direta. Com apenas 20% desse gasto para o desenvolvimento de softwares livres, abriríamos muito mais espaços para empresas nacionais, incentivaríamos as comunidades de desenvolvedores e melhoraríamos a qualidade das nossas soluções voltadas ao serviço público.

3. É necessário apostar no bloqueio da espionagem econômica, diplomática e política praticada pelas agências de inteligência das potências expansionistas

O discurso das corporações de tecnologia que trabalham para o sistema de inteligência norte-americana combina dois argumentos conflitantes. O primeiro diz que não é possível se proteger diante do poder tecnológico e da capacidade de espionagem norte-americanos. O segundo propõe a aquisição de soluções e produtos completamente seguros (algo que efetivamente não é possível prometer). Os dois argumentos são usados alternadamente. O primeiro visa principalmente impedir o desenvolvimento de soluções nacionais para segurança. O segundo quer desqualificar o uso de softwares livres. O fato é que softwares fechados são por definição inseguros, pois não permitem a auditabilidade plena em seu código-fonte. Já passou da hora de o governo incentivar e



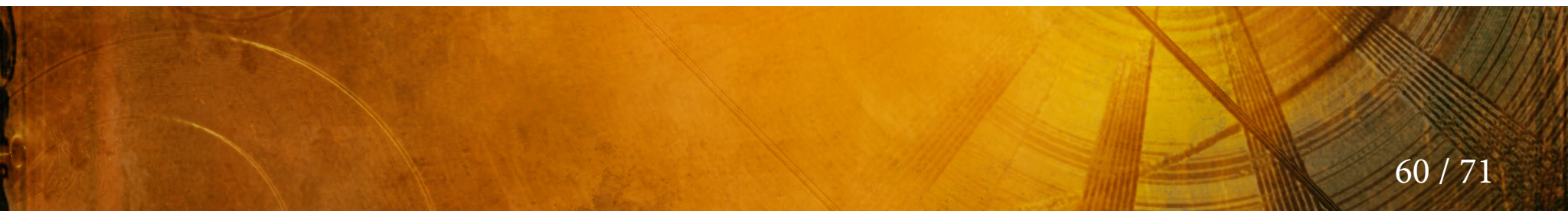
apoiar financeiramente as comunidades que criam soluções de segurança abertas e livres, tal como o projeto Leap de aplicação da criptografia para a comunicação em rede.

4. A aposta nas tecnologias livres e abertas pode nos levar à fronteira da criação tecnológica de ponta

Megaempresas inovadoras, como Yahoo, Twitter, Google e Facebook, construíram sua infraestrutura e suas soluções com base no software aberto e livre. O próprio reposicionamento dos aplicativos e soluções para aparelhos móveis ocorreu quando o Google retrabalhou o Linux, lançando o Android. Tais exemplos mostram que devemos explorar, recombinar e reconfigurar as soluções em código aberto e com licenças permissivas de uso. Devemos incentivar e apoiar com recursos públicos a invenção de novos projetos em áreas pouco exploradas pela indústria do software proprietário. Com isso, poderemos criar polos de inventividade, o que não será possível no mundo dominado pelas patentes e licenças de propriedade.

5. Constituir uma instituição federal de incentivo ao desenvolvimento de software livre e tecnologias abertas traria efeitos excelentes no médio e longo prazos

Assim como o governo incentivou nossos jovens a estudar no exterior, devemos atrair os melhores cérebros das comunidades de software livre do planeta para



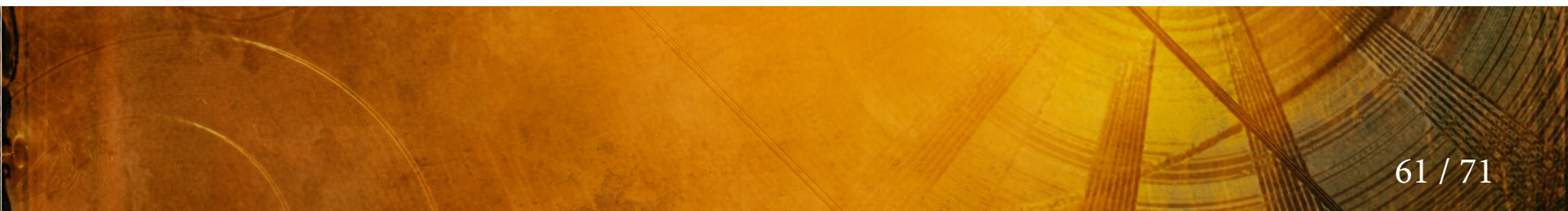
trabalhar em nosso país. Com isso, criaremos espaços de sinergia que podem gerar ondas criativas vitais para nosso reposicionamento tecnológico no mundo. Essa instituição de incentivo ao software livre buscaria, por intermédio de editais e bolsas, financiar projetos específicos localizados no Brasil e comunidades de desenvolvimento, desde que tivessem desenvolvedores residentes no país.

6. É preciso fortalecer o uso de software livre no ensino profissionalizante

A implementação de um programa de bolsas de aprendizagem e a criação de estágios para a instalação e administração de servidores com software livre, para estudantes do ensino médio, permitirão um salto tecnológico em várias regiões do país.

7. O currículo escolar deve ser adaptado à realidade atual, em que a programação é conhecimento essencial

Devemos preparar a introdução de aulas de programação desde os primeiros anos da escola, bem como precisamos aprimorar o ensino da matemática com as possibilidades abertas pelas tecnologias digitais. O FISL (Fórum Internacional de Software Livre) deveria abrir um intenso debate para sugerir a inclusão do ensino de software já no ensino fundamental. O resultado deste processo de formulação colaborativa resultaria em uma proposta a ser enviada para o Ministério de Educação.



8. A pesquisa e o desenvolvimento de novas tecnologias no Brasil devem almejar suplantar o estado da arte

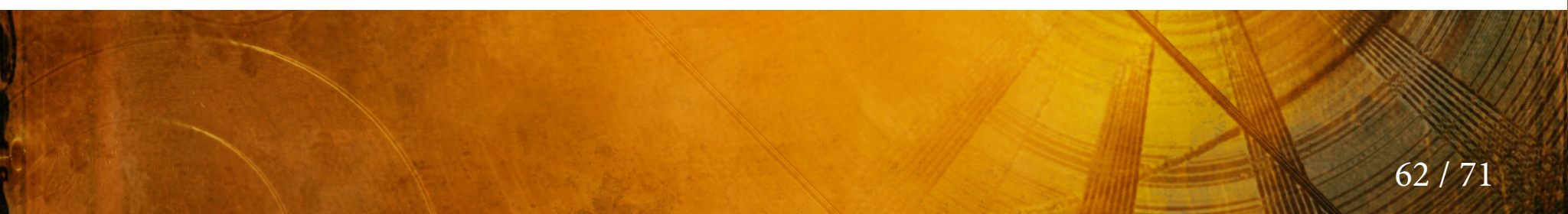
Temos que apostar em tecnologias que solucionem nossos grandes problemas e que ainda não foram criadas. O Brasil deveria escolher áreas de fronteira científica e tecnológica para lançar esforços investigativos e inventivos. O setor público deveria reduzir o gasto com licenças de propriedade de software e alocar parte desse recurso em tentativas de solução inovadora e inventiva de problemas.

9. O Brasil precisa ser agente ativo dos fóruns internacionais da área de tecnologia

Empresas estatais, como o Serpro (Serviço Federal de Processamento de Dados), deveriam criar convênios com universidades para alocar recursos em projetos de acompanhamento dos diversos órgãos internacionais de padronização das tecnologias da informação e de definição de protocolos digitais.

10. A aposta em redes mesh e no uso de espectro livre para a Internet das Coisas tem potencial altamente transformador em termos tanto sociais quanto econômicos

A tecnologia *mesh* de conexão de última milha, através da qual computadores e celulares dos próprios usuários são responsáveis pelas interconexões, pode ser uma



solução de inclusão digital e de redução do custo Brasil de telecomunicações. A indústria não explora as diversas possibilidades das tecnologias *mesh* exatamente pelo seu grande potencial de gratuidade, o que apavora as operadoras de telecom. Podemos apostar no *smart radio*, capaz de otimizar o uso compartilhado do espectro eletromagnético, para a ocupação desse espectro para conectar coisas, infraestruturas e pessoas como nunca se viu. ●

SÉRGIO AMADEU DA SILVEIRA | É professor da UFABC. Doutor em Ciência Política. Foi presidente do Instituto Nacional de Tecnologia da Informação (2003-2005). Integrou o Comitê Gestor da Internet no Brasil (2003-2005 e 2010-2013). Coordenador do Governo Eletrônico e da Inclusão Digital da Prefeitura de São Paulo (2001-2003). Coordenou o primeiro Comitê de implementação de Software Livre no governo federal (2003). É membro do Comitê Científico da Associação Brasileira de Pesquisadores em Cibercultura, ABCiber.

SOFTWARE LIVRE E CONTEÚDOS EDUCACIONAIS ABERTOS NO ENSINO DE COMPUTAÇÃO

por **Ellen Francine Barbosa, Maurício Massaru Arimoto, Seiji Isotani e José Carlos Maldonado**

SOFTWARE LIVRE E CONTEÚDOS EDUCACIONAIS ABERTOS TÊM GANHADO CRESCENTE IMPORTÂNCIA COMO FORMA DE PROMOVER MECANISMOS EFETIVOS PARA A EDUCAÇÃO ABERTA E FLEXÍVEL, AMPLIANDO O ACESSO AO CONHECIMENTO COM CUSTOS REDUZIDOS E VALORIZANDO A COOPERAÇÃO E COLABORAÇÃO COMO PRÁTICAS EMERGENTES E INOVADORAS.

NA ERA DO CONHECIMENTO E DA INFORMAÇÃO, o ensino de Computação torna-se cada vez mais relevante. Discussões relacionadas ao ensino de Computação vão além da educação em nível superior. Um tema recorrente de debate é a inserção de tecnologias computacionais nas escolas, sobretudo no ensino básico e fundamental. Almeja-se, com isso, tanto o crescimento pessoal e social do cidadão como a qualidade de sua formação.

Atualmente, no ensino de Computação (assim como ocorre em outras áreas do conhecimento), ainda há uma forte dependência pelo uso de software e conteúdos educacionais "proprietários". Esses formatos fechados possuem diversas restrições quanto ao seu uso, sejam elas questões técnicas, barreiras legais ou de preço. Como consequência, a evolução do software e conteúdos educacionais pode não ocorrer regularmente, tornando-os ultrapassados e provavelmente inadequados a um cenário educacional mais dinâmico, que seja capaz de motivar o aprendiz na descoberta e consolidação de novos conhecimentos. Ainda, com o passar do tempo, versões de software e conteúdo proprietário acabam sendo descontinuadas, tornando-se obsoletas. Problemas como esses inibem a criatividade e a inovação no contexto educacional.

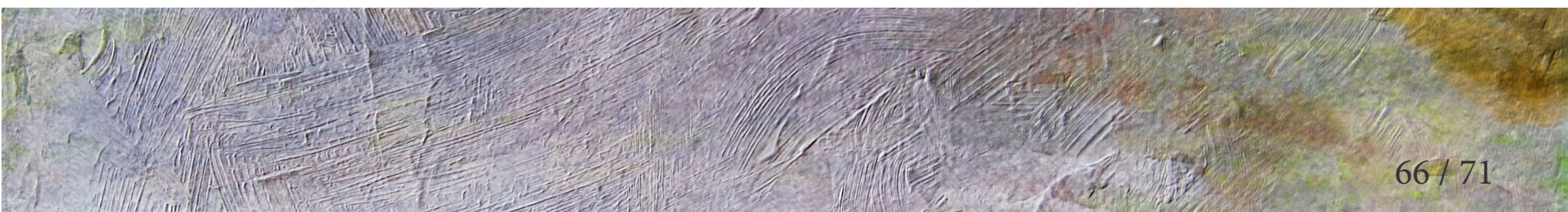
Fonte: edutechdebate.org

Alternativamente, software livre e conteúdos educacionais abertos (também conhecidos com REAS – recursos educacionais abertos) têm ganhado crescente importância como forma de promover mecanismos efetivos para a educação aberta e flexível, ampliando o acesso ao conhecimento com custos reduzidos e valorizando a cooperação e colaboração como práticas emergentes e inovadoras de se "ensinar e aprender".

Sob o ponto de vista da filosofia aberta e livre, o usuário, seja ele professor ou aluno, detém determinados direitos e liberdades para (re)usar, adaptar, revisar e (re)distribuir o software ou conteúdo educacional.

Sob o ponto de vista da filosofia aberta e livre, o usuário, seja ele professor ou aluno, detém determinados direitos e liberdades para (re)usar, adaptar, revisar e (re)distribuir o software ou conteúdo educacional de modo que outras pessoas possam ser beneficiadas. Assim, características importantes da utilização de software livre e conteúdos educacionais abertos estão relacionadas às possibilidades de "compartilhamento" e consequente " (re)produção" desses recursos. Novos trabalhos podem ser produzidos de maneira mais rápida a partir da derivação de trabalhos produzidos por outros usuários. Pode-se, ainda, reutilizar recursos existentes, adequando-os e adaptando--os a fim de atender a contextos particulares ou necessidades e objetivos educacionais específicos.

Além disso, iniciativas voltadas à adoção de software livre e conteúdos educacionais abertos também são responsáveis por propiciar oportunidades de empreendedorismo e inovação em instituições de ensino, sejam elas públicas ou particulares. A ruptura de paradigmas, do tradicional/proprietário para o aberto/livre, além de requerer novos modelos de gestão de negócios, motiva novas práticas pedagógicas, fomenta novos planos de ação, desperta nichos de mercado diferenciados e

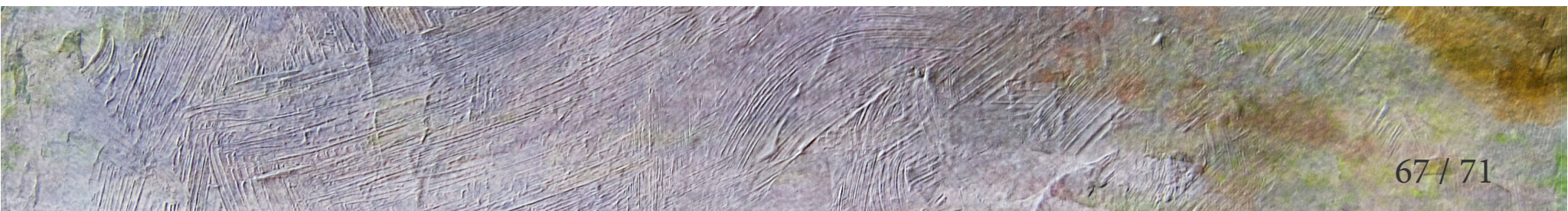


promove a inovação, a pesquisa científica e o empreendedorismo social.

Considerando o ensino de Computação, em especial, todos os aspectos e características aqui discutidos assumem fundamental importância. A Computação como ciência envolve experimentação. Frequentemente, alunos, professores e pesquisadores conduzem experimentos com o objetivo de: (1) identificar novas áreas de pesquisas; (2) explicar uma observação e/ou fenômeno; ou (3) descrever e melhorar características e fenômenos de um estudo. Nesse contexto, o projeto experimental, incluindo materiais e métodos, dados coletados, ferramentas e softwares utilizados, resultados da análise e lições aprendidas podem ser compartilhados livre e abertamente, facilitando seu entendimento e reprodução por outros estudantes e pesquisadores. Some-se a isso o estreito relacionamento entre tecnologias computacionais, inovação e empreendedorismo, demandas essas que necessitam ser adequadamente supridas pelos cursos de Computação, sejam eles básicos ou de nível superior.

A adoção de software livre e conteúdos abertos vem, aos poucos, avançando nas universidades brasileiras.

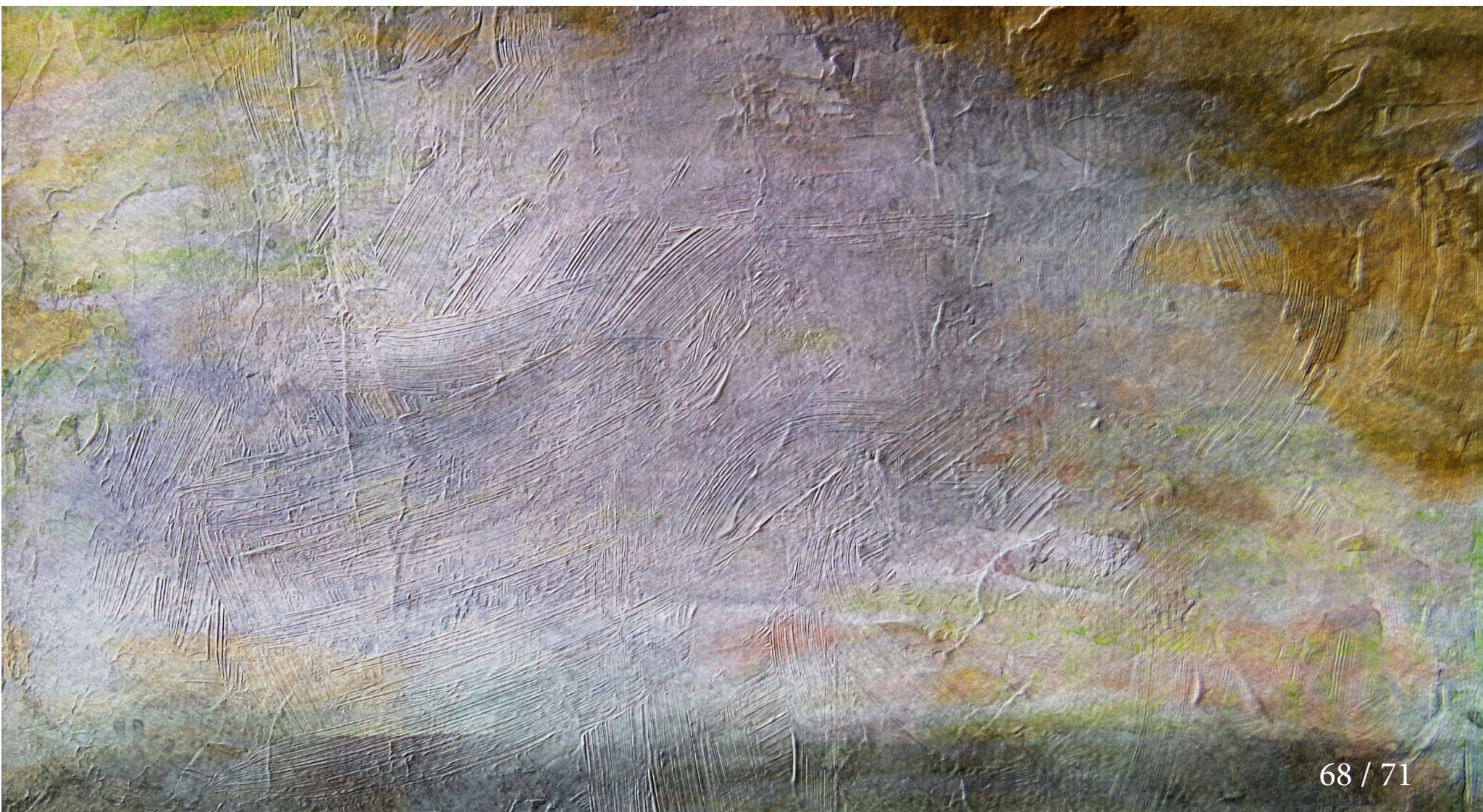
A adoção de software livre e conteúdos abertos vem, aos poucos, avançando nas universidades brasileiras. Um bom exemplo disso são o ICMC-USP (Instituto de Ciências Matemáticas e de Computação, São Carlos) e o IME-USP (Instituto de Matemática e Estatística, São Paulo), que, juntos, mantêm o CCSL (Centro de Competência em Software Livre), centro que agrega atividades relacionadas ao ensino de graduação e pós-graduação, pesquisa, desenvolvimento e divulgação de software livre e conteúdos educacionais abertos. Também no ICMC-USP, destaca-se ainda o NAP-SoL, um núcleo de apoio à



pesquisa em software livre e recursos educacionais abertos. Tanto o CCSL como o NAP-SoL têm atuação em diferentes áreas do conhecimento, sobretudo em Computação.

Entretanto, apesar das vantagens e benefícios já identificados, há ainda um caminho árduo a ser percorrido: iniciativas voltadas para o uso de software livre e conteúdos educacionais abertos no ensino de Computação, e em diversas outras áreas do conhecimento, ainda se mostram incipientes. Salienta-se, portanto, a importância de abrir mão de formatos proprietários que imponham restrições à produção intelectual, à disseminação e à partilha de conhecimento. De fato, é preciso que haja uma conscientização de que conteúdos educacionais abertos em conjunto com software livre configuram um meio efetivo para a colaboração, a troca de experiências e, sobretudo, para a inovação e o aprimoramento dos atuais modelos e práticas de ensino, inclusive na área de Computação. ●

Apesar das vantagens e benefícios já identificados, há ainda um caminho árduo a ser percorrido.

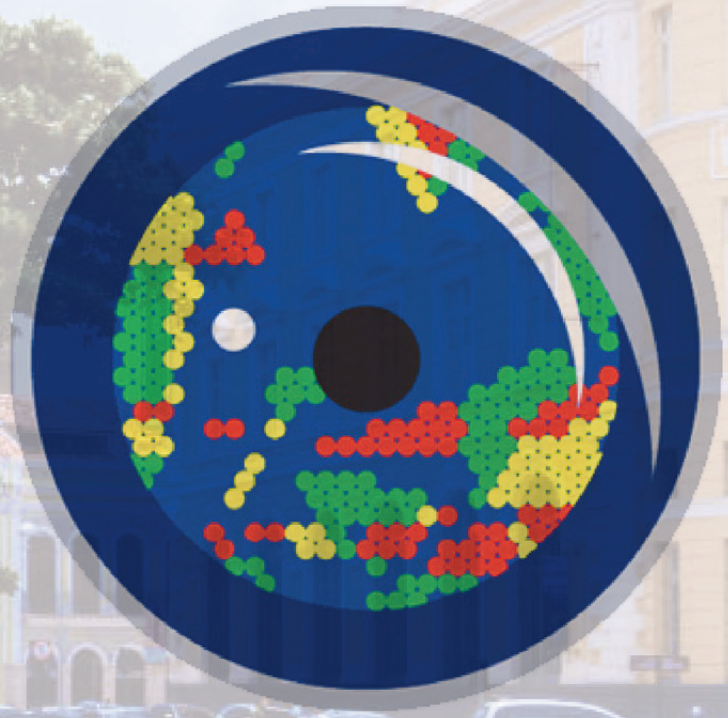


ELLEN FRANCINE BARBOSA | Professora Associada do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC-USP). Possui Bacharelado em Ciência da Computação pela Universidade Estadual de Londrina (UEL), Mestrado e Doutorado em Ciência da Computação e Matemática Computacional pelo ICMC-USP. É membro do Centro de Competência em Software Livre (CCSL-USP), do Núcleo de Apoio à Pesquisa em Software Livre (NAP-SoL) e do Conselho Municipal da Secretaria da Educação do Município de São Carlos/SP.

MAURÍCIO MASSARU ARIMOTO | Doutorando em Ciências de Computação e Matemática Computacional pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC-USP). É Mestre em Ciência da Computação pelo Centro Universitário Eurípides de Marília (UNIVEM). Possui experiência na área de Ciência da Computação, com ênfase em Engenharia de Software e Computação Aplicada à Educação.

SEIJI ISOTANI | Professor Associado do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC-USP). Possui Bacharelado e Mestrado em Ciência da Computação pela Universidade de São Paulo (IME-USP) e Doutorado em Engenharia da Informação pela *Osaka University* (Japão). Realizou seu Pós-Doutorado na *Carnegie Mellon University* (EUA). É Membro do Conselho Municipal da Secretaria da Educação do Município de São Carlos e Representante da SBC no *Technical Committee on Education* (TC3) da IFIP.

JOSÉ CARLOS MALDONADO | Professor Titular do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC-USP). Possui graduação em Engenharia Elétrica pela Universidade de São Paulo (USP), Mestrado em Engenharia e Tecnologias Espaciais pelo Instituto Nacional de Pesquisas Espaciais (INPE) e Doutorado em Engenharia Elétrica – Computação e Automação pela Universidade Estadual de Campinas (UNICAMP). Foi Presidente da Sociedade Brasileira de Computação SBC (2007-2009, 2009-2011). É coordenador do Centro de Competência em Software Livre (CCSL-USP) e do Núcleo de Apoio à Pesquisa em Software Livre (NAP-SoL).



20 a 23 JULHO 2015

RECIFE | PERNAMBUCO

CSBC2015

XXXV CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

a internet de tudo, toda observada

csbc2015.cin.ufpe.br