

Open Source Software Evolução e Tendências

O que mudou?

Cezar Taurion

Prefácio

É com grande prazer que escrevo este prefácio, pois foi através de grande parte dos textos que estão aqui compilados que me instruí sobre o Open Source.

Acho importante entretanto destacar que quando os textos começaram a ser publicados, eram poucos os profissionais com a experiência do Taurion que se dedicavam tanto a tratar do tema e a estudá-lo de forma aprofundada e racional.

Conheço inúmeros executivos de TI que de fato só entenderam de verdade o que é o Open Source e seu impacto nos negócios após lerem alguns dos textos aqui compilados e a assistirem palestras do Taurion por este Brasil afora. Não foram também raras as discussões sobre o assunto que vi sendo encerradas com “mas o Taurion escreveu que...” e fim de papo.

Confesso que sou um apaixonado pelo “mundo livre” e que me envolvi de corpo e alma com o desenvolvimento de padrões abertos e software livre e por isso sempre me envolvi de forma mais apaixonada em discussões sobre o tema, e por isso considero esta compilação extremamente importante, pois nela a paixão é colocada de lado e a celeridade e racionalidade estão sempre em evidência. Quem viveu nas trincheiras do Software Livre nos últimos anos sabe exatamente do que estou falando e sabe como é difícil manter “os pés no chão” de vez em quando.

Uma coisa interessante que temos no mundo hoje, é que na América do Norte e em grande parte da Europa Open Source e Software Livre são de certa forma tratados como sinônimos, enquanto que na América Latina existem diferenças ideológicas muito fortes entre os dois temas e por isso discussões, debates e conflitos são constantes. De forma geral, empresas como Google, Red Hat, IBM e a finada (e saudosa) Sun Microsystems, sempre advogaram e trabalharam com o Open Source (e raramente vemos estas empresas se posicionando sobre Software Livre).

Partimos neste livro em um 2007 onde muito se sonhava sobre o Open Source no mundo real (ou mercado corporativo) e chegamos a 2011 onde você provavelmente carrega um software Open Source em seu novo smartphone adquirido no Natal, que vai certamente se conectar à Internet através de um hotspot wifi que também roda Open Source, para utilizar uma implementação Open Source dos protocolos Internet e finalmente chegar a um servidor de conteúdo. Ali, a probabilidade é enorme de que exista mais Open Source, do sistema operacional à aplicação sendo acessada, passando pelo middleware, onde o Open Source mais uma vez domina e não para de crescer...

Se em 2007 ler os artigos aqui compilados te ajudaria a ver para onde o mundo estava caminhando, lê-los agora em 2011 certamente vai te ajudar a entender o mundo em que vivemos, e observando o caminho percorrido, vai ficar bem mais fácil ver onde chegaremos.

Bem vindo à bordo, e boa leitura a todos !

1. Introdução

A idéia de publicar alguns livros com a coletânea de posts que já escrevi para o meu blog na comunidade MydeveloperWorks (www.ibm.com/developerworks/blogs/page/ctaurion) vinha me martelando há algum tempo. Gosto de escrever e um blog me dá a liberdade de expressão que as colunas nas revistas especializadas, por razões editoriais, não permitem. De maneira geral levanto um post pelo menos uma vez por semana. O blog começou em janeiro de 2007 e em agosto de 2010, quando comecei a preparação desta série de blogbooks, já somava mais de 500 posts. As minhas observações sobre os acessos dos visitantes ao blog mostravam que depois de algum tempo os posts eram “esquecidos”. Ou seja, depois de algum tempo, praticamente não eram mais acessados. Muitos posts mais antigos, embora plenamente válidos e provavelmente ainda de interesse para os leitores, não poderiam ser trazidos de volta à vida?

Surgiu a idéia: por que não agrupar os posts mais relevantes por assuntos e publicá-los em blogbooks? Uma conversa com meu amigo, desenvolvedor, escritor e agora editor da SingularDigital, Claudio de Souza Soares, definiu o projeto. Sim, vou publicar os posts em forma de blogbooks.

Na primeira etapa do projeto agrupei os posts por temas, identificando as conexões entre eles. As tags me ajudaram muito nisso. Assim, cada assunto ou conjunto de assuntos se tornará um livro. Este especificamente, aborda um tema no qual venho me envolvendo entusiasticamente desde o final dos anos 90: Open Source.

Em 2004 publiquei um livro sobre Software Livre (Software Livre - Potencialidades e Modelos de Negócios, editora Brasport). Já fazem mais de seis anos desde então e muita coisa aconteceu. Creio que é tempo de fazer um balanço do que mudou nestes seis anos. Open Source já é, indiscutivelmente, parte integrante da paisagem da indústria de software. O modelo continua evoluindo e se disseminando pela sociedade. Hoje ignorar o Open Source é ignorar a própria evolução da indústria.

Open Source é inevitável para qualquer empresa, seja ela produtora ou usuária de software. Sua barreira de entrada é praticamente zero e qualquer software Open Source está apenas a um clique de distância. Basta acessar um site e fazer download. Além disso, muitos produtos comerciais embutem soluções Open Source. Segundo estimativas do Gartner, em torno de 2013 pelo menos 85% dos produtos comerciais do mercado embarcarão algum componente Open Source. Portanto, Open Source deve fazer parte das estratégias das empresas usuárias de software como das que produzem software.

O modelo Open Source e suas regras de liberdade, como liberdade para acessar seu código fonte e redistribuir cópias, alteradas ou não, abre espaço para novos modelos de negócio. O

tradicional modelo de ganhos econômicos baseados em comercialização de licenças pode ser substituído por ganhos em serviços, como distribuição, suporte e educação. A explicação é simples: no modelo comercial, todo o custo de desenvolvimento é do produtor de software, que para recuperar rapidamente este investimento, precisa vender licenças de uso do seu software. No modelo Open Source, fundamentado em desenvolvimento aberto e colaborativo, os custos são distribuídos por todos os participantes de uma comunidade. Não havendo a pressão para a recuperação rápida dos custos de desenvolvimento, pode-se desenhar modelos de negócios mais flexíveis.

Uma outra característica interessante é que sendo aberto e colaborativo, o modelo Open Source desacopla o software do controle exclusivo de uma única empresa. Qualquer usuário ou empresa pode acessar o código, distribuí-lo e operá-lo. Não existem pagamentos de licenças nem de royalties. Entretanto, esta mesma liberdade abriu espaço para a criação de novos negócios, como a distribuição de produtos Open Source empacotados. Um exemplo são as distribuições Linux. Na prática a distribuição é um serviço que uma empresa presta a outros, assumindo as tarefas de integrar componentes de software, testá-los e adicionar utilitários que auxiliem o processo de instalação e configuração.

Nestes últimos anos foi indiscutível que o modelo de Open Source amadureceu, bem como as percepções do mercado mudaram significativamente. Lembro que há alguns anos atrás ainda haviam muitas dúvidas básicas como, por exemplo, se Open Source era ou não um software de domínio público. Hoje está claro que Open Source não é um software de domínio público. Um software Open Source é um software licenciado, com algumas pessoas detendo os direitos sobre o código fonte ou sobre a marca. Por exemplo em <http://www.linuxfoundation.org/about/linux-foundation-trademark-usage-guidelines> podemos acessar as orientações para uso da marca Linux. Além disso existem diversos tipos de licenciamento, algumas mais restritivas e outras mais abertas. A licença GPL (General Public Licence), por exemplo, impede que o código seja privatizado por alguém e incorporado em algum projeto de código fechado. Já as licenças MIT e Apache permitem que o código seja privatizado e acoplado a softwares fechados. E, existe também a opção de dual-licensing, que tem se popularizado pela indústria de software.

No site da Linux Foundation (<http://www.linuxfoundation.org/publications>) existem uma série de artigos bem interessantes que mostram como as empresas podem obter um compliance entre as diversas licenças disponíveis. Uma comparação das diversas licenças Open Source pode ser encontrada no Wikipedia em http://en.wikipedia.org/wiki/Comparison_of_free_software_licenses.

Outra percepção que amadureceu no decorrer destes anos foi que imaginava-se que todo e qualquer projeto de Open Source apresentava inerentemente código de melhor qualidade que muitos dos softwares fechados. Em muitos casos, quando a comunidade era ampla o suficiente para analisar o código fonte e sugerir mudanças, era verdade. Mas para a maioria dos projetos Open Source onde as comunidades são pequenas e pouco atuantes, os códigos nem sempre são de boa qualidade. Outra fator que afeta significativamente a qualidade do código é o modelo de governança do projeto. Comunidades geridas por processos ditatoriais, com pouca motivação para seus membros, gera forks e a qualidade do código acaba sendo comprometida. Sabe-se hoje que a qualidade do código Open Source está

diretamente relacionada com fatores como maturidade e qualidade do modelo de governança do projeto, amplitude e atitude da comunidade, disciplina dos processos de detecção e correção de bugs e um road map bem planejado e não feito aleatoriamente.

Um ponto que me chama atenção é que hoje as idéias ideológicas que nortearam as primeiras ações de Open Source já ficaram para trás. Tecnologia e ideologia não devem ser misturadas. Para uma empresa, seja ela pública ou privada, a decisão de optar por uma tecnologia ou outra deve ser baseada no valor para a sua estratégia e o seu negócio e não por preferências ideológicas. Quando se misturou ideologia com tecnologia dizia-se que era “uma luta do bem contra o mal” e que de alguma forma o modelo Open Source era mais ético que o modelo comercial. É uma linha de pensamento que eu, pessoalmente, não concordo. Open Source não é fundamentado em ser mais ético ou menos ético, mas é um modelo de desenvolvimento de software colaborativo, que permite a criação de novos modelos de negócios, alternativos ao modelo tradicional de vendas de licença. Um livro que descreve bem isso e que li nos idos de 2001, mas ainda atual é “The Cathedral & the Bazaar”, de Eric Raymond.

Outra percepção que ao longo do tempo foi amadurecendo era que todo e qualquer projeto de implementação de Open Source trazia automaticamente redução de custos significativos. Nem sempre é verdade. O que acontecia? Na maioria das vezes comparava-se custos de aquisição entre softwares Open Source (zero) versus os dos softwares comerciais. Olhando-se exclusivamente pelo prisma dos custos de aquisição e manutenção, a alternativa Open Source é mais vantajosa. Mas analisando-se os custos de propriedade, que inclui custos de migração, integração com outros produtos, atualizações, suporte e educação, as diferenças nem sempre são significativas e motivadoras. Além disso, alguns softwares como Linux são distribuídos a custos bem acima de zero, devido ao trabalho dos distribuidores em integrar e testar os milhares de componentes que fazem parte de uma distribuição. Claro que uma empresa poderia fazer isso, mas o custo não seria zero, pois estaria alocando dezenas de técnicos que poderiam estar atuando em outros projetos. Na minha experiência com diversos projetos Open Source vi que a imensa maioria dos projetos que trouxeram benefícios econômicos palpáveis para as empresas foram os bem gerenciados, onde a escolha da alternativa Open Source foi feita com racionalidade, fundamentados por uma análise bem feita de retorno do investimento.

Em resumo, que lições podemos tirar destes anos de evolução e amadurecimento do modelo Open Source? Sem sombra de dúvida que Open Source está afetando a indústria de software como um todo. Projetos inovadores são baseados neste modelo e praticamente toda a indústria de software está em maior ou menor grau, envolvida com Open Source. Um exemplo prático é o comprometimento de uma empresa de grande porte como a IBM, que pode ser visto em <http://www-03.ibm.com/linux/ossstds/oss/ossindex.html>.

Alguns domínios de aplicação de software, como sistemas operacionais e web servers já são bastante populares como alternativas Open Source. Basta ver o sucesso alcançado pelos projetos Linux, Eclipse e Apache. Outros segmentos ainda estão mais incipientes, mas ao longo do tempo o modelo Open source vai se entranhar por toda a indústria.

As idéias românticas da “luta do bem contra o mal” estão agora racionais. Existe parceria entre os modelos de código aberto e fechado, com sinergia entre eles. O modelo dual-licencing é um exemplo concreto desta sinergia. O que precisamos é continuar amadurecendo nossas idéias e adotando Open Source quando for a solução mais adequada. Temos ainda muita pista para correr.

A proposta deste blogbook é a de compartilhar idéias e colaborar para o debate de como e quando adotar Open Source nas empresas. Ele é fruto de projetos, estudos e conversas com executivos de diversas empresas, resumidos em posts publicados no meu blog entre 2007 e 2010.

Para oferecer uma visão cronológica e histórica da evolução do Open Source, os posts foram divididos em blocos, cada um deles cobrindo um ano, de 2007 a 2010. Adicionei também alguns textos publicados na minha coluna na revista eletrônica Espírito Livre (<http://www.revista.espiritolivres.org/>), publicação que recomendo a leitura, pela qualidade de seu conteúdo.

Procurei manter estes posts, na medida do possível iguais aos publicados originalmente. Corrigi alguns crassos erros ortográficos, que passaram em branco quando foram inicialmente levantados.

Todos os URL que aparecem nos textos foram acessados, checados e quando necessário, atualizados durante a preparação deste blogbook, mas como a Web é extremamente dinâmica, existe a grande possibilidade de alguns destes URL já não existirem ou terem sido alterados, pelos quais pedimos antecipadamente nossas desculpas. Adicionalmente, li alguns bons livros sobre o assunto, tomei a liberdade de sugerir sua leitura como material de pesquisa e estudo adicional. Esta relação bibliográfica foi inserida no fim do blogbook.

Lembro também que as opiniões expressas neste blogbook e em toda a série de blogbooks (como foram os posts publicados no blog original) são fruto de estudos, análises e experiências pessoais, não devendo em absoluto serem consideradas como opiniões, visões e idéias de meu empregador, a IBM, nem de seus funcionários. Em nenhum momento, no blog e aqui, falo em nome da IBM, mas apenas e exclusivamente em meu nome.

Cezar Taurion

Janeiro de 2011

2007

O assunto Open Source não é absolutamente novidade. O conceito já existia desde a formação do grupo de usuários SHARE, em 1952, para desenvolver e compartilhar código de programas para o computador de primeira geração IBM 701. Também sempre foi uma prática no meio acadêmico. Mas, como o advento da Internet e principalmente com o sucesso do Linux (criado em 1991), o Open Source começou a se disseminar a partir da segunda metade dos anos 90. Esteve no auge das buscas nos search engines de 2001 até cerca de 2006. Vejam abaixo o gráfico gerado pelo Google Trends, de 2004 até agora. Em 2007, ainda despertava muito interesse, e já começávamos a ver mais ação e menos conversas. Entretanto, ainda havia muito receio e algum desconhecimento por parte de gestores de TI, daí ter abordado o tema em vários posts, discutindo principalmente seu modelo e sua aplicabilidade nas empresas.



Começando a trocar idéias (Janeiro)

Este é o foi o primeiro post sobre Open Source e o primeiro post do blog.

Estou realmente muito entusiasmado pela possibilidade de trocarmos idéias sobre a indústria de software. E para começar nosso relacionamento, vamos falar um pouco de um tema muito importante, que é Open Source e seus modelos de desenvolvimento.

Acredito que ninguém tenha ainda a clara percepção de até onde o modelo Open Source poderá ir. Muitas vezes pensamos de forma linear, baseados nas nossa próprias experiências. Mas talvez estejamos vendo um fenômeno que muda significativamente a nossa percepção e realidade da indústria de software. Vemos exemplos fantásticos de uso de “inteligências coletivas” como a enciclopédia Wikipedia ([//en.wikipedia.org](http://en.wikipedia.org)), onde qualquer um de nós pode colaborar e acabou-se gerando uma enciclopédia que não deve nada às mais tradicionais. Palavra da respeitada revista Nature em um artigo muito interessante que comparou a Wikipedia com a famosa Britannica.

Se imaginarmos o desenvolvimento de um projeto deste tipo nos moldes tradicionais o que teríamos? Para começar seria uma especificação fantasticamente ambiciosa de criar um sistema de autoria colaborativa que abrangesse todo o conhecimento humano, envolvendo colaboração de centenas de milhares de pessoas em todo o mundo, que poderiam colaborar de forma livre e espontânea (e voluntária, sem ganhos financeiros) sobre qualquer tema. Imagine desenhar os workflows para gerenciar o processo de captação de artigos, de validar o seu conteúdo (checando consistência e ortografia, em múltiplas linguagens), gerenciar as alterações, implementar controles de segurança e direitos de acesso, e tudo o mais que um complexo sistema como esse demandaria? Além disso, imagine a capacidade computacional para armazenar este imenso conteúdo que cresce rapidamente (são mais de 1500 novos artigos escritos por dia) e ao mesmo tempo preservando todo o histórico dos textos, pois cada colaboração e suas subseqüentes modificações deve ser preservada por toda a vida.

Se tentássemos desenvolver este sistema da forma tradicional é provável que ainda estivéssemos debatendo exaustivamente suas especificações. Mas, o que aconteceu? Simplesmente começou-se a fazer a enciclopédia. Foram criadas algumas normas e procedimentos (regras de conduta, que são baseadas fundamentalmente em comportamento social), especificou-se os formatos para criação de textos, definiu-se uma visão clara e consistente e lançou-se o projeto ao cyberspace! Claro que o projeto não é perfeito. Muitas regras foram criadas e estão sendo adaptadas ao longo do tempo, ajustadas pela própria comunidade. Um ponto interessante: não existe um gerente de projeto que dê ordens ou determine o que pode ou não ser inserido na enciclopédia. A própria comunidade é que resolve por si, fazendo as modificações, corrigindo erros e até mesmo eliminando textos considerados preconceituosos ou parciais.

Para um projeto destes dar certo, é importante quebrar paradigmas. A rede social é um fator que não pode ser ignorada. A Internet faz com que cada desenvolvedor esteja a apenas um clique de distância um do outro, não importa a real distancia geográfica que os separa. Isto muda muita coisa. Projetos como o Linux e o Apache são exemplos de como o processo colaborativo pode transformar uma indústria.

Este impacto pode ser visto na última lista das 100 marcas mais valiosas do mundo que a Business Week publicou recentemente em conjunto com a empresa Interbrand, consultoria especializada em avaliação de marcas. A Microsoft, segundo os analistas perdeu valor de marca por uma combinação de três fatores: o avanço do Google, a expansão do Linux e o afastamento progressivo de seu fundador Bill Gates. A propósito, os servidores do Wikipedia e do Google rodam Linux. Portanto, a combinação de Open Source e “inteligência coletiva” podem estar embutindo um novo mundo. Vamos acompanhá-lo e debatê-lo aqui, com vocês.

Qualidade do Software Open Source (Janeiro)

Ainda encontramos profissionais que se mostram apreensivos quanto a qualidade do código dos softwares Open Source. Questionam se estes softwares tem realmente qualidade que atenda às necessidades de um exigente ambiente corporativo.

Existem dois métodos básicos de desenvolvimento de software: os princípios catedral e bazar. Estes nomes foram cunhados a partir do célebre trabalho “The Cathedral and the Bazaar” de Eric Raymond. O método batizado de catedral é baseado no planejamento centralizado, com evolução top-down e rígido relacionamento entre a gerencia e os desenvolvedores, quanto a prazos, metodologias adotadas e tarefas, dentro de uma hierarquia organizacional. O desenvolvimento é interno à empresa e apenas nos ciclos de teste alfa e beta é que o produto é exposto ao mercado, através de uma restrita e controlada comunidade de usuários (individuais ou empresas clientes) que se prontificam a cooperar nos testes e depurações. Mas todo o código fonte é proprietário e fechado ao mundo externo e restrito apenas aos olhos dos desenvolvedores da empresa que produz o software. É o método tradicionalmente adotado pela indústria de software em seus produtos comerciais. Seu nome deriva do processo de construção das catedrais na Idade Média. O princípio bazar, como o nome indica, é baseado em uma forma mais livre de desenvolvimento, sem centralização do seu planejamento e execução. É o modelo típico do Open Source. O desenvolvimento é efetuado em rede, por uma comunidade de desenvolvedores voluntários, na maioria das vezes sem vínculos entre si, em uma organização informal. A comunicação é efetuada pela Internet, virtualmente sem fronteiras geográficas e apenas existem alguns princípios que regulam o trabalho. A liderança do projeto não é definida de maneira prévia e formal, mas emerge naturalmente pelos méritos de um determinado membro da comunidade de voluntários.

Esta é uma característica diferenciadora do movimento Open Source: a sua exploração de forma inteligente do poder da produção colaborativa. O que torna este poder extraordinário é a capacidade de melhorar com o tempo, curando-se organicamente, como se o enorme exército de colaboradores de um projeto como o Linux fosse um sistema imunológico, sempre vigilante e ágil na reação a qualquer coisa que ameace o organismo. E apesar dos receios naturais causados por um modelo de desenvolvimento inovador para os padrões tradicionais, os projetos de Open Source não descambam para a anarquia, mas mantêm uma coesão impressionante.

Os códigos são revisados pelos próprios pares e geralmente o melhor código é selecionado (meritocracia). Como não existe um departamento de marketing influenciando prazos, o ritmo de desenvolvimento é direcionado pela disponibilidade de tempo e dedicação dos desenvolvedores voluntários. O método bazar gera uma forte tendência de gerar código de alta qualidade. O código é lido e analisado por diversos, as vezes centenas, de desenvolvedores, o que acelera o processo de depuração e correção de erros. Depois, não existe pressão de prazo, o que permite mais ciclos de depuração. A decisão de liberar o código é fruto de consenso do grupo e não uma imposição de marketing, como muitas vezes ocorre em um projeto comercial.

O resultado é um software de alta qualidade, como inúmeras estudos independentes tem demonstrado. Portanto, a preocupação com a qualidade é basicamente decorrente do desconhecimento do que é realmente Open Source do que de fatos reais.

A economia do modelo Open Source (Fevereiro)

Vamos falar um pouco de Open Source do ponto de vista econômico.

Um economista, Alfred Eichner, em seu livro “The Megacorp Oligopoly”, descreveu um modelo de definição de preços em mercado em situações de oligopólio e monopólio.

Para Eichner, em uma indústria oligopolizada, o preço pode ser representado por:

$$P = \text{custo variável médio} + ((\text{custo fixo} + \text{margem}) / \text{quantidade}).$$

Os custos variáveis médios constituem os custos unitários de produção, como matéria prima e mão de obra. No caso da indústria de software, praticamente os custos se concentram em mão de obra e como matéria prima teríamos as tecnologias disponíveis para construção do software, como estações de trabalho e softwares de apoio ao desenvolvimento.

Nos custos fixos estão incluídos custos com remuneração da gerência e também os dividendos a serem pagos aos acionistas. Uma característica de uma empresa oligopolista é a separação entre a propriedade e direção, o que faz com que quem esteja na direção obrigue-se a remunerar bem os donos do capital, para se manter nela.

A quantidade é o resultado da multiplicação entre a capacidade técnica instalada, que representa a capacidade produtiva da empresa e o percentual da capacidade instalada. No caso da indústria de software, não existem muitas restrições em termos de capacidade. A indústria de software não precisa de uma linha de produção física e nem de estoque e custos logísticos como uma indústria automobilística. Assim, na fórmula, esta variável pode ser ignorada.

Por fim, a margem corresponde à fonte interna de recursos para financiar as despesas de investimento necessárias para que a empresa concretiza seu objetivo de maximização de crescimento no longo prazo. Em termos contábeis, a margem será formada por lucros retidos, depreciação, gastos com propaganda e gastos com pesquisa e desenvolvimento.

O importante para o modelo é que a margem é predefinida pela empresa líder da indústria, de acordo com suas necessidades de investimento em capital fixo, pesquisa e desenvolvimento, e propaganda. Em outras palavras, nesse modelo, o preço é definido pela empresa a partir de seus custos e da margem pretendida.

Vamos analisar o mercado de suítes de escritório e sistemas operacionais de desktop. É um mercado com uma estrutura claramente monopolista, com uma única empresa dominando mais de 90% do mercado. É também um mercado que está claramente em situação de excesso de desempenho. O excesso de desempenho do produto é uma oferta de funcionalidade além da demanda do mercado. Para uma imensa base de clientes, com menores exigências tecnológicas, a funcionalidade (e o conseqüente preço) da combinação da última versão oferecida pelo fabricante está acima das suas demandas. Neste contexto, a base de competição se desloca para preço.

Adaptando a equação de Eichner ao modelo de Open Source, veremos a seguinte situação: os custos variáveis (custo de desenvolvimento) diluem-se pela comunidade que colabora com o desenvolvimento do software, levando, portanto a esta variável a ter valor mínimo na equação. Os custos fixos também são distribuídos pelos mantenedores dos projetos de softwares abertos. Os investimentos em pesquisa e desenvolvimento também são diluídos pela comunidade. Os custos fixos, portanto, podem ser considerados como tendo valores mínimos na equação. Margem também inexistente, pois a comunidade não tem que prestar contas a acionistas.

Assim, o modelo de Eichner quando adaptado ao modelo de Open Source nos leva naturalmente a um cenário de preço mínimo. Quando um determinado segmento de software estiver caminhando na direção da competição por preço, como no caso das suítes de escritório, e a alternativa de Open Source atender às demandas de funcionalidade, credibilidade e conveniência, o fator preço torna-se o diferencial. Cresce então o interesse por alternativas muito mais baratas baseadas no modelo Open Source. O que faria então então uma empresa dominante? Procurar manter a posição através do aprisionamento forçado, evitando ao máximo adotar padrões abertos que diminuam as barreiras de saída para sua base instalada!

Open Source alcançando a maturidade (Fevereiro)

O movimento de Open Source já é uma realidade na indústria de software. As dúvidas iniciais já foram em grande parte sanadas e hoje não se questiona mais a validade deste modelo alternativo de negócios, que fornece desafios, benefícios e com certeza novas oportunidades, diferentes dos modelos convencionais.

Mas ainda encontramos alguns questionamentos. Um deles diz respeito a questão da segurança, uma vez que os modelos de licença de Open Source divulgam o código fonte. Talvez este debate ainda esteja vivo porque o modelo de Open Source impacta diretamente a receita (e lucratividade) de empresas de software baseadas exclusivamente no modelo de licenças proprietárias. Por exemplo, no relatório anual da Microsoft (www.microsoft.com/msft) está claramente dito, na seção “Risk Factors”: “In recent years certain “open source” software business models have evolved into a growing challenge to our license-based software model”. Também fica claro o potencial de perda financeira: “To the extent open source software gains increasing market acceptance, sales of our products may decline, we may have to reduce the prices we charge for our products, and revenue and operating margins may consequently decline”. Nada mais natural que estas empresas, lutando pela sobrevivência de seu modelo de negócios, questionem o Open Source, levantando dúvidas e buscando eventuais pontos de falha.

A disponibilidade do código fonte, mortal para o modelo de licenças proprietárias é então alvo de ataque: sua disponibilização não seria um risco para a segurança, uma vez que hackers podem livremente vasculhar o código fonte e identificar vulnerabilidades?

Bem, uma inferência errônea é que o modelo de código fonte fechado seria inerentemente mais seguro. Os freqüentes casos de vulnerabilidades que vemos em softwares proprietários demonstram que esta não é uma assertiva verdadeira. Um exemplo: diversas pesquisas (e a vida prática) tem demonstrado que o Apache apresenta menos falhas de segurança que o Internet Explorer da Microsoft. Esconder o código fonte não garante segurança. A maioria dos ataques dos hackers tem sido feito em cima de softwares como Windows e Explorer, cujos códigos fontes são fechados. Existem sofisticados “disassembly tools” que facilitam aos hackers derivar o código fonte do executável. Quem está interessado em explorar uma vulnerabilidade pode usar estas ferramentas, sem precisar do acesso direto ao fonte.

E quando a disponibilidade do fonte, como proposto pelo Open Source? O efeito, ao contrário do que pensam os críticos, é positivo, pois a comunidade consegue detetar vulnerabilidades e corrigi-las rapidamente, pois são muitos olhos atentos e não apenas uma pequena equipe interna de uma única empresa. No sistema fechado os usuários ficam a mercê da capacidade do proprietário do software em detetar e corrigir a falha.

Portanto, o modelo de Open Source está cada vez mais consolidado e lutar contra ele seria como defender de forma fanática as máquinas de escrever contra os microcomputadores. Uma luta inglória. Pois, queiramos ou não, as máquinas de escrever tornaram-se obsoletas!.

Usando Open Source dentro de casa (Março)

Projetos de Open Source já demonstraram sua viabilidade em dezenas de produtos, como Linux, Samba, PHP, Apache e muitos outros. O que é realmente inovador no Open Source é o modelo de desenvolvimento colaborativo. E aqui faço um questionamento: porque as empresas não adotam este modelo dentro de casa, no seu processo de desenvolvimento?

Vamos pensar juntos: o modelo de desenvolvimento tradicional tem mostrado muitas falhas...Uma parcela significativa dos projetos atrasa ou acabam sendo cancelados. E colocar mais desenvolvedores atrapalha mais ainda! Ora, mas o modelo Open Source utiliza centenas e às vezes milhares de desenvolvedores, sem gerências burocráticas e as coisas funcionam. Será que não podemos pensar em adotar alguns dos processos utilizados por este modelo? Não temos aí algo para pensar?

Na minha opinião o modelo Open Source pode trazer muitos benefícios a uma organização de TI. Por exemplo, incentiva a comunicação e o compartilhamento de idéias entre os desenvolvedores da empresa. A essência do processo colaborativo é a comunicação e a colaboração. Um processo colaborativo usa intensamente ferramentas como wikis, lista de discussão, blogs, etc.

Aumenta também a transparência do projeto, permitindo que mais pessoas, de fora da equipe, como usuários, se envolvam e opinem sobre o projeto. A documentação também é desenvolvida em colaboração e até mesmo pelos próprios usuários. Além disso, os usuários e outros desenvolvedores podem colaborar ativamente no processo de testes e validação.

Um projeto típico de Open Source compreende uma pequena equipe de “core-developers”, que respondem pela maioria do código, um numero bem maior de desenvolvedores contribuidores (contribuições esporádicas) e um grande número de outros colaboradores, como usuários e outros desenvolvedores, que podem colaborar com sugestões e na depuração. Este intercambio profissional, transparente, com uso intenso de “peer review” tende a aumentar a qualidade do código, pois incentiva os desenvolvedores a buscarem um nível maior de qualidade. Afinal, todo código é visto por outros desenvolvedores.

Claro que algumas características típicas do Open Source não se adaptam a um projeto interno. Por exemplo, deixar que os desenvolvedores definam os prazos por si...Afinal, eles têm que respeitar a demanda da empresa, que exige determinado prazo para o sistema entrar no ar.

Mas, enfim, antes de acharmos a idéia absurda, que tal estudarmos um pouco mais os processos de desenvolvimento colaborativo, usado pelo modelo Open Source e validar se alguns deles não poderiam ser usados dentro de casa?

Ganhando dinheiro com Open Source (Março)

Há pouco tempo falar em ganhar dinheiro com Open Source era visto como algo inimaginável. Software Livre tem que ser gratuito diziam alguns radicais. Hoje o mercado está muito mais maduro. Já se compreende que Open Source é um modelo de desenvolvimento de software que permite criar novos modelos de negócios.

Muitos negócios em torno do Open Source são negócios bem sucedidos que geram receita. Temos aí a Red Hat, Suse Novell e MySQL como alguns exemplos.

Outro dia um parceiro nosso me procurou para discutir se valia a pena ou não abrir seu código fonte. Discutimos o assunto e como devem existir diversos outros ISVs que estão pensando se vale a pena ou não transformar seu negócio em Open Source, creio que alguns dos comentários que surgiram neste papo devem ser compartilhados.

Criar um modelo de negócios baseado em Open Source não é simplesmente abrir código fonte e pronto. É necessário criar um balanceamento entre os interesses da comunidade e do próprio negócio. Tem que ser uma relação ganha-ganha. Existem muitas questões a serem discutidas. Por exemplo, qual o modelo de licenciamento? GPL? Baseado em BSD? Quem vai desenvolver e manter o código, a própria companhia ou a comunidade?

Também recomendo analisar os seguintes fatores:

- a) Qual a maturidade do seu projeto? É um código estável e maduro, pronto para ser usado, ou é excessivamente instável, cheio de bugs? Dependendo do estágio, você pode esperar contribuições diferentes da comunidade. Um código estável demanda poucas e simples contribuições. Por sua vez, um código instável demandaria um esforço bem maior da comunidade. E ela estaria disposta a este esforço?
- b) Tamanho da comunidade. Um software de utilização restrita vai despertar pouco interesse da comunidade e, portanto os benefícios esperados de contribuições, sugestões, etc, será muito pequena ou inexistente. Softwares comoditizados, como sistema operacional, banco de dados mais simples, suítes de escritórios são mais fáceis de atrair interesse, pois muita gente conhece suas funcionalidades. Softwares radicalmente diferentes terão muita dificuldade de atrair desenvolvedores que realmente possam contribuir com idéias e códigos. Um projeto de Open Source que não consegue atrair interesse da comunidade simplesmente vai para o buraco...
- c) Maturidade do mercado. O software vai atender a uma demanda existente ou será necessário criar este mercado? Criar mercado demanda ações que os modelos Open Source provavelmente não atenderão adequadamente.
- d) O seu mercado compreende Open Source e está disposto (ou pode) a trocar modelo de licenciamento por serviços? É um mercado que compra serviços e consultoria? Existem nichos de mercado a serem explorados?

O modelo Open Source demanda algumas mudanças na organização, nas estratégias de marketing e comercialização, e nos serviços. É necessário criar um modelo de distribuição diferente do modelo atual. É necessário criar vínculos estreitos com a comunidade e é

absolutamente fundamental entender que o modelo de serviços é diferente do modelo de licenciamento. Você também deve considerar que nem todo o código fonte necessita ser aberto à comunidade. Mas, quais partes serão abertas e quais não serão abertas?

Enfim, Open Source é um modelo de negócios que pode trazer excelentes resultados, mas para ter sucesso tem que se entender muito bem os seus conceitos e peculiaridades.

Desenvolvimento em Open Source: porque alguns softwares despertam interesse e outros não? (Abril)

Embora Open Source venha se expandindo exponencialmente, ainda existem alguns mitos que precisam ser derrubados.

Bem, primeiro vamos lembrar que nem todo projeto de open source terá sucesso. Muitos não atraem interesse da comunidade e tendem a desaparecer. Para cada projeto de sucesso existem milhares de projetos que fracassam. Poucos projetos despertam interesse e atraem um número expressivo de voluntários. Muitos projetos são iniciativas isoladas e permanecem desta maneira até desaparecerem. Não sobrevivem à versão alfa 0.1. O site http://www.unmaintained-free-software.org/wiki/Main_Page é um interessante wiki com projetos de open source órfãos, ou seja, não tem mantenedores, e que aponta mais de 90 projetos em busca de adoção. Portanto, não se pode replicar automaticamente o sucesso do Linux, do Apache e alguns outros a todos os demais projetos de open source.

Algumas pesquisas tem sido efetuadas para identificar porque alguns projetos Open Source despertam interesse e outros não, e já começamos a dispor de informações razoavelmente confiáveis para melhor compreendermos este fenômeno. À medida que estudamos o tema open source, compreendemos que existem muitas diferenças entre a teoria e a prática. O mundo real é diferente do mundo romântico e idealista visualizado por alguns ideólogos do open source.

No modelo tradicional, o desenvolvimento de um software segue um padrão bastante conhecido. Uma empresa escreve o software segundo metodologias e objetivos claros e bem definidos, e o comercializa. O código fonte é propriedade particular e considerado “segredo de estado”. O open source tem outro paradigma. O código fonte é desenvolvido e mantido por uma comunidade de voluntários e está disponível a todos.

Outros estudos sustentam que os ganhos indiretos (até mesmo o desejo de um mundo melhor...) e a busca por status são os principais motivadores dos voluntários. É o que os motiva a investir seu tempo e até dinheiro para contribuir voluntariamente em um projeto de open source. Existe também um aspecto pouco conhecido: o papel e a motivação do mantenedor do projeto de open source. Este indivíduo é o responsável por assumir papel de liderança e responsabilidade, com muito maior investimento de tempo e dedicação que os demais voluntários. Por que assume este papel?

Estudar este fenômeno não é fácil. Nem todos os projetos de Open Source apontam de maneira clara e estruturada seus autores. Mas analisando-se o código fonte a partir de sites repositórios de movimentos de projetos Open Source como SourceForge (<http://sourceforge.net/>) pode-se tirar algumas conclusões relevantes. Uma inspeção nos seus projetos cadastrados nos mostra que uma minoria deles se encontra no estágio de maturidade.

Alguns estudos (“The Ecology of Open Source Software Development, de K. Healy e A. Schussman; “Cave or Community? An empirical examination of 100 mature open source projects”, de S. Krishnamurthy; “The nature and composition of the Linux kernel developer

community: a dynamic analysis” de R. Ghosh e P. David; e “FLOSS: Free/Libre/Open Source Software Study”, entre diversos outros, mostraram resultados interessantes quanto ao processo de produção de softwares livres. Uma pesquisa no Google traz artigos muito interessantes e vale a pena um investimento de tempo nesta empreitada.

Entre outras informações, identificou-se que a contribuição da comunidade é distribuída de forma desigual entre os colaboradores. Na maioria dos projetos de Open Source, uma pequena parcela de desenvolvedores contribuiu com a maior parte do código. Em números, cerca de 10% dos voluntários contribuiu com quase 75% de todo o código escrito. O fato real é que apenas um pequeno grupo de desenvolvedores arca com maior parte do código.

Os estudos mostraram também que em média cerca de cinco desenvolvedores contribuíam para cada projeto. Em número de linhas de código, a grande maioria dos projetos de open source são softwares de pequeno tamanho. Não surpreendentemente observou-se que o número de colaboradores aumenta diretamente com o aumento do tamanho do projeto.

Verificou-se também que a maioria estava sendo desenvolvida por um grupo muito pequeno de voluntários. Os estudos também chegam a observações interessantes: projetos com muitos download não significam necessariamente projetos com intensa atividade de programação. Softwares como sistemas operacionais atraem o interesse de desenvolvedores, que por sua vez geram muita atividade de programação. Por outro lado, softwares mais focados no usuário final demandam mais downloads de interessados em seu uso, mas não atraem muitos voluntários para seu desenvolvimento e manutenção.

Outra observação curiosa é que na maioria dos projetos que envolvem comunidades mais amplas de colaboradores, como o Linux, a contribuição ao projeto é massiva, mas a colaboração entre a comunidade acontece apenas entre grupos pequenos de desenvolvedores, que estejam trabalhando na programação dos mesmos pedaços de código. Na prática a colaboração acontece apenas entre grupos reduzidos, envolvidos nas mesmas tarefas.

Aglutinando-se as informações acima se conclui que as maiorias dos projetos de software abertas são de pequeno tamanho e a maior contribuição vem do seu próprio autor e que esta sua contribuição e entusiasmo é crucial para manter o projeto vivo e atraente para novos colaboradores. O modelo de desenvolvimento em comunidade depende de alguns fatores para deslanchar. Um deles é a atratividade do projeto. Quanto mais popular e atrativo o projeto, maior a comunidade envolvida. Por outro lado, projetos com pouca atração não conseguem aglutinar uma comunidade significativa.

Estes estudos nos indicam as primeiras pistas que nos ajudam a identificar porque um projeto obtém sucesso e outros fracassam. Uma primeira observação contraria a teoria que propõe que o open source é desenvolvido por uma comunidade de adolescentes e estudantes com pouca experiência em programação: os projetos de sucesso são fruto principalmente da contribuição de desenvolvedores profissionais. Conclui-se que open source é coisa de profissional e não de amadores.

Outra observação prática demonstra que um fator crítico de sucesso é o papel executado

pelo mantenedor. Uma clara, sólida e reconhecida liderança é essencial ao sucesso do projeto. E terceiro, ao contrário do que se imagina popularmente, um projeto de sucesso depende de uma estrutura hierárquica para ser concluído a contento. Deve haver uma clara e reconhecida regra de subordinação entre o mantenedor, os líderes e os demais contribuidores para que o projeto obtenha sucesso. O exemplo do Linux é emblemático. Existe uma hierarquia, com Linus Torvalds decidindo em última instância que patches serão aceitos na nova versão do kernel e quais não o serão. Ou seja, o projeto de open source não é caótico e desorganizado. Por outro lado a hierarquia não é formal, mas é aceita por fatores como competência e méritos profissionais. É meritocrática por excelência.

Um dos aspectos ainda pouco conhecidos dos projetos de open source é a real motivação dos voluntários. O que os motivam a dedicar tempo e esforço a colaborar com um projeto em que aparentemente não obtém ganhos financeiros? Conhecer esta motivação é importante para conhecer a sustentabilidade do modelo.

Alguns esforços tem sido feitos nesta direção. Pesquisas com comunidades de desenvolvedores já tem apontado as primeiras pistas. Claro que são ainda incipientes e apenas com a experiência e estudos mais aprofundados é que chegaremos a respostas mais precisas.

Uma forte motivação é a intrínseca, onde o desenvolvedor desenvolve atividades que considera agradáveis. Como não existem pressões por prazo ou cobranças por tarefas além da capacidade individual (o limite da competência e do desafio técnico é determinado pelo próprio desenvolvedor), participar de um projeto de open source torna-se um trabalho agradável. É diferente de um trabalho enquanto empregado, pois neste as pressões por prazo e exigências acima da capacidade técnica tornam-se altamente estressantes. Existe também a motivação causada pelo senso de participação em uma comunidade, com suas obrigações e recompensas. Participar de uma comunidade de open source e ser reconhecido pelos pares torna-se um fator altamente motivador.

Mas existe também a motivação extrínseca, que se reflete em compensações como evolução profissional (chance de participar de um projeto conhecido internacionalmente), evolução técnica (chance de desenvolver técnicas de programação que não podem ser obtidos em trabalhos regulares) e obviamente oportunidades de melhor empregabilidade, pela exposição a uma rede de contatos muito maior e influente. Existe também, é claro a compensação financeira pela oportunidade de explorar serviços em torno do open source.

As pesquisas demonstram claramente que uma comunidade envolvida em um open source de sucesso abriga colaboradores com diversas e variadas motivações. É uma comunidade heterogênea pela sua própria natureza e as motivações não podem ser focadas apenas em um ou outro aspecto.

À medida que o projeto do open source evolui e obtém uma maior utilização e reconhecimento pelo mercado, torna-se parte importante do conjunto de softwares essenciais às empresas, como o Eclipse, Apache e o Linux. Nesta situação identifica-se que uma parcela significativa dos seus desenvolvedores são remunerados, muitas vezes pelas empresas da própria indústria. Um bom exemplo de como uma empresa líder de indústria,

como a IBM, está atuando em Open Source, pode ser visto em www.ibm.com/developerworks/opensource. Sugiro também fazer uma visita ao site do Linux Technology Center, em <http://www.ibm.com/linux/ltc/>. Empresas deste porte, pela experiência e tecnologia que agregam à comunidade, acabam por influenciar, direta ou indiretamente, os rumos do projeto.

Como não poderia deixar de ser, as pesquisas demonstram que os colaboradores que recebem remuneração (direta ou indireta) pela sua colaboração tendem a dedicar muito mais horas e energia ao projeto de open source que os que contribuem apenas por prazer e diversão.

Entender o processo Open Source abre novas oportunidades de desenvolvimento de projetos por parte de empresas de software, que mesmo sem capital suficiente, poderiam, sob o abrigo do modelo colaborativo desenvolver projetos que seriam impossíveis no modelo tradicional.

É indiscutível que as interações entre projetos de open source e empresas de software privadas estão se tornando cada vez mais comuns. O movimento do open source já não é mais visto como um romântico movimento ideológico e sim já é encarado como alavancador de modelos de negócios sustentáveis e complementares aos tradicionais modelos da indústria de software.

A (des)organização do modelo de desenvolvimento Open Source (Abril)

Outro dia participei de um debate sobre Open Source, onde um dos principais temas debatidos foi o modelo de desenvolvimento. Acho que vale a pena compartilharmos algumas idéias. Existem dois métodos básicos de desenvolvimento de software: os princípios catedral e bazar. Estes nomes foram cunhados a partir do célebre trabalho “The Cathedral and the Bazaar” de Eric Raymond. O método batizado de catedral é baseado no planejamento centralizado, com evolução top-down e rígido relacionamento entre a gerencia e os desenvolvedores, quanto a prazos, metodologias adotadas e tarefas, dentro de uma hierarquia organizacional. O desenvolvimento é interno à empresa e apenas nos ciclos de teste alfa e beta é que o produto é exposto ao mercado, através de uma restrita e controlada comunidade de usuários (individuais ou empresas clientes) que se prontificam a cooperar nos testes e depurações. Mas todo o código fonte é proprietário e fechado ao mundo externo e restrito apenas aos olhos dos desenvolvedores da empresa que produz o software. É o método tradicionalmente adotado pela indústria de software em seus produtos comerciais.

O princípio bazar, como o nome indica, é baseado em uma forma mais livre e colaborativa de desenvolvimento, sem centralização do seu planejamento e execução. É o modelo típico Open Source. O desenvolvimento é efetuado em rede, por uma comunidade de desenvolvedores voluntários, sem vínculos entre si, em uma organização virtual e informal. A comunicação é efetuada pela Web, sem fronteiras geográficas e existem apenas alguns princípios que regulam o trabalho. A liderança do projeto não é definida de maneira prévia e formal, mas emerge naturalmente pelos méritos de um determinado membro da comunidade.

Os códigos são revisados pelos próprios pares (peer review) e geralmente o melhor código é selecionado (meritocracia). Como não existe um departamento de marketing nem acionistas influenciando prazos, o ritmo de desenvolvimento é direcionado pela disponibilidade de tempo e dedicação dos desenvolvedores. O método bazar gera uma forte tendência de gerar código de alta qualidade. O código é lido e analisado por diversos, as vezes centenas, de desenvolvedores, o que acelera o processo de depuração e correção de erros. A decisão de liberar o código é fruto de consenso do grupo e não uma imposição de marketing, como muitas vezes ocorre em um projeto comercial.

Geralmente é necessário a figura de um líder ou mantenedor, que se encarrega de coordenar as mudanças e decidir (muitas vezes em colegiado) quando o produto pode ser liberado. Os ciclos de teste e revisões são constantes e o feedback é contínuo. Todos, a qualquer momento, podem interceder e comentar sobre o código, uma vez que este é livre e disponível a todos.

Os princípios de desenvolvimento em comunidade ainda não são totalmente conhecidos. Apenas nos últimos anos é que os primeiros estudos comportamentais de como a comunidade desenvolve software, gerencia e cobra tarefas, e como os contextos organizacionais são desenvolvidos é que começaram a ser desenvolvidos.

Mas já sabemos de algumas coisas. Uma é a importância do mecanismo de controle dos projetos. Ao contrário do modelo hierárquico, com rígidas normas de controle e subordinação, o modelo colaborativo tem outras características, algumas positivas e outras extremamente desafiadoras. Em um projeto fechado, típico de softwares proprietários, uma equipe de desenvolvedores profissionais são alocados a tarefas de acordo com suas especializações e gerenciados quanto ao cumprimento de prazos e orçamentos. No projeto de Open Source, a equipe é virtual, interage pela Internet (email, newsgroups, wikis, etc) e não existe subordinação direta. A participação dos desenvolvedores no projeto é voluntária e portanto não está submetida aos padrões de gerenciamento típicos dos projetos fechados.

Este modelo gera algumas incertezas, advindas do próprio modelo de contribuição voluntária. Como não existe uma rígida subordinação ou contrato empregatício, os colaboradores podem variar em muito de intensidade em suas colaborações. Podem contribuir intensamente como podem, sem prévio aviso, esquecerem em suas colaborações. Não existem planejamentos de produção e medidas de produtividade tipo “colaboradores estarão desenvolvendo quantas linhas de código por determinada unidade de tempo”. Também não existem divisões formais de trabalho, com prévia alocação de desenvolvedores à determinadas tarefas.

Para contrabalançar estas incertezas, alguns mecanismos de governança são adotados nos projetos de Open Source. Embora não exista hierarquia rígida, acaba-se criando um modelo de hierarquia informal, com o mantenedor e alguns membros assumindo posição de liderança e gestão do projeto. Cria-se um mecanismo de gestão centralizada, onde um grupo pequeno de colaboradores assume um papel executivo, responsabilizando-se pelas decisões e rumos do projeto. Esta organização é criada informalmente, impulsionada pela própria necessidade de existir uma estrutura razoavelmente estável, que oriente e direcione os esforços do restante da comunidade. Entretanto, ao contrário de projetos tradicionais, este papel não é assumido por escrito ou por posição hierárquica em uma empresa, mas por reputação ou conquista de espaço.

Na prática, acaba sendo criado um sistema hierárquico informal, baseado na meritocracia, com os desenvolvedores mais atuantes e experientes desenvolvendo as tarefas mais avançadas (codificação de novas funcionalidades e melhorias de desempenho) e os colaboradores menos experientes assumindo tarefas mais simples, como depuração de código.

Uma outra questão, inerente ao modelo colaborativo é o processo de seleção da codificação a ser inserida no projeto. Os colaboradores contribuem com seus códigos, e estes precisam passar por mecanismos de filtragem para serem aceitos e incorporados ao corpo do software. Não existem regras únicas. Os mecanismos podem variar de decisões autocráticas a sistemas de votação, sendo estes também variando de abrangentes (todos interessados podem votar) a votos restritos a um comitê selecionado de colaboradores.

O modelo colaborativo exige também mecanismos eficientes de controle de versões. Como a contribuição de novos códigos é livre, é necessário que o mantenedor gerencie cuidadosamente que pedaços de código deverão ou não ser incorporados ao software e decida quando o volume de modificações for suficiente para que seja liberada uma nova

versão.

Outro aspecto importante é que os projetos de Open Source devem ser modulares para permitir o trabalho de muitos colaboradores simultaneamente.

O método colaborativo quebra o tradicional paradigma criado por Fred Brooks em seu famoso livro “The Mythical Man-Month” onde ele observava que adicionar mais programadores a um projeto em atraso simplesmente aumentaria este atraso. Um programador em 12 meses não é igual a 12 programadores em um mês. A razão seria simples: um maior número de desenvolvedores envolvidos aumenta a complexidade e os custos da comunicação entre os integrantes do projeto. A complexidade aumenta exponencialmente enquanto que a quantidade de trabalho adicionada seria aumentada apenas linearmente. O método colaborativo simplesmente ignora este fato e propõe que quanto mais desenvolvedores são incorporados ao esforço de escrever código livre, mais eficiente é o produto gerado.

E quanto à correção de bugs? Bugs não são novidade no software. O próprio processo de construção de software é baseado na descoberta e correção de bugs. O processo de depuração de software é um processo cuja eficiência aumenta quase linearmente na direta proporção do número de depuradores envolvidos na tarefa. É um processo que tem muito a ganhar com o paralelismo de atividades, pois não demanda muito esforço de coordenação e gerenciamento. Os depuradores não precisam de muita interação entre si, apenas precisam ser coordenados por um responsável pela decisão de aprovar ou não as correções.

O método colaborativo potencializa o processo, pois permite que milhares de desenvolvedores analisem e testem peças de código, depurem os erros e submetam as correções aos coordenadores dos módulos do sistema. É um processo similar ao sistema acadêmico de revisão de textos científicos para publicação, conhecido como “peer review” ou revisão pelos pares, quando um ou mais revisores atestam a qualidade do texto e sugerem correções e modificações. Este processo garante o nível de qualidade e integridade dos textos científicos a serem publicados.

Entretanto, o modelo de desenvolvimento bazar muitas vezes negligencia alguns aspectos como a questão do prazo e definição clara do escopo. Muitos softwares Open Source começam com uma parte do código sendo disponibilizada e algumas vagas intenções de escopo. À medida que o software é desenvolvido comunitariamente, o próprio escopo muda sensivelmente. Prazo pode ser fundamental para alguns produtos de software. Como a contribuição é voluntária, o modelo colaborativo não pode forçar prazos. Assim pode-se questionar a validade do método bazar para desenvolvimento de softwares que necessitem de escopo e prazos bem definidos.

Com o tempo, diversas variantes do princípio colaborativo vem sendo adotados. Vemos hoje projetos de Open Source onde a gestão está a cargo de uma determinada empresa, que sustenta financeiramente a iniciativa, pagando desenvolvedores e custeando muitos dos esforços da comunidade de voluntários. Mas, mesmo neste caso, as decisões são tomadas pela comunidade e não pela empresa. Em outros projetos, a gestão está a cargo de uma fundação, sustentada pela doação de empresas e indivíduos.

O modelo de Open Source tem seu processo de desenvolvimento efetuado por métodos que em muitas vezes contradizem os parâmetros básicos da engenharia de software, aprendidos nas salas de aula. É uma quebra de paradigmas...Mas tem dado certo, basta ver os inúmeros projetos de sucesso. Por que não olhar com mais atenção este modelo?

OpenOffice survey (Abril)

Recentemente li o relatório “IDC and OpenOffice.org Survey”, publicado em novembro de 2006. Este relatório contém uma pesquisa bem detalhada feita com usuários do OpenOffice (foram 5819 pesquisados, que responderam afirmativamente a questão se haviam usado OpenOffice alguma vez no ultimo ano) e gostaria de compartilhar com vocês algumas das informações.

Entre as dezenas de perguntas, destaco:

1) O sistema operacional mais usado pelos usuários do OpenOffice é o Windows, com 72% dos usuários domésticos e 62% dos usuários que usam OpenOffice no trabalho. Linux aparece em segundo, mas principalmente entre os usuários domésticos (20%).2) O OpenOffice é mais usado em casa do que no trabalho. Na minha opinião pessoal porque em casa (a maioria dos respondentes são dos EUA e da Europa), o custo zero para seu uso (apenas o tempo de download) quando comparado a uma versão proprietária e paga, é sem sombra de dúvida um grande atrativo. Além disso, a maioria dos usuários domésticos não demanda recursos sofisticados que só existem no Office da Microsoft, como planilhas elaboradas...O OpenOffice atende perfeitamente.3) Com relação ao uso nas empresas, 14,7% disseram que a utilização do OpenOffice pode ser vista como significativa. É um número pequeno e talvez reflita a dificuldade que as empresas enfrentam diante de mudanças culturais... Entretanto, mais de 23% dos pesquisados afirmaram que suas empresas pretendem aumentar o uso do OpenOffice. Bom sinal!

Minhas conclusões: O OpenOffice ainda tem muito espaço a percorrer...Vemos seu uso muito incentivado em governo e em empresas públicas.Nas privadas seu uso está mais concentrado nas empresas de pequeno a médio porte, com raras exceções de empresas de grande porte...Acredito também que seu uso irá aumentar à medida que a padrão ODF se dissemine. O OpenOffice é um dos diversos softwares que suportam o ODF.

Para usuários domésticos é claramente um atrativo, pois não demanda compra de licenças.Entretanto, devido à alta incidência de pirataria no Brasil, as diferenças de custo acabam ficando pequenas...Compara-se um custo zero contra cinquenta ou sessenta reais de um Office pirata. Aliás, pirataria é uma sombra que paira sobre a indústria de software é a pirataria, prática ilícita, que é simplesmente o uso, ou a cópia e distribuição de produtos comerciais sem autorização. É uma atitude ilegal, mas infelizmente muito popular, ocorrendo com frequência na indústria de computadores “cinza”, montados por empresas de “fundo de quintal”.

De maneira geral algumas estimativas apontam que a pirataria no Brasil prejudica fortemente a indústria de softwares de consumo, que deixaria de faturar algumas centenas de milhões de dólares. A diferença entre os softwares instalados (demanda) e o software legal (fornecimento) é igual à estimativa de softwares pirateados. A pirataria é medida como a quantidade de softwares instalados sem licença oficial.

Entretanto alguns estudos apontam que este valor corresponde à expectativa de lucro caso

não houvesse nenhuma outra alternativa. No modelo tradicional de software de massa (voltado ao consumidor), a evolução do software é muitas vezes direcionada pelas estratégias de marketing do fornecedor, que constantemente cria novas funcionalidades, nem sempre necessárias aos usuários. Algumas pesquisas de mercado demonstram que uma grande parcela das funcionalidades de softwares de escritório não são utilizadas. O modelo cria também um processo de obsolescência programada, impedindo a aquisição de versões anteriores, mesmo que estas atendam perfeitamente bem as necessidades de um grande contingente de usuários. O modelo não permite a comercialização de softwares usados, como ocorre na indústria automobilística. Outra característica deste modelo é que muitas vezes as decisões de liberação de versões são direcionadas pelas estratégias de marketing, levando eventualmente a liberação de código que não tenha sido devidamente testado, com muitos bugs ainda existentes. Os próprios usuários é que acabam sendo os “testadores” e comunicando aos fornecedores as ocorrências dos bugs.

Com opção de open source, com custos reduzidos de aquisição (ou grátis, caso seja um simples download) uma parcela significativa de usuários não teria interesse em adquirir software proprietário similar. Assim, a perda da indústria seria apenas do valor das licenças que a parcela restante de usuários, ainda interessadas nos softwares proprietários, optasse intencionalmente pela pirataria.

Com um crescente uso de open source no mercado de consumo, o custo total da oficialização das licenças seria bem menor. A conta é simples. Imaginando que o índice de pirataria no Brasil seja de 50%, um esforço legal para combater a pirataria obrigaria a outra metade dos usuários de softwares de escritórios a adquirirem licenças oficiais, afetando a balança de pagamentos. Se o open source fosse adotado como programa substituto no mercado de consumo, a balança de pagamentos não seria tão afetada.

O modelo de open source pode ser uma alternativa bastante interessante no tocante aos acordos comerciais internacionais que são firmados por governos para combate a pirataria de software. Creio que vale a pena pensar nisso...

Open Source e The LongTail (Abril)

Estava relendo algumas partes do livro “A Cauda Longa” de Chris Anderson e comecei a pensar na relação deste conceito com a indústria de software. A Cauda Longa (The Long Tail) está impulsionando grandes transformações em vários mercados, como o de mídia e o fonográfico. Juntando este conceito ao de Open Source, veremos grandes transformações também na indústria de software. Queria debater com vocês alguma destas idéias...

O conceito de Cauda Longa propõe que determinados negócios podem obter uma parcela significativa de sua receita pela venda cumulativa de grande numero de itens, cada um dos quais vendidos em pequenas quantidades. Isto é possível porque a Internet abre oportunidades de acesso que antes não existiam. É um modelo diferente do mercado de massa, onde poucos artigos são vendidos em quantidades muito grandes. Na indústria de livros, música e de mídia faz todo o sentido. Por exemplo, a Amazon reporta que 57% de sua receita vem de produtos da Cauda Longa que não estão disponíveis (e jamais estariam) nas livrarias tradicionais, limitadas pelos caros espaços físicos das lojas.

E como Open Source e o conceito de Cauda Longa afetam a indústria de software? No Open Source o custo de desenvolvimento de um software é rateado por uma comunidade de desenvolvedores. Não existe um centro de custo único...Portanto o custo individual é muito pequeno.

Assim, softwares que tinham seu projeto cerceado pelo pequeno tamanho do seu mercado potencial (seu custo de produção não gerava retorno financeiro suficiente) podem agora, se desenvolvidos em Open Source, entrar no mercado. Os custos de comercialização destes softwares também tendem a zero, pois não é necessário hordas de vendedores, mas simples downloads e marketing viral (blogs e outros meios de disseminação de informação). O que isto significa? Softwares economicamente inviáveis pelo modelo tradicional, podem agora ser produzidos, baseados em Open Source, mas é claro, desde que consigam gerar também um modelo de negócios.

Temos então campo para explorar o mercado da Cauda Longa no software. E já temos exemplos bem sucedidos disso! Alguns exemplos da aplicação do conceito de Cauda Longa em software são o Eclipse, com seu mecanismo de plug-ins e o Linux. Os plug-ins do Eclipse podem atender a um mercado bem menor, que não seria alcançado caso fosse necessário um caro e sofisticado software monolítico, com 30 a 40 funções de plug-ins aglutinadas. Por exemplo, se vocês visitarem a página <http://www.eclipse-plugins.info/eclipse/index.jsp> vão acabar descobrindo um plug-in que atenda às suas necessidades, mesmo que elas sejam bastante específicas. Uma consulta que fiz no início de abril deste ano me trouxe mais de 1440 plug-ins disponíveis.

A modularização do Linux também permite criar variantes para processadores de pouca disseminação no mercado. Existe algum outro sistema operacional que rode em tantos processadores como o Linux? Claramente o Linux também atende o mercado da Cauda Longa.

O modelo comercial tradicional prende a indústria no mercado de grandes volumes. O modelo de massa precisa de base significativa de clientes para se sustentar. E acaba criando excesso de funcionalidades, pois precisa aglutinar muitas funções para atender a uma gama muito grande de clientes.

Este mercado vai morrer? Na minha opinião, absolutamente não! Acredito que conviveremos em um contexto onde os modelos de massa e de cauda longa vão compartilhar os palcos... Continuaremos a ter produtos de software bem abrangentes em funcionalidades, que venderão suficientemente bem para manter um mercado próprio. Mas também teremos muitos outros produtos, baseados em Open Source, que se encaixariam muito bem no contexto da Cauda Longa.

Gerenciando projetos de Open Source: métodos e regras (1) (Abril)

Em breve teremos o FISL 8.0 em Porto Alegre. Estarei lá mais uma vez e inclusive apresentarei uma palestra abordando modelos de negócios gerados a partir do Open Source. E como Open Source é o tema do evento, nada mais natural que debater o assunto mais uma vez aqui no blog. Vou dividir com vocês a adaptação de um capítulo do meu livro “Software Livre: Potencialidades e Modelos de Negócio”, publicado em 2004/2005 pela editora Brasport (www.brasport.com.br). Como o texto é relativamente extenso, vou dividi-lo em dois pedaços, a primeira parte hoje e a segunda no próximo post.

Já sabemos que uma iniciativa de Open Source não é apenas entusiasmo e trabalho pesado nos fins de semana. A utilização profissional e empresarial do desenvolvimento de Open Source demanda um novo método de gerenciamento, com suas características e peculiaridades próprias. Não existem, na prática, metodologias que já tenham sido comprovadas em muitos projetos. As iniciativas de maior sucesso como Linux, Apache, Eclipse e outros não usam as mesmas metodologias e nem mesmas ferramentas de apoio. Por exemplo, alguns têm um único líder que coordena todo o projeto e outros mantêm uma equipe gestora, com decisão em colegiado. Entretanto, podemos analisar pontos em comum entre eles e identificar alguns aspectos que aparentemente os levam a ter sucesso.

Gerenciar um projeto de Open Source é gerenciar pessoas em uma organização virtual, formada por colaboradores voluntários espalhados pelo mundo todo, que trabalham de maneira autônoma e não tem vínculos oficiais entre si, nem com o próprio projeto. Podem sair quando quiserem, sem aviso, simplesmente deixando de colaborar. É um mundo diferente do modelo de gerência típico de uma organização hierárquica, onde as pessoas estão no mesmo local e obedecem a regras corporativas e a hierarquias bem definidas.

O grupo de colaboradores é diverso em suas competências técnicas (desenvolvedores super-seniores a estudantes), que tomam decisões por si e se engajam por motivos diversos, que vão do ideológico e messiânico ao financeiro, em busca de projeção e novas oportunidades profissionais. Não existe o contato direto nem reuniões de projeto e, portanto ambigüidades de comunicação (línguas e países diferentes, com seus costumes diferenciados) e a falta de uma estratégia bem definida dos rumos do software, além de inexistência de cronogramas e prazos criam toda uma nova dinâmica de gerenciamento de projetos. Basicamente um projeto de Open Source deve, como qualquer outro projeto, gerenciar os processos necessários a desenvolver as tarefas, bem como gerenciar as próprias tarefas.

Um grande desafio a estes projetos é a construção dos mecanismos de controle e coordenação, uma vez que os membros são autogeridos e controlam por si mesmo seus ritmos e intensidade da colaboração. Assim, um fator crucial no Open Source é criação de motivação e confiança entre os membros do projeto. A colaboração voluntária só acontece quando motivada. Conquista-se a confiança dos colaboradores quando o trabalho individual é considerado adequadamente, os conflitos de opinião são julgados e decididos com base em critérios bem definidos e livres, e todas as informações sobre o projeto estão disponíveis a todos.

A questão da disponibilização livre das informações é muito importante. Quanto mais compartilhadas forem as informações referentes ao projeto, maiores as chances dos colaboradores contribuírem mais eficientemente. Um exemplo é a inspeção de código fonte. Nenhum código deve ser restrito em sua visualização. Qualquer membro da comunidade pode ter acesso a ele e fazer comentários ou modificações. Se for aceito ou não é outra questão, mas o acesso deve ser livre. Um outro exemplo são os processos de tomada de decisão. De maneira geral o mantenedor e seus auxiliares não devem trocar informações e opiniões sobre a aceitação ou não de determinado código, de maneira restrita, mas sim sempre em discussões abertas para a comunidade. É um modelo diferente de projetos convencionais, onde muitas informações são restritas e trafegam apenas entre os gerentes responsáveis. Entretanto, algumas informações podem ser mantidas privadas, como identificação dos membros da comunidade. Salvo por decisão pessoal, os membros identificam-se por e-mails, muitas vezes apenas sob pseudônimos, não cedendo endereços ou telefones para outros contatos.

Os principais meios de comunicação entre os membros do projeto são os eletrônicos, como e-mails, wikis e listas de discussão. São métodos ágeis, mas devido à impossibilidade do contato direto, tornam-se perigosos quando mal utilizados. Em e-mail não existe feedback imediato, não existe comunicação implícita, de se olhar no rosto, ver gestos e se ouvir a entonação da frase. Um texto mal escrito ou uma opinião sucinta e direta demais pode gerar conflitos desnecessários. A situação se agrava sob pressão, quando mais e-mails são direcionados e nem sempre respondidos com a calma e clareza necessária. Os líderes do projeto (mantenedor e seus auxiliares) sofrem bastante com esta pressão, pois concentram em si um volume imenso de mensagens. Respondê-las de forma inadequada pode criar desavenças inconciliáveis entre os membros, chegando a gerar dissidências no projeto.

Os projetos de Open Source devem procurar criar políticas de uso de e-mails (netiquetas) de modo a evitar geração de conflitos desnecessários. Estas políticas geralmente são informais e aceitas pela comunidade. O site do projeto deve explicitar as regras de conduta que a comunidade deseja que os membros adotem. O site deve também definir as regras de participação e como a organização virtual está estruturada, como e quando acessar os seus líderes, como colocar mensagens, como participar de debates e assim por diante. O uso de chats também encontra adeptos, embora não permita a participação de toda a comunidade, pelo simples fato das agendas pessoais (e fusos horários) nem sempre coincidirem.

Existe também o desafio da tecnologia de apoio ao projeto. Algumas ferramentas adicionais são importantíssimas, além dos tradicionais e-mails e listas de discussão. Como o projeto é colaborativo e assíncrono, com diversos desenvolvedores trabalhando simultaneamente na mesma solução, é necessário dispor de ferramentas que permitam este tipo de controle. O gerenciamento das diversas versões (estável, em teste, e instável ou em desenvolvimento) e da versão a ser liberada para uso geral é fundamental em qualquer projeto de Open Source. Não existem regras únicas para esta tarefa e os principais projetos adotam mecanismos diferentes. É necessário definir claramente de quem é a autoridade para liberar a versão para uso e que critérios devem ser obedecidos para que a versão seja liberada publicamente.

De maneira geral, salvo exceções de projetos de grande popularidade, como o Linux,

Eclipse ou Apache, os projetos de Open Source são registrados em sites especiais que atuam como espécies de câmaras de compensação (clearinghouses), como o SourceForge (www.sourceforge.net), o FreshMeat www.freshmeat.net e outros.

Gerenciando projetos de Open Source: métodos e regras (2) (Abril)

Bem, continuando com o assunto de gerenciamento de projetos Open Source, que começamos no post anterior, vamos abordar a questão dos líderes de projetos ou mantenedores.

O líder do projeto é de fundamental importância. Tem como papéis principais dar a visão do projeto para a comunidade; assegurar-se que o projeto está adequadamente modularizado para ser trabalhado de maneira simultânea; atrair e reter colaboradores; e manter o projeto unido, sem dissidências. Os líderes de projetos de Open Source conquistam este posto ou por terem iniciado o projeto (geralmente assumem a função de mantenedor) ou por méritos próprios, reconhecidos pelos seus pares. Devem assumir uma postura de coaching, liderando os esforços da comunidade, resolvendo conflitos e tomando decisões sobre os rumos do software. Não existe a figura do chefe ou do patrão, uma vez que não existem vínculos empregatícios entre os membros da comunidade e toda colaboração é voluntária.

Reconhecer os fatores motivacionais e incentivá-los é uma das principais funções dos líderes de projetos de Open Source. Os valores para motivação são diferentes. Por exemplo, entre a maioria do pessoal técnico um fator motivacional é o reconhecimento de sua expertise pelos pares. A meritocracia é o modelo de incentivo. Em áreas de vendas, a motivação principal é geração de dinheiro, com bônus por resultados. Manter a motivação da comunidade em todo o ciclo de vida do software é questão de sobrevivência do projeto. Desenvolver software em comunidade implica em total colaboração. E colaborar não é uma ação incentivada em muitas organizações tradicionais, muito mais focadas nas competições individuais pela conquista de uma nova posição hierárquica ou bônus financeiros.

Nos projetos com muitos colaboradores, observa-se que uma grande parcela do código vem de apenas um reduzido número de desenvolvedores. Alguns projetos recebem colaboração apenas de estudantes, que desenvolvem código como parte de seus cursos de graduação. Se as colaborações não vierem também de desenvolvedores de nível sênior, a qualidade do código pode ser bastante sofrível. Os desafios gerenciais aumentam sensivelmente, pois os líderes do projeto devem ter muito mais cuidado (e trabalho) na aceitação do código.

Na prática, controlar a qualidade do software é uma tarefa pesada, pois como a comunidade é diversa em sua expertise e proficiência técnica, a variedade de estilos de codificação que chega aos líderes de projetos populares é muito grande, com muita contribuição de qualidade duvidosa. Este é um outro grande desafio do gestor do projeto: o que aceitar ou recusar das colaborações. Muitas colaborações não são aceitas, sejam por existirem outras mais eficientes ou simplesmente por não atenderem aos critérios de codificação do projeto. Os líderes devem ter cuidado na comunicação destas recusas e as decisões devem ser sempre baseadas no voto majoritário da comunidade. Decisões isoladas não fazem parte da cultura colaborativa e são adotadas por poucos projetos.

Diante do sucesso de muitos projetos de Open Source, surge o questionamento da possibilidade de aplicarmos seu processo de desenvolvimento internamente, por uma

empresa comercial, que não tenha como objetivos de negócios desenvolver software.

Software é importante para a maioria dos setores empresariais, pois são fundamentais em criar diferenciações competitivas. Processos de negócios podem ser inteiramente redesenhados, mas para serem implementados exige-se suporte de software. E velocidade para esta implementação é cada vez mais um fator crítico para o sucesso.

Analisando uma empresa comercial observamos que o conhecimento de informática está permeada pela organização. A maioria dos usuários finais tem acesso e utiliza computadores. Não sabem programar com proficiência em linguagem de baixo nível como C ou C++, mas sabem o que querem e precisam dos seus sistema de informação.

Existe então a possibilidade de engajar todos os interessados em projetos de software interno e eventualmente ampliar esta colaboração para empresas parceiras.

Uma preocupação sempre latente é se estes projetos podem implementar processos e “best practices” de engenharia de software. Existem iniciativas nesta direção que podem servir de orientação. Um exemplo é a comunidade Tigris (www.tigris.org) que se propõe a ser uma comunidade de software livre voltada a desenvolver processos e ferramentas para uso das práticas de engenharia de software em projetos de Open Source. Também já contabilizamos algumas experiências bem sucedidas de uso de processos de engenharia de software em projetos de software livre, como nos projetos Eclipse (www.eclipse.org) apoiadas por empresas fortemente estruturadas em seus processos de desenvolvimento como IBM.

O modelo de desenvolvimento de Open Source já passou por alguns testes práticos. Existem projetos de grande complexidade, com milhões de linhas de código, desenvolvidos com qualidade e rapidez. O ritmo de desenvolvimento incremental de projetos como o kernel do Linux tem se mostrado superior a de muitos projetos de softwares houses.

Existem projetos de alta qualidade, devido ao uso de peer review, com custos diluídos por toda a comunidade. Claro que o que deu certo em Linux e no Apache não necessariamente dará certo em qualquer outro projeto de software. As circunstâncias peculiares que envolvem estes projetos (timing, motivação, área de domínio, apoio da indústria e outras) não se replicam automaticamente.

Para o sucesso de uma iniciativa de Open Source dentro de uma empresa ou instituição nos atrevemos a fazer algumas recomendações:

- a) O projeto pode começar do zero ou aproveitar o código já existente em algum repositório como sourceforge e outros. Começar do zero é sempre mais complicado, pois neste caso não existe nenhuma peça de código que funcione, mesmo precariamente. É sempre melhor começar com algum código, mesmo defeituoso, uma vez que este pode ser depurado e tornar-se rapidamente um software funcional. Mas, se começar do zero, defina claramente o propósito do software, para que as colaborações sejam direcionadas no mesmo sentido.
- b) É importante que exista colaboração de codificação. Os usuários finais devem estar engajados, principalmente para apoiar as questões relativas a usabilidade. Por ser um software para uso de usuários finais, motive a participação destes no projeto. Um ponto a

ser debatido é a possibilidade do uso de linguagens de programação de alto nível. Não recomendamos usar linguagens pouco conhecidas, mesmo que sejam também livres. Sem massa crítica de colaboradores escrevendo código o software simplesmente não sai do lugar.

c) Utilize adequadamente recursos tecnológicos que permitam facilidade de comunicação e controle como mailing lists, wikis, listas de discussão, softwares de controle de versões e assim por diante. O controle de versões, dos códigos fonte e o rastreamento dos patches a serem aplicados são atividades essenciais ao sucesso do projeto.

d) Defina claramente as posições chave da comunidade. Existirão colaboradores pouco ativos e alguns muito ativos. Não esqueça que são funcionários da empresa e voluntários, e talvez não tenham muita flexibilidade para alocar tempo. É sempre bom contar com apoio da alta administração para estes projetos, inclusive para facilitar o recrutamento de colaboradores.

e) Defina claramente as regras do jogo, os processos de tomada de decisão e resolução de conflitos, e de aceitação das colaborações. As regras de participação devem ser bastante claras e estarem disponíveis a todos os interessados.

f) Defina as necessidades de documentação e procedimentos de teste. Documentação é importante para usuários finais, que não irão recorrer a códigos fonte para tirar dúvidas de como usar o software. Quanto aos testes, pode-se adotar o clássico mecanismo de peer review ou outro procedimento qualquer. Peer review pode ser adotado se houver massa crítica de usuários suficiente para garantir a qualidade das inspeções de código.

g) Definir política de distribuição de versões. Pode-se usar critérios de três níveis, com a versão estável disponível para uso público; uma versão em teste, aguardando a liberação em um próximo release; e a versão em desenvolvimento, instável e liberada apenas para desenvolvedores.

h) Não esqueça que um Open Source é um projeto que não tem fim. Sempre cabe mais uma modificação ou adição de funções.

FISL 8.0 e o amadurecimento do modelo Open Source (Abril)

Estive no FISL 8.0 em Porto Alegre. Minha percepção é que a cada ano fica mais patente o amadurecimento da comunidade de Software Livre. Nos primeiros FISL havia muito de conteúdo ideológico e pouca racionalidade. Parecia uma Guerra Santa! Hoje já vemos discussões e apresentações muito pragmáticas. Todos saem ganhando com isso.

O movimento Software Livre ou Open Source (embora ainda haja alguma discussão com relação aos dois nomes...) pode ser simbolizada pelo crescimento do Linux. Este começou sem maiores pretensões (o e-mail do Linus Torvalds em 25 de agosto de 1991, dizia: “Hello everybody out there using minix I’m doing a (free) operating system (just a hobby, won’t be big and professional like gnu) for 386(486) clones. This has been brewing since april, and is starting to get ready. I’d like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).... Hoje Open Source é uma realidade. Softwares como o Linux, Apache, Firefox, Eclipse, JBoss, PostgreSQL, MySQL, Sendmail, PHP e diversos outros já fazem parte do portfólio de software de muitas e muitas empresas.

O movimento de Open Source, que trouxe em seu bojo a inovação no processo de desenvolvimento (tem uma frase emblemática de Eric Raymond, em seu livro “The Cathedral and the Bazaar”, que diz: “ I think Linu’s cleverest and most consequential hack was not the construction of the Linux kernel itself, but rather his invention of the Linux development model” .Este novo modelo de desenvolvimento colaborativo permitiu criar diversos novos e inovadores modelos de negócio que está afetando e vai afetar mais ainda a indústria de software.

Claro que ainda existem críticos que acham que Open Source não é coisa séria. Mas a cada dia estes críticos estão em número menor. O movimento está cada vez mais maduro e alguns mitos dos seus tempos pioneiros e românticos começam a ser eliminados. Quando começamos a debater Open Source com mais intensidade, isto lá pelos idos de 2000/2001, havia quase um consenso que os desenvolvedores que atuavam nos projetos Open Source eram 100% voluntários, o processo de desenvolvimento era anárquico, sem um road map claro e nenhuma preocupação com datas para entrega. Havia também a percepção que os fundamentos econômicos do Open Source eram intangíveis, a chamada gift economy.

Hoje se sabe que nos projetos de maior sucesso cerca de 90% dos desenvolvedores mais ativos estão na folha de pagamento da própria indústria de software, existe uma organização clara, com road map definido (os exemplos da Linux Foundation, Eclipse Foundation e Apache Software Foundation são prova disso) e que existem sim, modelos de negocio que podem ser construídos em cima do contexto Open Source.

A sinergia entre a indústria e a comunidade é positiva para todo mundo. Uma pesquisa, ainda em draft, em elaboração por pesquisadores da Universidade de Pisa, na Itália, tem gerado dados muito interessantes. Eles pesquisaram 300 projetos mais ativos da SourceForge.net (este repositório tem hoje mais de 140.000 projetos Open Source) e avaliaram o dinamismo dos projetos quando estes estavam sendo apoiados intensamente

por empresas de software (ou foram criados por estas empresas), junto com comunidade, versus os projetos que estavam sendo desenvolvidos exclusivamente pela comunidade.

A pesquisa até o momento está demonstrando que a sinergia indústria-comunidade é altamente positiva. Os projetos que tem apoio da indústria tem em média 19 desenvolvedores ativos, contra cinco dos demais projetos. Também apresentam maior atividade (numero de mailing lists, reportes de bugs, ritmo de geração de código, etc) e demonstram um viés mais focado na tecnologia Java que na família C (C e C++). Os projetos apoiados pela indústria são desenvolvidos principalmente em Java , 47%, contra 40% de projetos em C/C++. Já os desenvolvidos apenas pela comunidade apresentam C/C++ como principal opção de linguagem, com 65%.

A pesquisa também mostra um maior pragmatismo quanto à escolha das opções de licenciamento. Nos projetos em que empresas de software participam ativamente a licença GPL foi a escolhida em 45% dos casos, contra quase 75% quando o projeto é conduzido apenas pela comunidade. As outras licenças mais usadas são basicamente a BSD, Mozilla Public Licence e a Apache Licence.

A conclusão que podemos chegar é que está ocorrendo um maior amadurecimento da comunidade, com conseqüente maior evolução e aperfeiçoamento do modelo Open Source. Na minha opinião estamos no caminho certo.

Mas o FISL também teve seu momento de preocupação. Cedi parte do meu tempo para o Jomar, da ODF Alliance Brasil falar um pouco do ODF (Open Document Format) e do papel da ODF Alliance. Para minha surpresa, quando ele perguntou para a platéia (umas 200 pessoas) quem conhecia ODF, apenas um parcela muito pequena levantou a mão. Temos claramente um problema. A questão do formato aberto de documentos é crucial para a sociedade e infelizmente ainda não vemos que lhe está sendo dada a devida importância. Precisamos ficar atentos para esta questão, antes que ela gere um problema insolúvel de aprisionamento forçado. Aliás, já está gerando.

Linux Park no Rio de Janeiro (Abril)

Particpei do Linux Park, no Rio de Janeiro. O evento teve um bom público, mais de 100 pessoas e um conteúdo muito bom. O Linux Park é organizado pela Linux New Media, que edita a Linux Magazine e a EasyLinux. O evento começou com uma apresentação de alguns dados abordando o Linux no mundo inteiro.

Várias pesquisas foram mostradas. Uma delas foi uma pesquisa de 2005, da Chadwik Martin Bailey com 500 executivos de TI. Ao serem perguntados quais os motivos que os levaram a adotar Linux, a resposta foi confiabilidade e segurança, seguido em terceiro por redução de custos. Além disso, citaram como pontos fortes a possibilidade de maior concorrência (sempre salutar para o mercado) e inovação mais rápida. Uma outra pesquisa, esta do Robert Frances Group (de agosto de 2005) mostrou que o TCO do Linux é cerca de 40% do TCO do Windows e 14% do TCO do Solaris.

A seguir foi apresentada uma pesquisa que a própria Linux Magazine e a Intel fizeram ano passado no Brasil, com 84 empresas de pequeno a grande porte. No universo pesquisado apareceu uma distribuição bastante homogênea das distribuições Linux, com RedHat/Fedora com cerca de 16%, Mandriva com cerca de 15%, SuSe/Novell e Slackware 14%, e Ubuntu, com aproximadamente 12%. Surpreendentemente o Linux também apareceu bem cotado em desktops, quando 48% das empresas pesquisadas disseram que já usam Linux em parte do seu parque de máquinas. E 90% das empresas pretendem aumentar o uso de Linux e Open Source no futuro. Bom sinal!

A seguir tivemos palestras bem interessantes, abordando casos reais de uso do Linux em empresas como Ponto Frio, Hoplon, Serpro e Petrobras. O Ponto Frio é um caso interessante, pois é a segunda maior rede de eletrodomésticos do país, com um faturamento de 3,8 bilhões de reais em 2006. Estão em processo de rollout do Linux nos servidores e nas estações de trabalho de vendas das lojas. São 370 pontos de venda. Usam Linux também no mainframe.

A Hoplon é a desenvolvedora de um game MMOG (massively multiplayer online game) que roda em um mainframe com Linux. As características de um game deste tipo são bem diferenciadas das aplicações comerciais típicas. Por exemplo, não tem limites de usuários (não se tem controle sobre o número de pessoas que entram no jogo) e com isso podem ter milhares de usuários simultâneos, demandando respostas em milésimos de segundos.

Posteriormente tivemos palestras do Serpro, contando sua experiência da Rede Local de Software Livre e da Petrobras, com o exemplo de clusters Linux, para computação de alto desempenho. Hoje a Petrobras, nas análises sísmicas utiliza cluster beowulf com mais de 9.000 CPU's, 100% Linux.

Que conclusão chegamos? Linux é uma realidade, qualquer que seja a aplicação.

Para terminar o evento, tivemos um painel com participação da IBM, Itaotec, Microsoft, Novell e Red Hat. Na primeira rodada, fez-se a mesma pergunta a todos os participantes:

“Do ponto de vista de negócios, qual é a importância de iniciativas de interoperabilidade e o uso de padrões abertos? O que a empresa que você representa está fazendo neste sentido?”. Bem, a minha resposta, de forma resumida, foi mais ou menos a seguinte...

Interoperabilidade pode ser percebida de várias maneiras. Uma delas é o grau de flexibilidade de uma organização de TI para responder às mudanças no cenário de negócios. Por exemplo, escalabilidade multiplataformas. Como nenhuma tecnologia resolve todos os problemas da melhor maneira, temos que adotar soluções tecnológicas diferentes. As palestras anteriores mostraram isso. Para termos interoperabilidade é necessário termos padrões abertos, que são padrões disponíveis a todos, sem pagamentos de royalties, flexíveis (extensíveis) e que devem ser gerenciados e publicados por uma entidade aberta e neutra, para garantir competição justa, evitar discriminação (favorecer ou prejudicar determinado fornecedor) e impedir práticas predatórias, quando um fornecedor tentar absorver para si o padrão. Exemplos de padrões abertos são o GSM para celulares, TCP e HTML para a Internet, e agora o ODF.

Para a IBM, interoperabilidade e padrões abertos está no cerne do negócio. Em software, estamos concentrados em middleware. Em um mundo cada vez mais interoperável um middleware fechado não tem vez. Assim, nosso middleware adota padrões abertos. O ambiente aberto e interoperável é que permite a adoção da arquitetura orientada a serviços (SOA), que entendo será o paradigma arquitetônico dos próximos anos, assim como o cliente-servidor o foi na década passada.

Open Source e o Mundo é Plano (Maio)

Não sei se vocês leram “The World is Flat: a Brief History of the Twenty-First Century”, que saiu no Brasil como “O Mundo é Plano: uma breve história do século XXI”, de Thomas L. Friedman. Se não leram, devem ler. Imperdível.

Mas queria colocar em debate alguns pontos que ele aborda no capítulo dois, onde descreve, na sua concepção, as dez forças que achataram o mundo. Uma delas é a que ele chama de Força numero 4, Código Aberto. Sim, ele aborda Open Source!

Neste capítulo, o autor descreve o contato que teve com algumas comunidades de Open Source, notadamente o pessoal envolvido com o Apache. Como todo bom jornalista, Friedman olha os dois lados da questão e entrevista executivos da Microsoft, e como ele mesmo diz “De todas as questões que já tratei neste livro, nenhuma desperta mais paixões por parte de defensores e opositores que o código aberto. Depois de passar algum tempo com a comunidade do código aberto, quis ouvir o que a Microsoft tinha a dizer, visto que este será um debate crucial, que vai determinar a força niveladora que terá o código aberto”.

Ele cita alguns questionamentos de executivos da Microsoft: “como estimular a inovação se todo mundo está trabalhando de graça e dando seu trabalho de presente?”. Cita também uma frase de Bill Gates, “Para haver inovação, precisa haver capitalismo. Um movimento que prega que a inovação não merece uma recompensa econômica vai de encontro ao futuro do mundo. Quando falo com chineses, eles me dizem que seu sonho é abrir uma empresa. Nenhum deles pensa em ser barbeiro de dia e escrever software livre à noite”.

Friedman conclui afirmando que acredita que o Open Source tem uma contribuição significativa a dar para o nivelamento do mundo, mas coloca em dúvida as questões financeiras: como poderá existir lucro com Open Source?

Aí entra meu questionamento. Na minha opinião ele está concentrado no software em si e não está vendo o multibilionário ecossistema que envolve o Open Source. Olha a árvore e não vê a floresta em torno...O modelo Open Source implica em novos modelos mentais e novas maneiras de conceituar criação de valor.

Tem uma frase de Linus Torvalds muito interessante que vale a pena citar aqui, quando ele afirma que Linux é como uma utility, que provê infra-estrutura básica para construção de novos aplicativos e negócios: “...proprietary source code at the infrastructure level can actually make it much harder for other players to enter the market. So if anything, open source is what makes capitalism in software possible at all. Without open source, you’d have just a set of monopolies: effectively, economic feudalism”.

Na minha opinião, inovação colaborativa não diminui as oportunidades de criar valor, pelo contrário, as aumenta.

Vamos exemplificar com algumas observações:

A velocidade e complexidade das mudanças tecnológicas já não permite que nenhuma empresa faça tudo sozinha. Cito uma frase muito legal de Irving Wladawsky-Berger, VP da IBM, que fala o seguinte sobre Open Source: “Essa etapa que ora se inicia caracteriza-se pela inovação de caráter colaborativo, empreendida por muita gente envolvida em comunidades talentosas, do mesmo modo como a inovação na era industrial caracterizou-se pelo gênio individual”. Penso que estamos vivenciando um novo paradigma. Temos que entender este novo mundo...Mas não o olhando pelos óculos antigos!

Uma vez que determinada camada de software caminha para comoditização, como sistema operacional para plataformas Intel, porque não se concentrar em camadas de maior valor agregado? Vejam o que o Financial Times disse em junho de 2004: “se quiserem diferenciar, os fabricantes de software comerciais têm de começar a atuar nos níveis mais altos dos softwares. Da infra-estrutura, quem está cuidando é primordialmente a comunidade de Open Source”. Não adianta remar contra a maré.

Com Open Source não existe mais espaço para lucratividades fantásticas como havia antes e empresas de software, acostumadas a venderem milhões de cópias sem concorrência, devem se reposicionar para viver neste novo mundo. Já existem alternativas: Linux, OpenOffice, Firefox... Capitalismo não é concorrência? É uma mudança inevitável, pois o Mundo é Plano!

Linux em Tempo Real (Maio)

Nos últimos anos temos visto uma crescente utilização de softwares embarcados em praticamente todos os objetos construídos pelo homem. Software embarcado é o software que é embutido dentro de um equipamento, como um sistema de injeção eletrônica de um automóvel, permitindo que este equipamento atue com maior funcionalidade e flexibilidade. Antes apenas utilizados em sistemas complexos como sistemas industriais, aeronaves e navios, hoje vemos softwares embarcados em geladeiras, televisores e fornos de microondas. Estes equipamentos tornam-se cada vez mais sofisticados, demandando mais e mais complexidade no seu hardware e software embarcado.

Muitas indústrias que tradicionalmente competiam pela qualidade do seu produto físico, como automóveis e eletrodomésticos, estão começando, de maneira crescente e rápida, a adotar software em seus produtos. Em alguns setores, como o automotivo as estimativas apontam que mais de 70% das futuras inovações serão baseadas em tecnologia de software e não mais nas partes mecânicas. Estima-se que por volta de 2010 os modelos de automóveis topo de linha deverão incorporar mais de 100 milhões de linhas de código de software em cada veículo. O uso do software embarcado na indústria não é mais uma questão restrita aos setores de Pesquisa e Desenvolvimento (P&D), mas parte integrante e essencial das estratégias de diferenciação competitiva dos seus produtos.

Alguns números já demonstram de forma inequívoca a importância do software embarcado no setor industrial. Quando falamos em diferenciação competitiva, já observamos que parcela significativa da diferenciação entre os produtos baseia-se na maior oferta de funcionalidade, suportada por tecnologia de software. A revolução digital tem mudado e vai continuar mudando a dinâmica de muitas indústrias. Na indústria de eletroeletrônicos vemos claramente a digitalização substituindo o mundo analógico.

As transformações não ocorrem apenas em máquinas sofisticadas, como automóveis de luxo, mas já vemos lavadoras de roupa com software embarcado possibilitando uma maior oferta de funções. As vantagens dos objetos falarem uns com os outros e com os computadores que processam as aplicações nas empresas são imensas. Para um fabricante de geladeiras, saber com antecedência de eventuais problemas de manutenção identificados por sensores e transmitidos via Internet aceleram as atividades da assistência técnica e transformam as relações com os seus clientes.

Neste mercado o Linux tem papel de extrema importância, principalmente pelo custo de entrada baixo. Como o Linux não demanda pagamento de royalties, o usuário não vai pagar um valor adicional pelo custo da funcionalidade adicionada a uma geladeira...O resultado é que o Linux está caminhando para se tornar o sistema dominante no ambiente de sistemas embarcados. Recentes pesquisas apontam que os desenvolvedores de sistemas embarcados estão migrando ou planejando a migração para Linux, migrando dos sistemas proprietários que anteriormente dominavam este mercado. Já existem muitos negócios gravitando neste mundo, inclusive distribuidoras especializadas em Linux embarcado.

A liderança do Linux nos sistemas embarcados tem um significado importante, pois à

medida que se dissemine em milhões de dispositivos (em número muito maior que os PCs), acarretará uma forte influencia nas plataformas tradicionais. Como software é software, qualquer que seja a plataforma, desenvolvedores acostumados a desenvolver aplicações para o ambiente Linux em um celular, podem rapidamente adaptar seus conhecimentos para desenvolver também aplicações para qualquer outra plataforma.

Porque o Linux tem feito tanto sucesso neste cenário? O mercado de software embarcado tem peculiaridades específicas. Apresenta uma ampla diversidade de funcionalidades e utiliza uma gama muito grande de processadores. É uma diversidade diferente do ambiente de computação tradicional, onde existe uma concentração em poucos processadores.

O software embarcado deve apresentar alta estabilidade. Uma aeronave ou uma usina nuclear não pode apresentar falhas no software. Outra característica de muitos dispositivos é a necessidade de operação em tempo real, principalmente nos equipamentos de controle de processo.

O software embarcado deve operar em ambientes de recursos computacionais limitados, como memória ou discos magnéticos. Assim, recursos como gerenciamento de discos, rotinas de grande impacto no desempenho de sistemas comerciais, torna-se pouca importância no contexto da computação embarcada. O fator custo é de grande importância em sistemas embarcados. Um telefone celular não pode custar o dobro de outro por causa do preço do software básico que esteja embutido nele. O sistema operacional de um equipamento destes é totalmente invisível e não desperta a percepção do comprador. As diferenças devem ser claramente perceptíveis, como um maior número de funcionalidades disponibilizadas. Um sistema operacional baseado nos princípios do Open Source tira um peso grande do custo, tornando-se bem atraente para este setor.

O Linux, pela qualidade de seu código já é hoje peça fundamental da arquitetura de tecnologia de sistemas embarcados militares dos EUA. A Marinha americana baseou sua arquitetura, denominada NOACE - Navy Open Architecture Computing Environment em padrões abertos e considera o Linux fundamental em sua estratégia de reduzir os tempos e custos de desenvolvimento de seus sistemas embarcados em navios, aeronaves e submarinos. O primeiro grande projeto da Marinha americana a adotar esta tecnologia é a nova classe de destroyers DD(X), cujo primeiro navio é o DDG 1000, Zumwalt. O software destes navios vai rodar em servidores Blade da IBM, que também contribuiu decisivamente para adaptar os ambientes Linux e Java (WebSphere Real Time) para operar em tempo real. O mesmo conceito de padrões abertos foi adotado pelo Exército americano em sua arquitetura chamada de WSCOE – Weapons Systems Common Operating Environment.

De maneira geral, os usuários (consumidores) destes equipamentos não sabem e nem precisam saber que o Linux está rodando em seus dispositivos. Mas para os fabricantes o fato do Linux estar disponível em código fonte, permite que seja usado sem pagamento de royalties para cada produto vendido, além de facilitar em muito a customização, praticamente montando-o para as necessidades específicas de cada aparelho. Isto dá aos fabricantes um completo controle sobre o dispositivo, permitindo fazer modificações e implementar inovações muito rapidamente.

Wide Open: Open Source indo além do software (Maio)

Recentemente li um paper muito interessante, “Wide Open: Open Source Methods and their Future Potential”, que pode ser acessado em <http://ploug.eu.org/doc/WideOpen.pdf>. Os seus autores mostram que o sucesso dos métodos Open Source (transparência, peer review, etc), comprovados por projetos de sucesso como Linux, Apache, Eclipse, Wikipedia e outros, podem ser adotados em diversos setores das atividades humanas e não apenas em desenvolvimento de software.

O Wikipedia, por exemplo, tem sido a prova de conceito que os métodos Open Source podem ser aplicados fora do âmbito do desenvolvimento de software. Desde sua criação em 2001, já acumulou mais de 5 milhões de artigos, é um dos sites mais acessados da Internet, chegando muitas vezes a mais de 16.000 acessos por segundo. É um fenômeno que retrata a colaboração por excelência e por isso tem sido estudado com mais intensidade. O criador do Wikipedia, Jimmy Wales é bem contundente. Recentemente declarou: “Encyclopedia Britannica will be crushed out of existence within five years. How can they compete? Our cost model is just better than theirs”. Difícil de contra-argumentar, não?

Mas, voltando a idéia de se usar os métodos Open Source além do software, já existem alguns exemplos pioneiros que mostram que esta idéia talvez não seja tão alucinada assim. Aliás, tem uma frase de Albert Einstein muito legal que tem a ver com tudo isso: Se à primeira vista uma idéia não parecer absurda, não há esperanças para ela”.

Mídia e entretenimento são segmentos que podem (aliás já estão) sendo transformados por estes métodos. Na mídia em exemplo bem sucedido de um “open newspaper” é o jornal eletrônico OhmyNews (<http://english.ohmynews.com/>). Este jornal foi criado em 2000, na Coreia do Sul e tem como lema a frase “Every Citizen is a Reporter”. Apenas uns 20% do conteúdo do site é escrito pelo seu staff. A imensa maioria das reportagens é enviada por contribuidores voluntários, que geralmente não são jornalistas profissionais. O jornal ganhou projeção pela influência que teve nas eleições presidenciais da Coreia, em 2002. Este novo tipo de mídia pode ter espaço como jornal de bairro, de determinada classe profissional ou mesmo de interesse comum a uma comunidade. Um exemplo deste tipo de jornal é o UKVillages (<http://www.ukvillages.co.uk/>).

Uma questão interessante é: será que amadores podem fazer o mesmo trabalho de profissionais? Muitas vezes a resposta será sim...É bem provável que vejamos muitas reações contrárias, uma vez que bastiões comecem a ser derrubados. Mas, as mudanças estão acontecendo, queiramos ou não.

Um outro exemplo de aplicação dos métodos Open Source pode ser nos debates públicos sobre determinadas leis. Um exemplo seriam wikis onde leis poderiam ser debatidas e refinadas pela própria população, antes de serem submetidas à discussões, votação e aprovação pelo legislativo. Pode-se pensar em livros didáticos sendo escritos por colaboradores. O próprio Wikipedia tem diversas iniciativas neste sentido, como a criação de livros (http://en.wikibooks.org/wiki/Main_Page) e um dicionário (http://en.wiktionary.org/wiki/Wiktionary:Main_Page).

E existe uma iniciativa muito interessante, chamada MySociety (<http://www.mysociety.org>) que é um centro de desenvolvimento de sites para entidades sociais e culturais, que não visam lucro. Os webdesigners e desenvolvedores contribuem voluntariamente para a criação destes websites.

Existem inúmeras outras possibilidades: biociências, pesquisas médicas, etc. Recomendo a leitura do documento, pensar no assunto e chegar às suas próprias conclusões. Métodos Open Source não são mais questões acadêmicas, mas estão chegando às portas de vários segmentos. Quais serão afetados? Como? Que transformações (riscos e oportunidades) nos modelos de negócio, legislações e impactos sociais resultarão disso?

IBM e Open Source (Maio)

Já faz muitos anos que estou intensamente envolvido com Linux e Open Source na IBM e acho que seria interessante compartilhar um pouco desta experiência. Gostaria de comentar minha visão pessoal do que é e o que será Open Source para IBM.

Há anos atrás quando a IBM começou a falar deste assunto, para muita gente a surpresa era grande...A IBM envolvida com Linux e Open Source? Porque uma empresa que durante muitos anos foi considerada um bastião dos sistemas fechados (a era dos mainframes) entraria neste negócio, e ainda quando o movimento de Open Source era considerado um território inexplorado?

Bem o envolvimento, ou melhor, o comprometimento da IBM com Linux e Open Source começou há uns dez anos. Uma das primeiras experiências da IBM com Open Source foi a liberação do código fonte de um compilador Java, o Jikes, ainda em 1997. Se vocês quiserem saber um pouco mais dele é só visitar <http://jikes.sourceforge.net>. Posteriormente a IBM se envolveu com o Apache, anunciando que ele seria colocado na oferta WebSphere, isto ainda em 1998.

A partir destas experiências iniciais, foram desenvolvidos diversos estudos internos para entender melhor o que era este movimento e seus impactos na indústria de software e na empresa. Em 1999 foi apresentado um relatório aos executivos do Corporate Technology Council mostrando que a IBM deveria se engajar no Open Source.

E em 2001, quando a IBM anunciou no Linux World um investimento de um bilhão de dólares em iniciativas em torno do Linux, o mercado compreendeu que a coisa era realmente séria. Um bilhão de dólares não é envolvimento, mas sim comprometimento!

E de lá para cá muitas iniciativas importantes apareceram: o Eclipse, Cloudscape, Gluecode, etc. Fizemos a doação de 500 patentes de software para a comunidade Open Source, com o objetivo de fomentar inovação, como diz o anúncio oficial da doação: “it is hoped that other patent holders will join IBM in establishing a patent commons for the benefit of OSS and to encourage innovation”. Criamos o LTC (Linux Technology Center), uma rede de mais de 600 desenvolvedores (inclusive no Brasil) que desenvolvem código, que é doado para a comunidade Linux.

Mas porque este comprometimento? Bem, na minha opinião Open Source é uma inovação no processo de desenvolvimento de software, que permite construir novos e inovadores modelos de negócios. Tem uma frase muito interessante de Eric Raymond, em seu livro “The Cathedral and the Bazaar” que expressa bem esta idéia: “I think Linus’s cleverest and most consequential hack was not the construction of the Linux kernel itself, but rather his invention of the Linux development model”.

O desenvolvimento colaborativo permite a construção de softwares fantásticos e abre novos mercados, antes inatingíveis. Com Open Source a indústria de software pode entrar no mercado da Cauda Longa (Long Tail), inacessível antes devido aos custos de produção

distribuição e comercialização de software pelos modelos tradicionais. Por exemplo, com o Gluecode (base do WebSphere Application Server Community Edition) conseguimos entrar em um mercado inviável anteriormente.

Bem, e o que estes anos de estrada nos ensinaram? No início havia uma preocupação com a qualidade do software desenvolvido de forma aberta e colaborativa. Hoje, as experiências práticas tem nos mostrado que estes softwares podem ser e muitas vezes são melhores que muitos softwares proprietários. Ganhamos uma melhor compreensão do modelo Open Source. As pessoas participam da comunidade como indivíduos não como empresa. Você não é empregado da IBM, mas uma pessoa que está colaborando com determinada comunidade. Os próprios processos de comunicação são muito mais informais e ágeis nas comunidades que nas empresas, e para isso criamos até o LTC, para criar uma comunidade mais ágil de desenvolvedores envolvidos com Linux.

Um aprendizado importante foi como alavancar projetos Open Source. Não adianta nada abrir código fonte se não existir uma comunidade interessada e ativa. E os receios que o mundo seria 100% Open Source são infundados. Tenho convicção que existe espaço para balancear modelos de negócios baseados tanto em softwares proprietários como em abertos.

Portanto, os objetivos da IBM com Open Source são: aumentar o ritmo e velocidade das inovações, incentivando o “caldo cultural” de inovação nas comunidades (inteligência coletiva), ser um contribuidor ativo das comunidades e não apenas consumidor, capturar e transformar inovações Open Source em valor para nossos clientes (como exemplos o uso do Linux nos hardwares IBM e a tecnologia Rational fundamentada no Eclipse) e alavancar modelos de negócios baseados em Open Source para entrar em novos mercados e expandir oportunidades de negócio. É, portanto uma estratégia e não um simples oportunismo para explorar uma onda , ou uma reação forçada pela pressão do mercado.

E o futuro? Que esperamos do Open Source? Bem, a estratégia da IBM com Open Source fica bem clara quando lemos um comentário de Sam Palmisano, gerente geral da IBM mundial, quando ele diz “Open Source is a method of tapping a community of experts to develop useful things. It began in software, but applies broadly, and is anything but anti-capitalist. It can raise quality at reduced costs, and vastly expands opportunities for profit. In a sense, open source fuels innovation much the way science fuels technology. Science is created by communities of experts, whose fundamental discoveries are typically made available to all, including individuals and companies that are able to capitalize on the new knowledge in novel ways. For IBM, the open source model is familiar territory, given our long track record in the sciences”. Isso aí!

Degustando alguns números sobre Linux e Open Source (Maio)

Há poucas semanas recebemos os resultados de uma ampla pesquisa, efetuada nos EUA, com uma representativa comunidade de desenvolvedores que desenvolvem aplicações usando tecnologias Open Source. Grande parte do trabalho é claro, será usada internamente, nas nossas estratégias de negócio, não podendo ser divulgada. Mas alguns dados interessantes podem ser compartilhados. Vamos a eles.

Este pessoal já usa Linux intensamente e tem condições de opinar com conhecimento de causa. E o avaliam muito positivamente. Por exemplo, Linux é o ambiente favorito quando se fala em desenvolvimento para Web (cerca de 63% o recomendam), contra 29% do Windows (qualquer versão) e menos de 6% de variantes Unix. Para aplicações mission-critical o Linux também aparece na frente (53%), tendo em segundo lugar variantes Unix, com aproximadamente 25%. Linux também lidera em sistemas embarcados (62%) e computação de alto desempenho (57%).

O Linux também é bem avaliado no quesito segurança, aparecendo como sendo considerado bem mais seguro que as diversas versões Windows e pelo menos tão seguro quanto conhecidas variantes Unix, como Solaris e AIX.

A pesquisa também mostra um predomínio do MySQL como banco de dados mais usado por estes desenvolvedores, bem como a liderança incontestada da linguagem Java. Aliás, falando em linguagens, depois da Java, aparecem na preferência destes desenvolvedores as linguagens dinâmicas (scripting languages) como PHP e Perl. Depois é que vemos C++. Python também aparece bem posicionado e interessante é vermos que a linguagem Ruby, bastante badalada, ainda aparece bem atrás destas todas...Enfim, é apenas uma pequena degustação, mas pode nos dar uma boa leitura da situação. Linux e Open Source já estão consolidados e seu impacto na indústria de software tem sido e será bastante significativo. Para muitas empresas este fenômeno tem sido extremamente positivo, abrindo muitas e até inesperadas oportunidades. Para outras a reação foi totalmente negativa.

E como as empresas são compostas de pessoas, suas formas de reação, muitas vezes emocionais, quando confrontadas com novas ameaças aos seus modelos de negócios, me lembram de um livro que havia lido há muitos anos atrás, chamado de “On Death and Dying”, da médica Elisabeth Kübler-Ross.

Adaptando as reações das empresas frente a quase catástrofes às das pessoas quando frente à perspectivas da morte observamos que as reações são muito similares. Os estágios das pessoas frente à perspectiva da morte são, primeiro o choque e a negação, depois vem a raiva, a barganha, depressão e finalmente a aceitação.

Vamos olhar a reação de algumas empresas frente ao movimento do Linux e Open Source. Primeiro foi o choque e a negação. Vimos no início declarações de executivos destas empresas dizendo que o Linux e o Open Source não eram movimentos consistentes e sólidos e que em breve seria apenas mais uma moda que ficaria para trás. A reação foi de negar a existência do movimento de Open Source ou pelo menos ignorá-lo. Exatamente

como as pessoas reagem: não acreditam no diagnóstico e se negam a acreditar que exista algo errado com elas.

Depois vem a raiva. As pessoas se mostram inconformadas e se perguntam “por que comigo?”. No mundo corporativo a reação é similar. Vimos estas empresas reagindo de forma emocional, atacando o movimento Linux e Open Source com intensidade. É o momento de empregar a técnica do FUD (Fear, Uncertainty and Doubt – medo, incerteza e dúvida), inclusive criando sites e campanhas de marketing que tentam mostrar as desvantagens do novo inimigo. Vem o momento da barganha. Acreditamos que promessas podem trazer a cura. “Se eu ficar curado, juro que...”. Na nossa analogia surgiram as iniciativas como lançamento de versões proprietárias mais baratas, que em princípio poderiam ser alternativas às opções de Open Source.

A depressão vem a seguir. As pessoas perdem interesse pela vida. Claro que isto não aconteceu na indústria de software, mas lemos na mídia extratos (provavelmente não autorizados...) de emails e atas de reuniões onde o fenômeno Open Source já era visto como algo que chegou para ficar. E o Linux e Open Source já aparecem nos relatórios para acionistas, nos itens relacionados com riscos para o negócio.

A última etapa é a aceitação. Se não posso lutar contra, melhor conviver da melhor forma. Começamos a ver estes sinais com algumas iniciativas apoiando projetos de Open Source, participação em parcerias e eventos focados em Linux e assim por diante.

Kubler-Ross em seu livro, desafiou uma cultura determinada a varrer a morte para debaixo do tapete e escondê-la ali. O Linux e o Open Source desafiaram o tradicional modelo da indústria de software e mostraram que não devemos temê-lo, mas ao contrário, nos abriram novas perspectivas para criação e uso de software. Não é mais uma questão de “se vou usar Open Source ” mas “quando e como vou usar”.

Novamente Linux em Tempo Real (Junho)

Há poucas semanas levantei um post abordando Linux embarcado. Interessante que foram colocados muitos comentários e recebi diversos emails sobre o assunto. O tema está realmente despertando interesse. Assim, acho que vale a pena extrair alguns parágrafos do meu livro “Software Embarcado: a Nova Onda da Informática”, editado pela Brasport e compartilhá-los com vocês.

O mercado de software embarcado tem peculiaridades específicas. Apresenta uma ampla diversidade de funcionalidades e utilizam uma gama muito grande de processadores. Os softwares embarcados apresentam uma variedade imensa de arquiteturas. É uma diversidade diferente do ambiente de computação tradicional, como desktops, onde existe uma concentração em poucos processadores, como Intel e AMD.

Uma característica do software embarcado é que deve apresentar alta estabilidade. Uma aeronave ou uma usina nuclear não pode apresentar falhas no software. Claro que existem dispositivos menos exigentes quanto à falhas, como por exemplo, uma máquina de venda de refrigerantes (vending machine), onde uma eventual falha não causa maiores danos ou riscos de vida. Outra característica de muitos dispositivos é a necessidade de operação em tempo real, principalmente nos equipamentos de controle de processo.

Algumas falhas de software embarcado podem ser catastróficas. Por exemplo, na guerra das Malvinas, a fragata inglesa Sheffield foi afundada porque o software de seu radar identificou um míssil argentino como “amigo” e não acionou as defesas antimísseis. Na primeira Guerra do Golfo, em 1991, uma pequena falha de software, com erros de precisão de 0,000000095 segundos em cada décimo de segundo, gerou uma imprecisão acumulada em 100 horas que fez com um míssil Patriot não conseguisse interceptar um míssil iraquiano Scud, que explodiu no alvo, matando 28 pessoas.

As interfaces de acesso também são variadas, indo de simples teclados e mouses, a sensores e atuadores especializados.

A maioria dos dispositivos dispõe de pouco espaço, e, portanto o software deve operar em ambientes de recursos computacionais limitados, como memória ou discos magnéticos. Assim, recursos como gerenciamento de discos, rotinas de grande impacto no desempenho de sistemas comerciais, torna-se pouca importância no contexto da computação embarcada. Esta heterogeneidade aparece também no volume de vendas do software. Dispositivos embarcados podem ser vendidos aos milhões, como telefones celulares ou em poucas unidades, como um sistema de automação de planta industrial. Os preços dos equipamentos também variam de poucos reais até milhões de reais.

Esta diversidade de hardware e funcionalidade leva a uma indústria altamente fragmentada. Este cenário, tradicionalmente ocupado por sistemas operacionais proprietários, começa a ser ocupado pelo Linux.

Esta tendência pode ser explicada em parte pela crescente sofisticação dos equipamentos e

dos sistemas embarcados. Quando as funções eram relativamente simples e por serem únicas, a solução mais adotada era a construção de um sistema proprietário e específico. À medida que os equipamentos se sofisticam, aumenta a complexidade do sistema operacional embarcado, e a manutenção de equipes para desenvolvimento e manutenção de sistemas proprietários de uso específico começa a ficar caro demais. Os sistemas operacionais começam a ter interfaces mais sofisticados como Ethernet, USB, Bluetooth e outros.

O Linux torna-se atrativo, pois com ele se pode partir de um sistema operacional escalável e modular, com uma base sólida e estável, que já suporta os principais interfaces e protocolos abertos. Não se precisa reinventar a roda, mas parte-se de uma porção significativa de código fonte já pronto e testado, para concentrar-se apenas na construção das especificidades necessárias a função de um determinado equipamento. A redução de tempo (time-to-market) e custos é bastante positiva. Outro impulsionador para o uso do Linux em sistemas embarcados é o próprio modelo de software livre, que não exige pagamento por licenças. Como muitos dispositivos são contados aos milhões, um pagamento de royalties tornaria o negócio inviável.

As empresas que atuam no setor de computação embarcada usam o código base do Linux, fazem modificações e criam aditivos, com funcionalidades específicas, direcionadas às suas necessidades. Além disso, como o código Linux é abrangente, faz-se também uma verdadeira lipoaspiração, cortando código desnecessário às funções embarcadas. O Linux não foi inicialmente projetado para o mercado de software embarcado e precisa, portanto, passar por estas cirurgias.

Entre as funções e modificações destacam-se as que permitem o Linux operar em tempo real; que permitem um start-up imediato (o tempo de boot de um equipamento embarcado não pode ser tão lento quanto o de um desktop); e que fazem o sistema ser extremamente eficiente, operando em um ambiente com um mínimo de recursos computacionais.

Como o kernel padrão do Linux não é um sistema em tempo real, seu tempo de resposta não é determinístico. Para operar em tempo real, existem duas alternativas, ou colocar o Linux operando em baixo de um sistema tempo real (na verdade um mini sistema operacional, apenas focado em tratar interrupções em tempo real) ou tornar o próprio kernel do Linux preemptível, ou seja, permitir que tarefas de alta prioridade obtenham controle do processador com a máxima presteza. O patch que permite o Linux ser preemptível foi adotado por Linus Torvalds no código base a partir da versão 2.5.

Adicionar um segundo kernel, de alta prioridade, significa que o kernel normal vai operar como uma tarefa de menor prioridade. Alguns puristas lembram que esta opção injeta um componente não Linux (e eventualmente proprietário) no cenário.

Embora as alternativas embarcadas não sejam compatíveis entre si, não chegam a criar variantes (forkings) do kernel do Linux. As empresas de sistemas embarcados criam suas novas versões de tempo real a partir da última versão do kernel, não fazendo atualizações em cima das versões anteriores já adaptadas, o que o distanciaria mais e mais do Linux standard. Em cima desta última versão é que atualizam e inserem as funcionalidades de

tempo real. Com isso se mantém a compatibilidade com o código base do Linux, e concentram-se em apenas uma pequena parte do kernel, exatamente a que precisa de modificações para se comportar na modalidade de tempo real.

Com esta estratégia, as empresas de sistemas embarcados conseguem usufruir do apoio e suporte da própria comunidade, no que se refere a bugs do código base, sem necessidade de retrabalhos.

Todos estes impulsionadores têm levado a uma crescente importância do Linux como sistema operacional para computação embarcada. O mercado de eletrônica de consumo é um dos segmentos mais atuantes na adoção do Linux. Uma pesquisa no site LinuxDevices (www.linuxdevices.com) retorna centenas de produtos, que variam de celulares, smartphones, telefones para VoIP, PDAs, set-top-boxes da televisão digital e diversos equipamentos baseados em versões embarcadas do Linux.

Bancos de Dados Open Source: presente ou futuro? (Junho)

Nos eventos sobre Open Source, volta e meia surge uma pergunta sobre bancos de dados Open Source. Bem, tenho minha opinião pessoal e quero compartilhar com vocês. Vamos ver se vai gerar muita discordância.

Os softwares de banco de dados são um dos mais importantes componentes de software de uma organização. Neste ambiente as alternativas de software livre já são bastante conhecidas e frequentemente são mencionadas na mídia, como MySQL, PostgreSQL, Ingres e Derby.

O MySQL é um produto de uma empresa privada, a MySQL AB. Seu código é desenvolvido pelos funcionários da empresa e com isso ela garante a propriedade intelectual sobre o produto. Existe uma comunidade envolvida, mas submissões de código são restritas apenas à correção de bugs. Uma pergunta: o MySQL pode ser considerado realmente Open Source, uma vez que não adota o modelo de desenvolvimento colaborativo? O MySQL é ofertado tanto em GPL como sob licença comercial. As duas versões são funcionalmente equivalentes, sendo diferenciadas pelo nível de suporte e certificação. Indiscutivelmente é o banco de dados Open Source mais popular, com o maior mindshare do setor.

Outro software é o PostgreSQL, que tem suas origens no Postgres desenvolvido pela Universidade de Berkeley. Podemos citar também o Ingres, que foi um banco de dados da Computer Associates e agora pertence a uma organização independente, a Ingres Corporation (www.ingres.com) e o Derby, originalmente o Cloudscape da IBM e recentemente doado para a Apache Software Foundation, onde agora é o projeto Derby. O Derby é um banco de dados em Java, geralmente embarcado em outros softwares. A IBM, por exemplo, o utiliza embutido em diversos softwares das famílias WebSphere, Tivoli e Lotus.

Nas minhas andanças pelo mercado tenho visto que na prática os bancos de dados Open Source só aparecem como competidores dos produtos mais avançados nas aplicações pouco sofisticadas ou bem específicas.

Por sua vez, os sistemas de banco de dados proprietários buscam competir com funcionalidades diferenciadoras, principalmente as relacionadas com administração de ambientes complexos; escalabilidade; desempenho com grande volume de transações; alta disponibilidade e capacidade de recuperação rápida; e recursos de data warehousing. Além disso, foi criado um ecossistema de negócios em torno dos principais software de banco de dados proprietários, com diversas empresas independentes oferecendo ferramentas de software complementares (geradores de relatórios, analisadores estatísticos e outros), serviços de suporte técnico especializado e formação de recursos humanos, e assim por diante, o que também cria uma barreira de entrada difícil de transpor por qualquer novo entrante.

Já o ecossistema empresarial criado em torno dos bancos de dados Open Source (onde se

gera dinheiro) ainda é incipiente, sendo formado por pequenas empresas com abrangência de atuação bastante limitada. Em 2006 a MySQL gerou cerca de 50 milhões de dólares em receita, mas ainda é um traço (cerca de 0,03%) no gráfico que mostra o mercado global de bancos de dados relacionais, estimado pelo IDC em 16 bilhões de dólares. Como comparativo, o IDC estima que neste mesmo ano, a receita da IBM com a família de produtos DB2 foi de aproximadamente 3,5 bilhões de dólares.

Qual o papel que os bancos de dados Open Source desempenharão? Na minha opinião estarão atuando (pelo menos nos próximos 3 a 4 anos) na chamada faixa de produtos com funcionalidades comoditizadas, onde as características de preço são as de maior importância. Os usuários típicos serão organizações e aplicações que não precisam de recursos mais sofisticados.

Como avaliar a qualidade de um banco de dados Open Source? Existem diversos critérios que podem e devem ser considerados em uma análise para seleção de um banco de dados. Os níveis de importância das variáveis da análise estão diretamente relacionadas com os objetivos do negócio e das necessidades a serem impostas aos softwares de bancos de dados.

Alguns dos principais fatores a serem considerados são:

- a) Recursos de gerenciamento e administração. São as ferramentas de apoio às tarefas do administrador do banco de dados.
- b) Desempenho e escalabilidade. Os recursos que o software oferece para garantir desempenho adequado, nos volumes de transações que serão demandados.
- c) Recursos técnicos. Disponibilidade de recursos como triggers, stored procedures, cursors, subqueries, capacidade de replicação, recursos de indexação, aderência a padrões (ANSI SQL), particionamento, backup/recovery, suporte a dados não estruturados, independência de plataforma e recursos de segurança.
- d) Custos de Propriedade.
- e) Suporte técnico e disponibilidade de recursos humanos. Abrangência do ecossistema em termos de serviços de suporte e qualificação de recursos humanos.
- f) Disponibilidade de aplicativos.
- g) Recursos de data warehousing e BI.
- h) Recursos de desenvolvimento de aplicações.
- i) Modalidade de licenciamento.
- j) Visão, estratégia e road map do produto.
- k) Tamanho e participação/envolvimento da comunidade.
- l) Modelo de governança adotado pela comunidade.
- m) Base instalada e adoção pelo mercado.

Bem e quanto a uma pergunta que muitos me fazem. Minha empresa deve adotar um banco de dados Open Source? Para mim, para mudar um software de banco de dados deve haver uma estratégia impulsionada por razões fortes e consistentes. Por exemplo, se houver desconfiças que o atual fornecedor esteja saindo do mercado; falta de funcionalidade do software (não é mais adequado às necessidade das novas aplicações da empresa); falta de visão estratégica por parte do fornecedor do software atual; custos de manutenção e

operação muito elevados para o resultado obtido; falta de pessoal gabaritado, que esteja disponível no mercado; carência de consultorias e serviços de suporte externos; relacionamento com o fornecedor cada vez mais deteriorado... Mudar para um banco de dados Open Source simplesmente por questões ideológicas deve estar fora de cogitação, pois banco de dados é muito sério para ser tratado de forma simplista.

OK, e quais seriam então os custos e riscos da migração? Existem custos de migração que não podem ser subestimados. Temos os custos da conversão de dados, custos da codificação, testes, e o que chamamos reconciliação entre as aplicações no novo e no antigo ambiente, sempre considerando que dificilmente conseguiremos fazer uma migração estilo big bang, mas que esta será gradual.

Quanto mais complexas forem as aplicações a serem convertidas, mais custosa será a migração. Esta complexidade pode ser medida pelo número de programas, número de tabelas relacionais, restrições de integridade referencial e tamanho do banco de dados. Existem custos indiretos como a construção de interfaces entre as aplicações já convertidas e as que ainda estão no banco de dados antigo. Também os custos de suporte técnicos aos dois ambientes implicam muitas vezes, em gastos adicionais elevados, principalmente quando o novo banco de dados não for de completo domínio da equipe técnica da empresa.

Em resumo, os custos da migração afetam os cálculos de custos totais de propriedade. A maioria das empresas é extremamente cautelosa em trocar de fornecedor de softwares críticos. O perigo de uma interrupção nos seus negócios decorrente de uma troca mal planejada ou inadequada faz com que os custos de troca possam ser extremamente elevados e desestimuladores. Migrar de um banco de dados para outro é sempre uma tarefa complexa e de alto risco, que só deve ser efetuada quando os benefícios forem claramente demonstráveis.

Open Source Think Tank: the Future of Commercial Open Source (Junho)

Em março deste ano ocorreu o 2º Open Source ThinkTank, nos EUA. Este evento reuniu mais de 100 executivos (CEOs e CTOs) de 80 empresas do mundo Open Source para debater o futuro. O próprio tema do evento diz isso claramente: “The Future of Commercial Open Source”.

A IBM esteve presente e lendo o relatório com as conclusões do evento, achei que seria interessante compartilhar algumas das informações com vocês. Em tempo, estas conclusões refletem a opinião da maioria dos participantes, mas não são necessariamente representam a opinião específica de alguma empresa.

Foram diversos painéis e sessões brainstorming. No painel dos CIOs, as principais conclusões foram que Open Source já é visto como uma opção viável e já não existe hesitação por parte dos CIOs em analisarem opções de Open Source, desde que as funcionalidades do software sejam adequadas. Entretanto, muitos ainda têm receios de adotar os softwares Open Source que não tenham construído um ecossistema saudável e consistente de apoio, com diversidade de ofertas de suporte e treinamento. Também surgiram queixas quanto a dificuldades de alguns produtos Open Source se integrarem e interoperarem, principalmente com ambientes proprietários já instalados. E já está bem claro para todos que Open Source não significa software gratuito embora os custos de propriedade tendem a ser menores que os produtos proprietários correspondentes.

Na sessão brainstorming sobre tendência e desafios do Open Source, identificou-se como principais desafios:

- a) Como monetarizar Open Source?
- b) Como balancear os interesses dos usuários com a comunidade, uma vez que nem sempre os desejos dos usuários estão refletidos nas prioridades das features implementadas pela comunidade?
- c) Como reduzir e conciliar as dezenas de licenças, às vezes incompatíveis entre si?
- d) Quais serão os impactos do GPLv3?

E três pontos adicionais foram também abordados. Um é que a influência do modelo Open Source já está afetando os startups de software, reduzindo os investimentos dos VCs neste setor. E outro, que começa a haver uma carência de “core developers”, uma vez que muitos deles estão sendo contratados diretamente por empresas de software, causando o temor que acabem gerando conflitos entre os interesses de seus novos empregadores e os da comunidade que lideram. E finalmente, que os modelos Open Source e proprietários continuarão convergindo.

Houve também uma sessão brainstorming sobre licenciamento. Uma das principais conclusões é que o modelo de negócios a ser adotado é pré-requisito para a escolha do tipo de licença Open Source, uma vez que cada modelo de negócios baseado em Open Source requer determinadas características de licenciamento. A proliferação de licenças Open Source, incompatíveis entre si é um problema a ser resolvido. Por exemplo, foi citado que um software sob licença GPL não pode ser usado com softwares distribuídos sob licenças

Apache ou Eclipse. E como não poderia deixar de ser, os termos das licenças continuam muito confusos. Poucos conseguem realmente entender o que estes termos querem realmente dizer...A questão do GLPv3 também foi muito discutida e há um receio que acabe gerando forks entre softwares que manterão a GPLv2 e que tenham componentes ou bibliotecas que adotem GPLv3.

Uma terceira sessão brainstorming abordou a questão de como implementar modelos híbridos, aglutinando características dos modelos proprietários e Open Source. A conclusão quase geral é que os modelos estão convergindo e muitos dos fornecedores de softwares proprietários estão adotando características dos modelos Open Source. Por sua vez, os modelos Open Source também adotam características dos modelos proprietários, principalmente a busca pela lucratividade. Começa a ganhar corpo a tendência das iniciativas de Open Source a buscarem lucratividade através, não apenas de suporte e treinamento, mas também por licenças baseadas em subscrição. Também deve se acelerar o processo de aquisição de empresas Open Source pelas empresas de software proprietárias. O momento romântico de Open Source definitivamente já está se tornando história...

A conclusão que posso chegar, é que Open Source já está sendo visto de forma bem mais pragmática e realista. Discute-se francamente modelos de negócio que visem gerar lucratividade. Falar em ganhar dinheiro com Open Source já não é mais politicamente incorreto e nem uma heresia... Está claro que estamos no caminho da convergência entre os modelos baseados em Open Source e proprietários.

O mundo do software não será mais 100% proprietário, mas também não será 100% Open Source. Estamos em busca do ponto de equilíbrio.

De proprietário a Open Source: como fazer a transição? (Junho)

Recentemente levantei um post comentando um evento que debateu o futuro do Open Source, o Open Source Think Tank. Poucas horas depois recebi um e-mail de um empresário da indústria de software, perguntando se deveria ou não colocar seu produto como Open Source. Minha resposta foi específica para ele, mas alguns comentários e visões são genéricos e gostaria de compartilhar com vocês.

Open Source é um movimento que está consolidado. A sua aceitação pelo mercado é crescente e desperta, naturalmente, o interesse de empresários de software quanto à sua viabilidade comercial e a dúvida da maioria é saber até que ponto será vantajoso adotar algum modelo de negócios baseado neste conceito.

Bem, primeiro é necessário compreender as diversas alternativas de modelos de negócio que podem ser baseadas em Open Source. Por exemplo, o produto será totalmente liberado ou apenas uma parte? Existirão versões similares abertas e fechadas ou terão diferenças de funcionalidades? Receitas oriundas apenas de serviços não são as únicas alternativas...

Desenhando o modelo de negócios, escolha o modelo de licenciamento. Contribuições e extensões ao código fonte deverão ser obrigatoriamente liberadas ou seus autores poderão comercializar estas modificações em formato proprietário? Portanto, é necessário também conhecer os diversos modelos de licenciamento, que variam em muito em seu grau de flexibilização quanto à distribuição do código fonte.

Também é importante balancear os interesses da empresa com os da comunidade a ser envolvida. Se o produto não despertar interesse suficiente para gerar massa crítica de contribuidores ativos, o projeto simplesmente não irá decolar.

Como sugestão, desenhe uma estratégia que considere, entre outros fatores:

- a) Qual será a participação da comunidade? O seu produto já está maduro e é necessário apenas contribuições incrementais ou ainda está em alfa, demandando muito esforço da comunidade? Qual o nível de qualificação técnica que será necessário dos contribuidores?
- b) Qual o nível de interesse que o produto poderá despertar na comunidade? Terá condições de gerar e reter massa crítica de contribuidores ativos? Apenas abrir código fonte não gera comunidade.
- c) Qual o modelo de governança que será adotado? Quem vai controlar os direcionamentos evolutivos do produto (seu road map)? Existirão potenciais conflitos de interesse entre a empresa e a futura comunidade?
- d) Qual a perspectiva de aceitação do produto pelo mercado? Adotar Open Source vai aumentar a aceitação do produto?
- e) Será possível gerar um ecossistema de serviços (consultoria, suporte, educação) em torno do produto?
- f) O mercado alvo aceita naturalmente um software Open Source ou será necessário um esforço intenso de evangelização? De maneira geral o mercado aceita Open Source de soluções bem conhecidas e comoditizadas, como sistemas operacionais, suítes de escritório,

navegadores, compiladores, Web servers, etc. Produtos mais específicos, que geram barreiras de saída altas não são facilmente aceitos.

Não esqueça que a relação com a comunidade deve ser bem aberta e franca (a empresa sabe como fazer isso?), deve existir uma estratégia comercial bem definida e deve existir também uma clara proposição de valor para evitar que um concorrente crie uma alternativa gerando uma variante (fork) do seu código, corroendo seu espaço de mercado.

Uma sugestão adicional é estudar alguns projetos de Open Source que estão sendo bem sucedidos. Estude suas proposições de valor, como suas comunidades estão organizadas, seus modelos de governança, a cadeia de valor (ecossistema) envolvido, aceitação pelo mercado, etc. São uma boa fonte de referência e aprendizado. Que tal avaliar (mera sugestão...) projetos como MySQL, PostgreSQL, OpenOffice, JBoss, ActiveGrid, SugarCRM, PHP, Tomcat e Hibernate? E claro, sempre vale a pena olhar os projetos estrela, como Linux e Eclipse.

Um lembrete adicional. Não existe tal coisa como UMA comunidade de Open Source. Existem diversos projetos de Open Source, cada um com sua comunidade e cada projeto tem características próprias. Existem projetos liderados por organizações independentes como consórcios e fundações (Linux Foundation, Eclipse Foundation e Apache Software Foundation), patrocinados por empresas (MySQL) ou mantidos pela comunidade, como PostgreSQL e PHP. Cada projeto pode estar em um stack de software diferente, com grau variado de maturidade e aceitação pelo mercado. Não tem validade colocar tudo no mesmo bolo.

O que motiva o desenvolvedor Open Source? (Junho)

Os posts sobre Open Source tem despertado muito interesse, refletido nos comentários e e-mails recebidos. Temos muito mais a debater ainda. Na verdade o tema Open Source é um dos mais comentados no blog, não só devido a sua importância para a sociedade e a indústria de software, mas pelo fato que ainda estamos na fase de aprendizado. Ganhamos, é claro, muita compreensão sobre este movimento, mas ainda temos um longo caminho a percorrer.

Open Source também é estratégico para muitos governos. A Comunidade Européia, por exemplo, definiu Open Source como uma das principais armas para competir com os EUA na sociedade da informação. Uma análise das 100 maiores empresas de software do mundo (<http://www.softwaretop100.org/>) mostra que mais de 90% delas são americanas. Como a Europa tem uma cultura mais avessa à riscos que a americana e também se ressentida da ausência dos VCs (Venture Capitalists) que impulsionam o Vale do Silício, o movimento Open Source, baseado em desenvolvimento colaborativo, preenche esta lacuna.

Um aspecto do Open Source que vale a pena debater (e que leva a opiniões apaixonadas e divergentes...) é a motivação que leva um desenvolvedor a contribuir com seu tempo e conhecimento com determinado projeto.

Mas, antes de qualquer coisa, creio que é importante lembrar alguns aspectos do dia a dia de uma comunidade de Open Source. Um projeto Open Source pode ser visualizado como um círculo concêntrico, tendo no centro um pequeno grupo de core developers, que escrevem de 80% a 90% do código. Este grupo muitas vezes corresponde a menos de dez desenvolvedores... Depois temos um grupo maior de desenvolvedores que atuam esporadicamente, escrevendo os 10% do código restante. Esta distribuição do esforço de colaboração reflete a Lei de Pareto, com cerca de 10% da comunidade ativa escrevendo 90% do código. Na periferia do projeto aparece um grupo bem maior, de usuários e desenvolvedores que colaboram apontando erros (bugs) e fazendo sugestões, mas não escrevem código. E depois, a imensa maioria de usuários que usam o software, mas não fazem sugestões, não contribuem e nem interagem com a comunidade.

O início da colaboração de um indivíduo com um projeto Open Source começa geralmente como usuário “avançado”, colaborando com sugestões e identificação de bugs, e aos poucos ganhando confiança da comunidade. Escreve então algumas linhas de código. À medida que aumenta seu nível de colaboração (e credibilidade) ele começa a ganhar status até poder fazer parte do core developers team. Mas isto pode levar meses... Ou muito mais!

Outro aspecto que vale a pena lembrar é que não existe uma comunidade Open Source homogênea. Cada projeto tem características próprias, com objetivos e interesses diferenciados, atraindo uma gama diversa de colaboradores, com motivações diversas. De maneira geral quem colabora com um projeto Open Source não colabora com outro.

Portanto, uma pessoa escolhe um determinado projeto de acordo com seus interesses e motivações, quaisquer que sejam eles. E quais são as motivações? Bem podemos pensar em

recompensas financeiras (diretas ou indiretas), aprendizado (chance de se juntar a um grupo mais avançado tecnicamente), motivações sociológicas ou psicológicas, como diversão, ideologia, ego, obter maior status na comunidade, etc.

Claro que os fatores psicológicos e sociais se mantêm ao longo da sua vida como contribuinte ativo do projeto, mas é interessante que observamos que à medida que o colaborador aumenta seu nível de comprometimento com o projeto (mais horas investidas) a motivação começa a derivar para recompensa financeira. Ele aprende que existem chances de ganhar dinheiro, seja direta ou indiretamente. E mais curioso ainda, a maioria dos que pregam que a gift economy (http://en.wikipedia.org/wiki/Gift_Economy) deve ser a única base econômica do Open Source (ganhar dinheiro é uma heresia) faz parte do grupo que ou não colabora (são apenas usuários) ou colaboram muito esporadicamente, praticamente não escrevendo linhas de código.

Herético? Mas façam vocês uma pesquisa informal e provavelmente vão chegar a este mesmo resultado...Portanto na prática, à medida que o envolvimento com o projeto aumenta, a importância da recompensa financeira aumenta...

Na minha opinião isto é natural e saudável. Open Source não é exceção às regras da economia. A bolha da Internet mostrou claramente que as tentativas de ignorar as regras elementares dos fundamentos econômicos torna os negócios inviáveis. Na histeria do fenômeno pontocom imaginava-se que a regra básica de ter modelos de negócio lucrativos poderia ser ignorada. O estouro da bolha mostrou que não... Não podemos esquecer esta lição quando lidando com o Open Source. A gift economy pode ser uma das bases dos valores econômicos, mas não será a única. Ganhar dinheiro é sempre bom!

Implementando Linux e Java em tempo real (Agosto)

Sistemas embarcados são o coração da maioria das máquinas e equipamentos que estão no nosso dia a dia. Entretanto, muitas vezes não reconhecemos estes equipamentos como computadores. Os chips e os sistemas que operam dentro deles estão, na maioria das vezes, escondidos dentro das funções básicas dos equipamentos. Operam de forma invisível para nós e seus contatos com o mundo exterior acontecem através de sensores e atuadores.

Muitos destes sistemas trabalham em tempo real, onde não pode haver demoras ou imprevisibilidades do tempo de resposta. E o que isto significa? Um sistema de tempo real opera sob requerimentos críticos de resposta, ou seja, se uma tarefa não for completada em um período determinado de tempo ela simplesmente terá falhado. Um exemplo simples pode ser um robô em uma linha de montagem. Se o seu sistema de controle não reconhecer determinado objeto em um instante preciso, quando ele estiver passando exatamente pelo ponto da solda, o objeto seguirá sem ter sido soldado e a operação terá falhado. Não existe possibilidade de recomeçar, recuando a esteira para reposicionar o objeto na posição correta e tentar uma segunda vez.

Para garantir esta resposta imediata, em um sistema operacional de tempo-real, um processo de alta prioridade assume o controle do processador de imediato, sem espera. Processos de alta prioridade não esperam por processos de prioridade inferior. Este escalonamento é chamado de escalonamento determinístico (deterministic scheduling). É um mecanismo diferente dos sistemas de uso geral.

Neste contexto temos o Linux crescendo em popularidade. Claro, que para operar em tempo real são necessárias algumas modificações no kernel, para torná-lo determinístico. Bem, e além do sistema operacional? Não podemos esquecer a própria aplicação. O desenvolvimento de sistemas embarcados em tempo real apresenta desafios que não são encontrados nos sistemas tradicionais. Um fator essencial é desempenho. Uma aplicação em tempo real deve operar sob rígidas limitações de desempenho. Como estes sistemas estão cada vez mais complexos, o fator produtividade torna-se também essencial. Devido a estas características, aliadas à maior capacidade dos processadores, começamos a deixar de lado Assembly, Ada e C, adotando C++ e Java. Java, por exemplo, como podemos ver em <http://www.tiobe.com/> é uma linguagem cada vez mais disseminada. Falando em Java, recomendo a revista Mundo Java, onde mantenho a coluna “Tendências em foco”. A publicação é realmente de altíssimo nível.

Mas, como esta dupla Linux e Java opera em tempo real? É realmente possível? Bem, temos um caso concreto para mostrar que funcionam bem, que é a implementação pela IBM e Raytheon dos sistemas de controle (incluem-se command-and-control, navegação, detecção de alvos, controle de armas e sistema de radar) da nova geração de destroyers da US Navy, a série DDG 1000. Este sistema é baseado em servidores blade, operando WebSphere e Java, em cima de Linux, em tempo real!

Uma novidade muito interessante é a adaptação de Java e do WebSphere para operar em tempo real. Com uso de Java aumenta-se em muito a produtividade e amplia-se a base de

desenvolvedores aptos a desenvolver novas e mais complexas aplicações em tempo real. Não é mais necessário contratar-se experts em Ada, C ou Assembly.

Também sugiro a leitura de alguns papers interessantes que descrevem como o WebSphere e Java foram adaptados para operar em tempo real. Acessem <http://www-01.ibm.com/software/webservers/realtime/> para obterem estes papers. Um destes papers é “IBM WebSphere Real Time: Providing Predictable Performance”, que pode ser obtido em ftp://ftp.software.ibm.com/software/webservers/realtime/pdfs/WebSphere_Real_Time_Overview.pdf. Este paper descreve os componentes do ambiente Java em tempo real da IBM (IBM WebSphere Real Time).

Outro paper que recomendo ler é “Creating Predictable-performance Java Applications in Real Time”, obtido em ftp://ftp.software.ibm.com/software/webservers/realtime/pdfs/WebSphere_Real_Time_Technical_Overview.pdf. Neste paper vocês podem ver em detalhes como a J9 (Java Virtual Machine) da IBM funciona e o que foi necessário para adaptar uma JVM para operar em tempo real.

Bem, e não é só: acessem <http://www.alphaworks.ibm.com/topics/realtimejava> para mais informações sobre Java em tempo real, inclusive uma descrição técnica do Metronome, componente de Garbage Collection do Java em tempo real da IBM.

Economia do Open Source (Agosto)

Nesta próxima semana estarei em Belo Horizonte, na edição mineira do Linux Park. A palestra que farei abordará o tema Economia do modelo Open Source.

Quando começamos a falar com mais intensidade em Open Source, não havia muita compreensão dos seus aspectos econômicos, havendo mesmo a suposição que seria uma economia intangível, “gift economy”, com desenvolvedores voluntários desenvolvendo programas sem nenhum compromisso econômico. Esta visão fez com que alguns críticos imaginassem que o Open Source não poderia ser sustentável a longo prazo, pois não haveriam valores monetários para sustentar o ecossistema que eventualmente fosse criado em torno de seus projetos. Também empresas de software com modelos de negócio baseadas em licenças mantiveram distância e chegaram mesmo a propor que este modelo seria anti-econômico por natureza.

Hoje, temos razoável maturidade e experiência prática para analisar a economia do Open Source sob outro ângulo. Sim, existe dinheiro em torno dos projetos Open Source, só que ele não está diretamente ligado ao código, mas ao seu redor, no seu ecossistema.

Indiscutivelmente que Open Source mudou as regras do jogo da indústria de software. É um risco como oportunidade para as empresas já estabelecidas. A grande inovação do modelo é o processo de desenvolvimento colaborativo, que dilui os custos de produção do software por vários atores. Não é que o custo para desenvolver um software aberto seja zero. Ele pode ser tão elevado quanto um software proprietário, mas está diluído por toda uma comunidade. Por exemplo, em <http://www.dwheeler.com/essays/linux-kernel-cost.html> e <http://www.dwheeler.com/sloc/> podemos ver algumas estimativas de custo de desenvolvimento do kernel do Linux. Segundo esta estimativa, se este kernel (no caso 2.6) fosse desenvolvido a partir do zero teria custado mais de 600 milhões de dólares. E uma distribuição completa Linux como a Debian custaria mais de oito bilhões de dólares. Mas, ninguém arcou com este custo sozinho!

Este novo modelo de desenvolvimento colaborativo é que permite desenvolver novos modelos de negócio, onde o preço final do software em si pode ser mínimo. Um software proprietário joga nas costas do seu fabricante todo o custo e o risco de insucesso do projeto. O fabricante precisa de capital, pois a lucratividade vem muito tempo depois, uma vez que um software complexo leva anos para ser concluído, além do tempo necessário para vender cópias suficientes para amortizar o imenso investimento inicial empatado no seu desenvolvimento. Outro grande custo adicional é o marketing e a força de vendas necessários para que o produto seja comercializado com sucesso. No Open Source as regras são outras. Cada participante da comunidade adiciona uma pequena parte ao projeto e recebe em troca um software completo. E quando se disponibiliza o software para acesso via downloads, corta-se outra grande despesa, que são marketing e distribuição.

Bem, e como o ecossistema em torno de um projeto Open Source gera valores monetários? Vamos analisar dois lados da questão: o do integrador e o do produtor de software. Para integradores, que buscam entregar soluções que envolvam hardware e software para seus

clientes, cada real a menos que ele precisa dispende em licenças de software é ou um real a mais em lucratividade ou uma redução de preços nos serviços, o que lhe abre mais opções de mercado.

E a indústria de software? Bem, se a empresa não tem um produto líder de mercado, com poucas chances de disputar uma parcela maior do market share, abrindo seu código fonte ele dilui os custos de evolução e amplia as chances de criar um ecossistema de negócios. Com um novo modelo de comercialização, ele passa a ter potencial para afetar a posição das empresas líderes. Claro que para isso é necessário criar uma comunidade abrangente e entusiasmada, e um ecossistema saudável que gire em torno do software.

Mas, não é só...Open Source abre oportunidades de expansão de negócios e entrada em mercados antes inatingíveis (olha aí o conceito da Cauda Longa), além de incrementar inovação, com a sinergia criada com as comunidades.

E a IBM, como se situa neste contexto? Estamos neste negócio de Open Source desde fins dos anos 90 (quando foi liberado o código fonte do Jikes, compilador Java) e já escrevi um post sobre este tema aqui no blog (IBM e Open Source, em 18 de maio). Os objetivos da IBM com Open Source são: aumentar o ritmo e velocidade das inovações, incentivando o “caldo cultural” de inovação nas comunidades (inteligência coletiva), ser um contribuidor ativo das comunidades e não apenas consumidor, capturar e transformar inovações Open Source em valor para nossos clientes (como exemplos o uso do Linux nos hardwares IBM e a tecnologia Rational fundamentada no Eclipse) e alavancar modelos de negócios baseados em Open Source para entrar em novos mercados e expandir oportunidades de negócio. É, portanto uma estratégia e não um simples oportunismo para explorar uma onda , ou uma reação forçada pela pressão do mercado.

IBM Forum: Open Source, Apache, SCA e Tuscany (1) (Setembro)

Estarei no IBM Forum, o principal evento para clientes da IBM Brasil, todos os dias e uma das palestras que vou apresentar será sobre Open Source e seus modelos de negócio. Quando se fala em IBM e Open Source, imediatamente associamos o Linux. Mas, temos muito mais atividades em Open Source que só Linux. Se vocês acessarem <http://www.ibm.com/developerworks/opensource> vão ver uma lista enorme de projetos em que estamos envolvidos. E chama atenção o nosso envolvimento com a comunidade Apache. Este será o tópico principal da nossa conversa neste post e no próximo.

O Apache é atualmente o servidor web mais popular do mercado e também é um dos ícones do movimento de Open Source. O projeto Apache começou em fevereiro de 1995, através de um esforço combinado para coordenar correções (patches) a um programa httpd, desenvolvido por Rob McCool, da NCSA (National Center for Supercomputing Applications). Depois de vários meses adicionando features e correções, os desenvolvedores do Apache resolveram substituir o software por um novo projeto, desenhado por Robert Thau. Estava claro para os projetistas que seria necessário um novo modelo de gerenciamento de projetos para trabalhar com uma comunidade dispersa de desenvolvedores. A nova arquitetura representou um consciencioso esforço de resolver primeiro o processo de gerenciar o desenvolvimento de colaboradores geograficamente dispersos, sem ligações organizacionais formalizadas. A primeira versão desta nova arquitetura, o Apache 1.0 apareceu em janeiro de 1996.

Foi criada uma organização informal, denominada Apache Group, depois reorganizada como Apache Software Foundation (ASF, www.apache.org), com a missão de coordenar e direcionar o desenvolvimento do Apache e diversos projetos correlatos como Gerônimo, Tomcat, Struts e outros.

Na minha opinião, qualquer um que queira criar uma comunidade para alavancar projetos Open Source deveria adotar o modelo de governança do Apache como benchmark. A sua força de trabalho é constituída por desenvolvedores voluntários, que dedicam apenas parte de seu tempo a esta tarefa. A comunicação entre a comunidade é feita basicamente por listas de discussão, wikis e e-mail. O processo de resolução de conflitos adotado é o de votação, com quorum mínimo de votos. É adotado também o princípio da tecnocracia, para incentivar e motivar a comunidade engajada. Os desenvolvedores são premiados com níveis visíveis de senioridade e respeito dentro da comunidade Apache, com base nas suas colaborações.

A ASF também age como um porta-voz da comunidade, principalmente no estabelecimento de acordos com parceiros comerciais. A ASF tem um board ou comitê gestor de nove diretores. Os diretores, além de suas tarefas administrativas, coordenam comitês de gerenciamento de projetos, cada um com cinco a dez membros. O board autoriza a criação de projetos, mas não os inicia. Estes são concebidos pelos próprios desenvolvedores. O board também não impede a criação de projetos que compitam um com o outro, ou que se sobrepõem em algum ponto, desde que cumpram os guidelines da ASF. Na prática o processo darwiniano é que arbitra quais projetos vão se tornar bem sucedidos.

Existem também comitês focados na organização de eventos e conferências, e mais recentemente com incubação de atividades correlatas ao Apache. A primeira conferência para usuários e desenvolvedores Apache foi a ApacheCon de 1998.

Cerca de 100 membros “de elite” elegem o comitê gestor. Estes membros de elite são convidados pela organização com base nas suas colaborações. As indicações devem ser aceitas pela maioria, em votação, e podem ser revogadas, em casos especiais. Uma parcela significativa dos membros de elite vem da comunidade empresarial, de empresas como IBM, Sun e Red Hat. Estas comunidades de elite são constituídas basicamente por desenvolvedores e tem acesso e autorização para aprovar e fazer modificações no código do Apache. Importante frisar que apenas membros individuais são aceitos no Apache. As companhias, portanto, devem ser representadas por indivíduos.

Toda e qualquer modificação no código deve passar por um processo de votação. Ninguém pode fazer alterações no software sem aprovação da comunidade. Um único membro pode impedir uma modificação, desde que tenha argumentos sólidos que comprovem sua atitude.

O processo tecnocrático adotado pela comunidade Apache é bem diferente dos processos de decisão de empresas comerciais, muito influenciadas por decisões comerciais e por estratégias de marketing. O resultado tem sido um software de alta qualidade, comparável aos melhores softwares proprietários. Tem um paper muito interessante sobre o custo de falhas em sistemas, que pode ser visto em http://weblogs.java.net/blog/malcolmdavis/archive/2006/03/the_economics_o_1.html, onde o autor cita “In Corporate IT development, defect density is normally about .1 or 100 defects per 1K LOC. Commercial is about .05, and some high-end open source projects like Apache and Linux, are around .001 (1 defect per 1K LOC)”.

Outro fator que impulsionou o Apache é a modalidade de licenciamento adotada pelo projeto, bastante liberal e derivada da licença BSD da Universidade de Berkeley. Esta modalidade permite modificação e redistribuição comercial ou proprietária do código, sem obrigatoriedade de retornar as modificações para a comunidade ou pagar pelo seu uso. Esta licença bastante flexível é um dos motivos que levam o Apache a ter um envolvimento bastante intenso com a comunidade empresarial. O Apache é parte de soluções proprietárias como o WebSphere da IBM.

Alguns argumentam que esta permissividade de licenciamento facilita a criação de versões incompatíveis, uma vez que disputas comerciais poderiam fazer com que a indústria de software adotasse direcionamentos diferentes em suas versões Apache. Na prática isto não tem acontecido pelo simples fato que um servidor Web é considerado um software que complementa soluções das empresas, e não seria justificável economicamente para uma organização assumir todo o trabalho da comunidade (desenvolvimento e manutenção) em troca de um produto exclusivo, que não traria vantagens competitivas. Hoje, a base estimada de servidores Apache é de mais de 68 milhões, ou 50% de todos os servidores Web instalados. Dêem uma olhada em <http://news.netcraft.com/>. E em consequência desta grande base instalada, os projetos derivados do Apache também não geram variantes.

A arquitetura do Apache é modularizada, com clara separação das funções básicas do servidor (seu kernel) das facilidades específicas e customizáveis, escritas em módulos independentes, que podem ser seletivamente selecionadas e compiladas para cada configuração. O número de linhas de código deste módulos é muitas vezes superior ao do próprio kernel. O Apache é altamente componentizado e os projetos usam componentes uns dos outros sempre que possível.

A alocação de tarefas também é bastante livre. Uma vez identificada e aprovada uma determinada ação, busca-se um voluntário (ou voluntários) para escrever o código. Existem alguns “core developers”, que criaram partes importantes do código do Apache e tem uma reserva implícita, uma vez que detém bastante conhecimento sobre aquele pedaço de código. Se a mudança for em um destes módulos, eles têm a primazia de pegar a tarefa. Caso não queiram a tarefa, esta é repassada a novos desenvolvedores. Na prática, os “core developers” são responsáveis pela maioria das inovações e modificações importantes, cabendo à comunidade as tarefas de identificar e enviar patches de correção. Caso exista mais de uma alternativa para cada problema, estas são colocadas em discussão e a mais votada passa a ser o caminho adotado.

Uma vez que o código é escrito e testado, ele é colocado nas listas de discussão como um patch para revisão pela comunidade. Caso seja aprovado, ele então é submetido a um dos responsáveis pela manutenção do repositório do Apache, para incluí-lo no fonte oficial.

Cada nova versão tem um “release managers”, um dos “core developers” que se voluntaria para determinar se o software atingiu o ponto de estabilidade necessária para ser liberado como produção. Neste ponto ele é congelado e apenas correções são aceitas. Esta tarefa de gerenciar versões é feita de maneira rotativa entre os membros de maior experiência do projeto Apache.

IBM Forum: Open Source, Apache, SCA e Tuscany (2) (Setembro)

Muitos dos projetos da comunidade Apache estão sendo direcionados para SOA. A sua arquitetura altamente componentizada, com componentes plugáveis, facilita a construção de projetos orientados à SOA. Se analisarmos uma plataforma hipotética voltada a SOA, vamos identificar alguns macro componentes, como uma “core platform” que pode ser considerado a base para execução e integração dos serviços, um “service delivery bus” para roteamento de mensagens e transações, “configuration services”, que organiza a montagem dos componentes em serviços e uma plataforma de comando, para fornecer serviços de gerenciamento e governança.

Os projetos SOA desenvolvidos em torno do Apache atendem a esta plataforma genérica. Por exemplo, os projetos Geronimo e Tomcat são voltados para as funções de “core platform”. Em “configuration services” destaca-se o projeto Tuscany. Como “service delivery bus” temos o Synapse e o ServiceMix. Enfim, vale a pena investir algum tempo analisando os diversos projetos SOA do Apache. Tem muita coisa extremamente interessante. E voltando ao IBM Developerworks, vocês vão ver que a IBM está atuando intensamente em vários destes projetos, como Ant, Cocoon, Excalibur, Geronimo, James e Tuscany, entre outros.

O WebSphere Community Edition é baseado na tecnologia Apache, incluindo o Geronimo e o Tomcat, além de plugins Eclipse e banco de dados Cloudscape. Pode ser baixado “for free” do site da IBM (veja acima). Na minha opinião, a adoção do Apache http Server, Geronimo e Tomcat por uma empresa como a IBM é um sinal inequívoco que o modelo Open Source é sério e que os projetos oriundos da comunidade Apache são projetos altamente profissionais.

Mas vale a pena falar de um outro projeto extremamente interessante, que é o Tuscany. O Tuscany implementa um modelo de implementação SOA chamado de SCA (Service Component Architecture). A proposta da especificação SCA é simplificar o desenvolvimento de aplicações SOA, facilitando o desenvolvimento de componentes. Para isso cria um modelo para construção de componentes independente de linguagens de programação e facilita a construção de aplicações compostas, que podem ser “montadas” a partir destes componentes (lembrem-se do jogo Lego?).

A IBM vem enfatizando SCA. Já temos suporte no WebSphere ESB (vejam <http://www.redbooks.ibm.com/abstracts/sg247406.html>) e apoiamos fortemente o projeto Tuscany. O URL do projeto Tuscany está em <http://tuscany.apache.org/>.

Para maiores informações sobre SCA acessem o site da Open SOA Collaboration (<http://www.osoa.org/display/Main/Home>), grupo formado por diversas empresas, como IBM, BEA, Sun, RedHat e outras, e como dito em seu site, “The Open SOA Collaboration represents an informal group of industry leaders that share a common interest: defining a language-neutral programming model that meets the needs of enterprise developers who are developing software that exploits Service Oriented Architecture characteristics and benefits.”.

Tem um documento sobre conceitos de SCA que recomendo seja lido: “Introducing SCA”, que vocês poderão acessar em http://www.davidchappell.com/articles/Introducing_SCA.pdf.

Na minha opinião, todo desenvolvedor que está buscando implementar soluções SOA deve acompanhar de perto a especificação SCA e os projetos que lhe dão suporte. O Tuscany é um bom exemplo. Ele pode ser encarado como um indicador do nível de desenvolvimento prático da especificação.

Em tempo, a especificação da SCA já foi encaminhada à OASIS, em março deste ano, para ser validada, evoluída e distribuída como um padrão aberto. A OASIS já mantém os padrões de Web Services, como UDDI, WSDL, SOAP e WS-* e agora com SCA/SDO (System Data Objects) oferece os fundamentos arquitetônicos para construção de aplicações SOA. Interessante que entre as empresas que incentivam e apóiam a iniciativa SCA não encontramos a Microsoft, que optou, mais uma vez, por não adotar padrões abertos. Bem, talvez em algum dia a pressão do mercado a obrigue a tornar o .NET um padrão aberto.

Open Source, LongTail e a indústria de software (Setembro)

Estarei no Recife falando em mais uma edição do Linux Park (www.linuxpark.com.br). Um tema que vou abordar será a relação entre Open Source e o conceito da Cauda Longa (Long Tail). Aliás, no IBM Forum da semana passada falei deste assunto na minha palestra, e no intervalo algumas pessoas vieram debater um pouco mais o tema. Eu já o havia abordado de passagem no blog, mas acho que vale a pena aprofundá-lo um pouco mais.

Open Source indiscutivelmente está provocando mudanças na indústria de software e se o juntarmos com o conceito de Long Tail, vamos identificar novas e inovadoras oportunidades de mercado. O conceito da Cauda Longa ficou muito conhecido com o livro de Chris Anderson (“A Cauda Longa”), que resumidamente nos diz que muitos negócios baseados na Web obtêm uma parcela significativa de sua receita, da venda cumulativa de um grande número de itens, cada um dos quais vendido em pequena quantidade. É um modelo econômico que contrasta com os negócios tradicionais, onde as limitações físicas das lojas impõem que estas vendam apenas uma variedade bem limitada de itens, mas que devem ser vendidos em grande quantidade. É o modelo “hit-based” ou de produtos blockbusters. Os sucessos de venda.

O conceito da Cauda Longa vem sendo muito badalado (claro que há muito de hype...) e nem sempre pode ser aplicado. Mas, no software, pode e com certeza poderá trazer impactos significativos na indústria. No site www.thelongtail.com já aparecem algumas discussões sobre seu uso na indústria de software.

O modelo Open Source se encaixa bem no contexto da Cauda Longa e alguns casos demonstram a sua viabilidade quando aplicado ao software: os exemplos do Linux e Eclipse (<http://www.eclipse.org/>) são emblemáticos.

Vejamos, para se encaixar no Long Tail um mercado deve atender a alguns requisitos, como o valor total da cauda deve representar uma parcela significativa do mercado. No software vemos que ainda existem muitas e muitas empresas, principalmente as pequeno porte, que não adotam alguns softwares pelo seu custo. Esta parcela do mercado representa um pedaço substancial do todo.

No exemplo do Eclipse, o custo de se adicionar uma função (um plug-in) é de pequeno custo e não adiciona complexidade ao sistema. Um plug-in não influencia outro. E como o seu custo de desenvolvimento é diluído pela comunidade, não demanda que seja necessário obter uma imensa base de usuários para compensar o investimento, como acontece com o modelo de comercialização tradicional. Assim, pode-se ter plug-ins para frameworks menos populares como Hibernate. Sem o Eclipse, quem precisasse escrever um editor para, por exemplo, o Struts, teria que escrever todo o código do ambiente de desenvolvimento. Com o Eclipse, precisa-se apenas adicionar as funcionalidades do Struts...O framework já existe. Se vocês acessarem <http://www.eclipse-plugins.info/> vão ver que existem mais de 1500 plug-ins!

No Linux, o exemplo típico de aplicação do conceito da Cauda Longa são os sistemas embarcados, com suas funcionalidades altamente específicas. Dêem uma olhada em

www.linuxdevices.com e vocês verão que o Linux roda em centenas de processadores. Isto só é possível porque o Linux é Open Source.

OK, e como podemos dividir o mercado de software? Na minha opinião existe o mercado tradicional, onde a base instalada é importante para garantir a continuidade do produto e o retorno dos investimentos efetuado pelo seu produtor. Estes softwares geralmente atendem funcionalidades principais, como plataformas mais conhecidas, padrões mais usados, etc. Mas existe também o mercado de nichos, que são rarefeitos o suficiente para não despertar interesse comercial, quando olhando os modelos de comercialização tradicionais. Este é o campo ideal para Open Source e Cauda Longa.

Adotando Open Source e Long Tail podemos gerar produtos de software mais especializados e que atendem a nichos de mercado, antes inatingíveis pelos modelos tradicionais. Não é uma mudança nos princípios econômicos da indústria de software? Não merece uma reflexão mais ampla?

Inovação em Open Source: Dynamic Programming Languages (Setembro)

Em dos últimos eventos sobre Open Source em que estive envolvido, um participante me perguntou se os projetos Open Source eram realmente inovadores, uma vez que segundo este profissional, citando textualmente Linux e Open Office, não traziam nada de novo.

Bem, discordo de imediato. Primeiro, o próprio processo de desenvolvimento de projetos Open Source é inovador por excelência. É uma inovação disruptiva, pois muda o tradicional processo de desenvolvimento de software. Basta ver como algumas empresas de software dominantes se sentiram e ainda se sentem ameaçadas por este novo modelo. A teoria Recursos, Processos e Valores (RPV) ajuda a explicar por que estas empresas encontram tanta dificuldade ao enfrentarem inovações disruptivas, como o modelo Open Source. Esta teoria estabelece que os recursos (o que a empresa tem), os processos (como a empresa realiza seu trabalho) e os valores (o que a empresa deseja fazer) definem tanto a sua força como sua fraqueza. Vejamos o caso do Linux. Por que a Microsoft lutou muito para responder ao Linux? De maneira geral as empresas líderes dominam as inovações sustentadoras porque seus valores priorizam essas inovações e seus processos e recursos são planejados para lidar com esse tipo de inovação. E geralmente fracassam diante de inovações disruptivas porque seus valores não priorizarão as inovações disruptivas e os processos existentes na empresa não ajudam a fazer o que precisa ser feito. A mudança no modelo desenvolvimento Open Source produz mudanças nos modelos de negócio e as empresas que sustentam Linux, como a Red Hat, ganham dinheiro de forma muito diferente da Microsoft. Sugiro a leitura do excelente livro de Clayton Christensen, “O Futuro da Inovação” para um aprofundamento destes pontos.

Mas, além disso, o que temos de inovador? O próprio Linux é um exemplo único de um sistema operacional que pode rodar em qualquer computador, do mainframe a um relógio digital. Sem sombra de dúvida é o mais adaptável sistema que existe hoje em dia.

Mas, se formos a outro extremo para buscarmos mais inovações Open Source? Que tal falarmos das “dynamic programming languages” (ou como muitos ainda chamam, de scripting languages...), como Perl, PHP, Python e Ruby? Se visitarmos o www.tiobe.com e analisarmos o Tiobe Index, vemos que, na edição de setembro, PHP é a quarta, Perl é a sexta, Python é a oitava e Ruby a décima linguagem de programação mais utilizada. E, sabemos que a cada dia, pela pressão de resultados mais rápidos e a crescente demanda por aplicações “Internet-centric” como proposto pelo conceito da Web 2.0 (leia-se também mashup applications...) estas linguagens vão crescer e em muito, em utilização. Bem, e não podemos deixar de destacar a linguagem Lua, que em um ano saiu da posição cinquenta para décima oitava!

Estas linguagens são altamente flexíveis e inovadoras em seus approaches de programação. São mais fáceis de aprender que linguagens tradicionais (isto me foi dito por pelo menos dois professores e vários estudantes de computação, com os quais fiz uma pesquisa informal) e aceleram em muito o tempo de desenvolvimento. Alguns comentários que ouvi deles foram que algumas linguagens como Perl e PHP não precisam de conhecimentos de OO (um programador “procedural” se sente confortável programando nelas) e que a Python

tem uma sintaxe simplificada, que permite rapidamente a não programadores desenvolver aplicações. Inclusive lembraram que já existem algumas aplicações muito interessantes e bem conhecidas em cima de Python, como o Zope.

Não é toa que estas linguagens fazem parte da famosa combinação LAMP (Linux, Apache, MySQL e PHP/Perl/Python)) para softwares Open Source. Vejam [http://en.wikipedia.org/wiki/LAMP_\(software_bundle\)](http://en.wikipedia.org/wiki/LAMP_(software_bundle)).

Bem, todas estas linguagens são projetos Open Source, desenvolvidos de forma colaborativa, pela comunidade, com modelos de governança bem definidos e seus “benevolent dictators”, que são os seus mantenedores chefe. Vejam www.perl.org, <http://www.rubyonrails.org/>, <http://www.php.net/> e <http://www.python.org/>.

Então, estas linguagens não são inovadoras? Foram criadas e são desenvolvidas em Open Source. E é por elas existirem que temos hoje as inovadoras aplicações mashup.

Linux Verde? (Outubro)

O assunto conservação de energia em TI está se tornando mais e mais importante. No ultimo Linux World, em agosto, a IBM anunciou (com a participação da Linux Foundation) a iniciativa Big Green Linux (<http://www-03.ibm.com/press/us/en/pressrelease/22006.wss>). A IBM também vem contribuindo intensamente para tornar o Linux mais eficiente em consumo de energia, com o LTC (Linux Technology Center) colaborando com código para que o kernel tenha controle da voltagem e velocidade do clock, criando um estado de processador ocioso chamado de “tickless”. O kernel 2.6.21 já incorpora recursos tickless.

Também devemos dar os parabéns à Intel pela sua iniciativa de criar o LessWatts.org (www.lesswatts.org), que é uma comunidade voltada para gerar inovações de conservação de energia em Linux.

E não esqueçam de visitar o site do workgroup do projeto Green Linux da Linux Foundation (http://www.linux-foundation.org/en/Green_Linux). No seu texto de abertura diz seu objetivo: “The Green Linux Initiative is a working group organized by the Linux Foundation to improve power management in Linux. The working group is facilitating collaboration among kernel developers, hardware manufacturers, system vendors, distributions, and end users to understand and address the requirements for more effective power management in Linux. Better power management is a requirement for nearly every Linux environment”. O site contém links e papers que valem a pena serem vistos. Um deles é do Jon Maddog Hall, que pode ser acessado em <http://www.linuxjournal.com/node/1000309>.

Como vemos, Linux está se tornando um sistema realmente verde!

Repensando cursos de graduação em Open Source (Outubro)

Outro dia estive conversando com alguns alunos de cursos superiores de graduação em Software Livre. Mas nesta conversa ouvi algo que me surpreendeu bastante: eles falaram que nestes cursos aprenderam as disciplinas básica de graduação, como arquitetura de computadores, sistemas operacionais, linguagens de programação e outras (ok, é o currículo básico), mas com relação a Open Source o que tiveram foi apenas aulas práticas no uso do Linux, Apache e outros softwares. Sim, eles confirmaram que foi basicamente um aprendizado no uso destes sistemas. No meu entender, isto não caracteriza uma graduação específica em Open Source!

O diferencial do Open Source é o processo de desenvolvimento colaborativo de software e não o uso dos sistemas Open Source...Para aprender a usar Linux não é necessário um curso de graduação.

Bem, tentamos, então, juntos, montar um pequeno currículo que poderia diferenciar um curso de graduação em Open Source dos demais. Vou detalhar aqui um pouco mais do que acabou sendo gerado pela nossa conversa...Talvez ajude a alguma universidade a repensar seu currículo e criar um curso de graduação Open Source realmente orientado ao processo colaborativo.

Open Source é basicamente colaboração e inteligência coletiva. Precisamos estudar e entender os motivadores e os inibidores para criação de processos colaborativos e para isso é necessário entender o que é uma cultura de participação.

Um ponto importantíssimo neste contexto é criação e motivação de comunidades. É fundamental para o sucesso de qualquer projeto Open Source que uma comunidade ativa e atuante seja mobilizada. A comunidade é o coração dos projetos Open Source. Se for vibrante e ativa, os projetos decolam e evoluem. Uma parcela significativa dos mais de 220.000 projetos do SourceForge (www.sourceforge.net) não conseguem motivar comunidades com força suficiente para dar vida ao projeto. Entender a mecânica de criação e governança de comunidades é um conhecimento fundamental para qualquer um que deseje lançar um projeto Open Source.

Para isso seria bem interessante que o curso usasse como seus estudos de casos algumas das comunidades mais importantes do Open Source como a Eclipse Foundation, Apache Software Foundation e a Linux Foundation. Cada uma delas tem processos e modelos de governança diferenciados (sim, as comunidades Open Source não são iguais...) e podem ser benchmarks para a criação de outras comunidades. Temos aí uma disciplina diferenciadora : “Criação e governança de comunidades”.

O processo de desenvolvimento do software Open Source merece um estudo mais aprofundado. Apresenta características diferentes dos modelos de desenvolvimento comumente adotados nos softwares proprietários. Portanto esta disciplina poderia ser algo como “Desenvolvimento em Open Source: práticas de engenharia de software”.

Outro assunto essencial: modelos de negócio. Sem um ecossistema saudável, os projetos

não serão sustentáveis. Deve existir receita em algum lugar da cadeia de valor e como obter esta receita seriam os principais tópicos desta disciplina. Também deveria conter estudos dos diversos modelos de negócio já praticados em Open Source, não apenas os tradicionais como os de distribuição Linux, mas os das várias startups inovadoras, como Pleyo, Collaborative Software Initiative, e diversas outras.

E finalmente, estudos referentes aos aspectos legais, estudando as diversas formas de licenciamento.

Para concluir uma sugestão: uma disciplina que estudasse o caso real de uma empresa de tecnologia que adota Open Source de forma estratégica e abrangente. Minha sugestão? A IBM.

Tem um paper do Robert Frances Group “The Open Source Ecosystem Matures: Observations and Recommendations”, de agosto deste ano, que diz textualmente : “RFG believes IBM offers IT executives a useful case study of how the open source ecosystem has evolved and matured, and largely effective response to and leverage of that evolution and maturation. IBM is adopting elements of the open source ecosystem in ways that benefit the company and its customers and partners”. E mais a frente: “RFG believes IBM has demonstrated one of the most comprehensive, integrated, and pragmatic approaches to open source ecosystem support. The company has both an understanding of and commitment to open communities and open standards, as well as to open source technologies that integrate well with other alternatives in ways that enable business value for developers and enterprise customers”.

O paper também reproduz uma frase de Jeff Smith, VP Open Source and Linux Middleware da IBM, que reflete a importância do Open Source para a companhia: “Open Source is important and fundamental for us. Our goal is to build advantages from open source into IBM’s core businesses, rather than to build something interesting on the side”.

Bem, já temos material para começar a pensar em um novo curso de graduação...Quem sabe um dia ele existirá?

Empregabilidade e o profissional de TI (Outubro)

Nesta última semana estive na Universidade Estácio de Sá, no Rio, participando de uma palestra do projeto “Conferencistas da Estácio”. Este projeto visa levar profissionais para compartilhar suas experiências com os estudantes da Universidade e é uma iniciativa muito positiva.

Bem, o tema foi empregabilidade e a carreira profissional. Para preparar a apresentação recorri a alguns materiais muito interessantes, dos quais posso destacar o livro “O Futuro dos Empregos”, de Thomas W. Malone e uma pesquisa patrocinada pelo SIM (Society for Information Management), denominada “The Information Technology Workforce: Trends and Implications 2005-2008”. O relatório completo só está disponível aos seus associados.

A tese do livro, que eu concordo, é que a convergência de fatores tecnológicos e econômicos (como a globalização e o barateamento das tecnologias de informação e comunicação) está permitindo uma mudança muito profunda nas organizações e consequentemente na maneira de se organizar o trabalho. As empresas irão cada vez mais se tornarem menos centralizadas e hierárquicas e mais abertas e colaborativas. Muitos casos de novos modelos de organização já estão sendo demonstrados na prática, embora nem sempre os avaliamos com a profundidade que merecem. Por exemplo, a organização do trabalho em torno das comunidades de Open Source e iniciativas como a Wikipedia.

Um projeto Open Source pode ser visto como uma empresa sem limites geográficos (os participantes estão espalhados pelo mundo), sem hierarquias rígidas, com decisões descentralizadas (cada participante tem o poder para tomar decisão de onde e como colaborar), baseada na meritocracia e em valores que não apenas os monetários. O Wikipedia cria um modelo de trabalho ainda mais flexível, com as decisões de escrever um verbete e/ou editá-lo sendo da responsabilidade exclusiva de cada participante. Vale a pena ler o livro.

Mas, e quanto ao futuro profissional de TI? A pesquisa, feita principalmente com decisores de TI, reflete o mercado americano, com suas características e demandas próprias. Não podemos transplantá-las diretamente para cá, mas muitos de seus resultados poderão nos servir de referência.

Para a questão quais os skills que deverão ser mantidos in-house, as empresas pesquisadas apontaram principalmente funções que demandam habilidades não técnicas, como gerência de projetos, conhecimento de processos, da companhia e da indústria. As funções técnicas (desenvolvedores, suporte, help desk...) tendem a ser cada vez mais terceirizadas. No caso dos EUA muitas vezes para outros países, dos quais, esperamos, o Brasil seja contemplado com uma parcela significativa.

Uma frase da pesquisa, no item “Skills that are sourced”, é sintomática : “All ten skills shown in figure 4.3 are technical, with programming leading the pack, followed by support functions, including system testing and desktop support/help desk. The only two technical

skills that were considered critical to keep in house, system design and analysis, are also frequently sourced”.

Outra questão interessante é que mediu as necessidades e perfis dos profissionais para os próximos anos. No item “Skills predicted to be more critical by 2008”, a pesquisa mostrou “the top four skills are all from the business domain category. One skill is the IT administration category, IT governance, was identified as newly important, as were two skills from the sourcing category: managing third-party providers and sourcing strategy”.

Começa a ficar claro que o perfil profissional de TI está mudando. Precisaremos cada vez mais de profissionais mais versáteis, com sólidos conhecimentos tecnológicos, que conheçam bem negócios (a empresa e sua indústria) e que sabem como tirar proveito das inovações tecnológicas.

Aqui está uma pergunta que não quer calar: quantas universidades estão constantemente monitorando as tendências do mercado de trabalho e ajustando seus currículos?

Desenvolvendo software no modelo Open Source (Outubro)

Em um post anterior em que comentava algumas deficiências dos curso de graduação em Open Source, citei que um dos pontos que mais sinto falta é uma disciplina que estude os processos de desenvolvimento colaborativos, típicos do modelo Open Source.

A palavra chave no mundo Open Source é colaboração e uma frase do livro “The Success of Open Source” expressa isso claramente: “Open Source is a way of organizing production and making things jointly. It promotes software reliability and quality by supporting independent peer review and rapid evolution of source code”.

E como o processo de desenvolvimento Open Source acontece? Se o estudarmos sob a ótica de engenharia de software (engenharia de software não é só codificação, mas segundo definição do IEEE é “a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software”), vemos que existem diferenças significativas com relação aos processos tradicionalmente adotados nos desenvolvimentos de softwares proprietários.

Vamos analisar e comparar as fases tradicionais, como análise de requerimentos, design, implementação, integração, teste e pós-entrega entre os modelos Open Source e proprietários. As diferenças são bem interessantes.

Bem, não existe uma fase formal de análise de requerimentos para um software Open Source. O seu idealizador e o “core team” desenvolvem uma visão e o software acaba sendo idealizado pela contínua discussão entre os membros da comunidade. Também não existe nenhuma formalização das decisões (um documento oficial), a não ser o arquivamento destas “discussion lists”.

O modelo Open Source também se diferencia na que seria a tradicional fase de desenho. De maneira geral o software é implementado pelos membros da comunidade, cada um com suas próprias idéias e as melhores implementações são selecionadas e incorporadas à árvore de código (repositório do código fonte). Não existe documentação formal, e as decisões são baseadas em extensas discussões entre os membros da comunidade. As decisões de design e os códigos fonte que as implementam são selecionados de acordo com os critérios estabelecidos pelo modelo de governança do projeto. De maneira geral, os projetos Open Source são modulares, o que permite o desenvolvimento colaborativo. Um projeto monolítico impede que as tarefas sejam elaboradas em paralelo e é um impeditivo à forma de trabalho colaborativo. Na minha opinião, apesar de não haver uma fase formal de design, é indiscutível que muitos dos projetos Open Source tem um desenho altamente eficiente e inovador. Basta ver os exemplos do Linux, Apache e Eclipse, entre diversos outros.

A fase de implementação é central ao projeto Open Source. No meu entender, o processo de engenharia dos projetos Open Source é “implementation centric”, onde tudo gira em torno da implementação. O modelo também apresenta algumas características diferenciadoras: acontecem várias implementações ao mesmo tempo e a melhor opção é

selecionada, e existe um processo contínuo de peer review, com muitos desenvolvedores e usuários inspecionando o código e gerando comentários e eventuais correções e refinamentos. Na minha opinião, este approach “implementation oriented” torna o projeto Open Source mais eficiente e requer menos skills quando comparado ao tradicional approach “design oriented”.

A fase de integração também é crítica aos projetos Open Source, principalmente porque são softwares altamente componentizados. O processo de integração também é altamente participativo, havendo contínuas discussões entre os membros da comunidade até que as decisões de integração sejam tomadas.

O modelo Open Source também apresenta diferenciação na fase de teste, com extensiva participação da comunidade na inspeção e depuração do código. Isto só acontece porque o código fonte está disponível a todos. Aqui um parêntesis. Quando analisamos uma comunidade Open Source identificamos três camadas básicas. A primeira é constituída pelo chamado de “core team”, que são os membros mais atuantes e acabam sendo responsáveis por 80% a 90% do código implementado. Geralmente são poucos, não mais que algumas dezenas. Depois temos um grupo bem maior de colaboradores esporádicos, que contribuem com idéias e código, mas que acabam desenvolvendo apenas 10% a 20% do código implementado. Contam-se às centenas ou milhares, dependendo do tamanho da comunidade. E temos uma imensa base de usuários que acessam o código, inspecionam e geram idéias e discussões, mas não se envolvem diretamente com a programação. Lembro aqui uma frase lapidar que expressa bem esta idéia, chamada de Lei de Linus: “given enough eyeballs, all bugs are shallow”. Esta “lei” foi formulada e batizada por Eric S. Raymond, em seu livro “The Cathedral and the Bazaar”, que aliás todos que se interessam realmente em estudar o modelo Open Source deveriam ler...

Temos finalmente a fase post-delivery, onde também o modelo Open Source se diferencia significativamente: toda a gestão é feita via Web e a maioria dos projetos implementa mecanismos eficientes de reporte e acompanhamento de bugs, correções e novas versões. O modelo de desenvolvimento colaborativo Open Source é um subproduto da Internet! Não existiria sem e-mails, mailing lists, fóruns, web sites...

Mas as diferenças não terminam aí...Se analisarmos a estrutura organizacional dos projetos Open Source veremos que ao contrário dos processos tradicionais onde existe uma hierarquia na equipe, com gerentes de projeto nomeados pela empresa (top down), temos um modelo de voluntariado, com lideranças sendo conquistadas (e não indicadas) e assumidas por opção (o membro pode aceitar ou não assumir maiores responsabilidades no projeto). Se olharmos mais de perto as comunidades Open Source vemos que existe uma estrutura social bem definida, com base na meritocracia, que permite a quem mais contribui, assumir papéis mais importantes no projeto.

Um comentário final: um projeto Open Source não necessariamente termina...Se uma comunidade se desinteressa e abandona o projeto, uma outra comunidade pode assumir seu lugar. Também, nada impede que uma parte da comunidade opte por criar um projeto alternativo (forking), orientando sua evolução em uma outra direção, diferente do projeto original.

Portanto, entender Open Source não é apenas usar Linux ou Apache, ou mesmo gastar horas debatendo as diferenças conceituais entre “Software Livre” e “Open Source”. É muito mais que isso. É uma mudança nos paradigmas de desenvolvimento de sistemas, com um modelo participativo e colaborativo.

Sugiro que quem estiver interessado em se aprofundar mais no assunto, acessar o site do Institute for Software Research, ISR, na seção Open Source Software Development, em <http://www.isr.uci.edu/research-open-source.html>. Vocês terão acesso a papers e informações muito legais.

Eclipse e Open Innovation Network (Outubro)

Tive a grata oportunidade de assistir a uma palestra de Mike Milinkovich, diretor executivo da Eclipse Foundation. Como volta e meia me perguntam sobre o Eclipse, creio que será interessante compartilhar com vocês as informações que obtive então.

A Eclipse Foundation (www.eclipse.org), foi criada em 2004, a partir do projeto Eclipse, criado pela IBM em 2001. No site vocês tem acesso ao histórico da iniciativa e uma visão bastante completa do que é a Eclipse Foundation, seu modelo de governança, o processo de desenvolvimento adotado e detalhes do seu modelo de licenciamento. Vale a pena ler com atenção. Aliás, a Eclipse Foundation, a Apache Software Foundation e a Linux Foundation são comunidades benchmark para qualquer um que esteja interessado em conhecer o movimento Open Source além da superficialidade dos debates se a terminologia correta deve ser Open Source ou Software Livre.

A importância do Eclipse na indústria de software pode ser medido por alguns comentários publicados na mídia, como o InfoWorld de janeiro de 2006 que disse “..the Eclipse open source tools platform has come into its own, emerging as both an alternative to Microsoft...in the application development space and the de facto standard for developing Java”. Outros textos que valem a pena citar “Eclipse has won, what next for Eclipse”, dito pelo analista Carl Zetie do Forrester Research e “This year we find that six in ten respondents use Eclipse as their primary IDE...”, publicado a partir de pesquisa com desenvolvedores, efetuada pelo Evans Data, em setembro de 2006.

Mas, porque o Eclipse faz tanto sucesso? Na prática ele cria um modelo de Open Innovation Network, onde empresas concorrentes podem criar redes de inovação, cooperando no desenvolvimento de softwares Open Source, que servirão de base para produtos específicos (proprietários), com os quais concorrerão no mercado. O conceito de Open Innovation quebra o paradigma tradicional da P&D feito a portas fechadas, por uma única empresa. Este conceito trata a P&D como um sistema aberto onde tanto idéias externas e internas são debatidas e as melhores alternativas são selecionadas. Existem dois livros que recomendo a sua leitura (estão na minha biblioteca), que descrevem este conceito em maiores detalhes. Ambos são de Henry Chesbrough. Um deles é “Open Innovation, the new imperative for creating and profiting from technology”, e o outro é “Open Business Models: how to thrive in the new innovation landscape”.

Vamos dar um exemplo prático do que significa criar uma comunidade voltada para Open Innovation. O Eclipse é aberto a todos, sob as mesmas regras. Não existem regras que excluam ou minimizem a participação de quaisquer contribuidores, mesmo que eles sejam concorrentes diretos entre si. Todas as discussões e deliberações dos projetos são transparentes a todos, abertos e públicos. O processo que governa os projetos Eclipse é meritocracia: quanto mais você contribui, mais responsabilidade pode obter. Um projeto que representa esta rede de inovação é o RAP (Eclipse Rich Ajax Platform), que é uma plataforma para criação de aplicativos Rich Client baseados em Ajax. O RAP 1.0 foi liberado recentemente e o anúncio oficial pode ser lido em http://www.eclipse.org/org/press-release/20071015_raprelease.php. O RAP é um subset do

RCP (Rich Client Platform) que é a base da tecnologia cross-platform chamado Open Client adotado em diversos softwares da IBM como Expeditor, Notes e Sametime.

O sucesso do Eclipse, é portanto, não apenas fruto de uma comunidade Open Source vibrante e atuante, mas também de um ecossistema saudável onde empresas de software (concorrentes ou não) trabalham juntos na criação de uma plataforma que servirá de base para seus produtos e serviços. Em resumo, as empresas cooperam na construção da plataforma e competem em produtos específicos, construídos em cima desta mesma plataforma.

Este é um modelo que pode ser copiado para criar outras comunidades de desenvolvimento de softwares na forma participativa. Aliás, o Eclipse representa o estágio de maior maturidade na adoção de Open Source, que geralmente começa com uma simples adoção de softwares (a empresa é usuária de Linux e Apache) até chegar a este nível mais estratégico e agressivo de uso de Open Source: colaboração e co-criação de softwares com a comunidade e outras empresas, mesmo concorrentes. Na minha opinião, uma empresa que indiscutivelmente está no nível mais avançado de maturidade Open Source é a própria IBM.

Android & Linux & OpenSource (Novembro)

Agora no início de novembro vimos o anúncio do Google, em parceria com diversas operadoras e fabricantes de celulares, de uma plataforma aberta para desenvolvimento de software para estes equipamentos móveis. A aliança foi batizada de Open Handset Alliance (<http://www.openhandsetalliance.com/>). O Press release está em http://www.openhandsetalliance.com/press_110507.html.

A plataforma, denominada Android, baseada em Linux e inteiramente Open Source, se propõe a criar um ecossistema para desenvolvimento de aplicações para celulares. Qual o resultado esperado pelo Google? Criando uma plataforma única, possibilita um avanço mais rápido no desenvolvimento de aplicações inovadoras, aumenta a demanda pelo uso de novos serviços e gera maiores oportunidades de advertising, de onde o Google obtém sua receita.

O Android é um stack inteiro de software, composto de sistema operacional (construído em cima do kernel do Linux) que implementa ferramentas de desenvolvimento e APIs específicas para os desenvolvedores criarem suas aplicações móveis. A força do projeto se baseia no conceito de colaboração (leia-se Web 2.0 e Open Source) e um grupo bastante heterogêneo de empresas já associou, de operadoras a fabricantes de celulares. O stack de software será distribuído sob a licença da Apache Software Foundation (<http://www.apache.org/licenses/LICENSE-2.0>). As estimativas são que em meados de 2008 já estejam aparecendo no mercado os primeiros celulares com software baseado no Android.

O SDK (System Development Kit) para o Android foi liberado há poucos dias e é baseado no Eclipse. Roda em diversos sistemas, como Windows e Mac, além, é claro do Linux Ubuntu. E para acelerar o desenvolvimento de softwares usando este stack, a aliança criou o Android Developer Challenge, com prêmios de até dez milhões de dólares.

O mercado de celulares está em franco crescimento. Já são quase três bilhões no mundo (pelo menos uns 120 milhões aqui no Brasil) e cada vez mais eles se tornam mais aptos a serem um equipamento de acesso à Internet.

O cenário atual de aplicações para estes equipamentos é espinhoso. São pelo menos três stacks de software diferentes e os celulares não são padronizados, o que demanda um esforço muito grande para desenvolver e testar os aplicativos. Além disso, de forma diferente da Web, onde qualquer pessoa tem acesso a qualquer um dos milhões de sites disponíveis na Internet, no cenário atual dos celulares, na maioria das vezes, o acesso do usuário é restrito apenas aos sites autorizados pelas operadoras. O resultado é que apesar de termos muito mais celulares que PCs, o uso da Internet móvel ainda é bem restrito.

O que o Android vai trazer de mudanças? Bem, acredito que os custos de desenvolvimento de aplicações tenda a cair substancialmente, pois não haverão tantos retrabalhos de portar aplicativos de um celular para outro, como também não haverá mais necessidade de pagamento de royalties para uso de tecnologias como Windows Mobile ou Symbian. Além disso, é provável que muito código seja desenvolvido de forma colaborativa e aberta, reduzindo mais ainda os custos. O resultado final será uma barreira de entrada bem menor, abrindo espaço para novos entrantes. Atenção, atenção: esta é uma excelente oportunidade para empresas e desenvolvedores brasileiros, que já tem experiência no desenvolvimento de celulares, uma vez que vários jogos de sucesso mundial foram desenvolvidos aqui. E pelo efeito de rede, com mais usuários interessados em usar Internet pelos celulares, com aplicações inovadoras, (nada impede de outras empresas como Yahoo adotarem o Android, aumentando a demanda...) as operadoras abrirão seus modelos fechados para incorporar novas modalidades propostas pela plataforma aberta. Ou seja, todos saem ganhando.

Bem, nem todos: quem ganha dinheiro com royalties de licenciamento, como a Microsoft com seu Windows Mobile, só tem perder...Mais um desafio para o pessoal de Redmond!

OK, vamos explicar melhor este último comentário... Coincidentemente, estava lendo um relatório da VDC, "Linux in the Embedded Systems Market", que pode nos ajudar a entender um pouco mais o que será este impacto.

O modelo atual dos fornecedores de tecnologia é baseado fortemente em pagamento de royalties, o que encarece o custo final e conseqüentemente o preço dos equipamentos para o usuário final. Por exemplo, em 2006, dos quase 600 milhões de dólares movimentados pelo mercado de sistemas operacionais em celulares, 82,2% foram pagamentos de royalties. As empresas líderes neste segmento, em receita de royalties são a Microsoft e a Symbian.

O Linux quebra este paradigma. A maioria dos projetos de software embarcado não usam versões Linux de distribuidores oficiais e mesmo quando usando versões oficiais, os modelos de negócio dos distribuidores é baseado em subscrição anual e não pagamento de royalties.

Já existe uma forte tendência de uso de Open Source para sistemas embarcados e a vinda do Android vai realmente anabolizar o uso do Linux nos celulares. Na minha opinião o Android representa o ponto de inflexão para Linux Mobile e acredito que a partir de agora o Linux deve crescer bem mais rapidamente em celulares. Um outro ponto positivo, no meu entender, será uma mais rápida tendência à padronização do Linux no contexto dos sistemas embarcados, pois vai existir um forte fator catalizador para isso.

Desenvolvendo Software no Modelo Open Source (Novembro)

Há poucas semanas levantei um post abordando o processo de desenvolvimento de softwares no modelo Open Source. Recebi diversos emails, que geraram boas conversas sobre o assunto. Bem, acho que vale a pena voltar ao tema. Os princípios e as práticas adotadas pelo Open Source diferem em muito dos adotados pela engenharia de software tradicional. Mas, não impedem que excelentes softwares sejam produzidos, muitos deles extremamente complexos e sofisticados, como o próprio kernel do Linux, que em sua versão 2.6.22 tem mais de 8,5 milhões de linhas de código.

Mas, curiosamente, este assunto praticamente não foi abordado nos inúmeros eventos de Open Source (ou Software Livre) que aconteceram durante o ano. E, na minha opinião, é o principal diferenciador do Open Source. Acredito, inclusive que o desenvolvimento de software colaborativo e participativo, será o modelo das relações de trabalho a ser adotado no futuro. Aliás, antes do trabalho no campo (típico da sociedade agrícola) ser substituído pelo trabalho nas fábricas, símbolo da sociedade industrial, não havia o emprego, como o concebemos hoje, na sua formalidade de trabalhar em troca de um pagamento estipulado e definido anteriormente.

Mas, pensemos “out-of-the-box”: o emprego é apenas uma das formas de viabilizar e definir o trabalho. À medida que a sociedade do conhecimento se dissemina, é provável que cada vez mais “trabalhemos” mas sem “empregos” como o conhecemos hoje. As relações de trabalho, as áreas de RH e a legislação trabalhista terão de mudar de forma drástica. E pessoalmente, vejo o modelo de trabalho adotado pelas comunidades de Open Source como precursora desta nova forma de “emprego”. Daí que insisto em voltar ao assunto, explorando um pouco mais a sua dinâmica e organização.

Uma das mais marcantes diferenças no Open Source é que o software é desenvolvido por pessoas distribuídas globalmente, que na maioria das vezes não se conhecem pessoalmente. E coordenadas, não por estruturas hierárquicas com gerentes formais, mas por processos de governança baseados na meritocracia, conquistados e não impostos. Os desenvolvedores engajados nos projetos Open Source provêm seus próprios recursos, pois usam seus próprios computadores, provêm o acesso à Internet e escolhem suas próprias ferramentas de desenvolvimento. Não estão subordinados a um gerente que cobra presença ou horário, engajam-se nas partes do software que mais lhes interessam, usam suas próprias ferramentas e seus desktops e notebooks...E mesmo assim, geram projetos que muitas vezes são bem superiores aos projetos executados por equipes formais, subordinadas a processos padronizados, com gerentes profissionais... Esta é uma liberdade que temos dado pouca atenção...Não vale a pena entender um pouco mais este fenômeno?

OK, primeiro insight. O fato do modelo Open Source não seguir as regras tradicionais da engenharia de software não significa que seus processos sejam deficientes, mas apenas que são simplesmente diferentes. Um exemplo simples: no modelo tradicional adota-se o critério de aprovar e então implementar, isto é, após a aprovação das especificações, é que as alterações são efetuadas. No modelo Open Source, o critério é outro: implementa-se e

depois aprova-se. Ou seja, implementa-se as alterações, muitas vezes são várias alternativas, e escolhe-se a melhor.

O modelo Open Source substitui o formalismo e a rigidez hierárquica, aliás, típicas da sociedade industrial, por uma rede social, altamente interativa e colaborativa. Não será um novo paradigma da forma de se trabalhar? Afinal, se estamos desenvolvendo artefatos da sociedade do conhecimento (software) por que adotar meios e práticas da sociedade industrial?

Participar de uma rede social e conquistar respeito e reconhecimento dos pares é uma forma de valorização que complementa a tradicional forma de remuneração monetária. Claro que não se vive sem dinheiro, mas começamos também a valorizar o capital social. Talvez esteja aí um dos principais motivadores que fazem as pessoas investirem tempo e conhecimento para escreverem software sem remuneração direta. Algumas pesquisas mostram que é comum desenvolvedores trabalharem em mais de um projeto Open Source. E, claro, os membros da elite dos projetos Open Source (os core team) não apenas conquistam respeito da comunidade, mas conseguem recompensas financeiras substanciais, muitas vezes contratados por empresas de software para continuarem liderando seus próprios projetos.

Explorando mais a questão da rede social, vemos que os membros dos projetos Open Source tem que conquistar seu espaço pelos próprios méritos. Começam como usuários, fazendo downloads e usando o software, passam para “add-on developers”, colaborando de forma pontual, resolvendo bugs e submetendo patches. Neste estágio ainda estão na periferia dos projetos...Posteriormente, de acordo com a qualidade e intensidade das colaborações, podem assumir papéis mais importantes, e escalar a rede social, atuando inclusive como moderadores de fóruns de discussão.

De maneira geral, para ascender de posição, devem ser convidados e aprovados pela comunidade. Não existe um plano de carreira pré-definido ou avaliações formais. O ápice do prestígio na rede social é fazer parte do core team, onde são responsáveis pela orientação dos projetos. São a elite dos projetos Open Source. Existe dinamismo no processo, com aos papéis e ascendência social dependendo basicamente do esforço, motivação e comprometimento (leia-se contribuição) do indivíduo. Vejam que é uma organização bem diferente das que vemos nas empresas, onde as alocações das pessoas aos papéis são definidas pela empresa.

Outro insight...Não existem formalismos ou especificações prévias em projetos Open Source. Mas o “informalismo” do Open Source (na forma de emails, fóruns de discussão, blogs, wikis, etc) tomam o lugar de definições e documentações formais e não impede a geração de softwares de alta qualidade. Será que não é uma disrupção no processo de gerar conteúdo? O Open Source é “implementation-centric”, o que significa que as ações ocorrem em torno do próprio desenvolvimento, com a inteligência coletiva da comunidade direcionando o projeto. Sem passar por esta fase de especificação, tão celebrada e exigida pelos modelos tradicionais de engenharia de software, tivemos o Linux, o Apache, o Eclipse, o Wikipedia e centenas de outros casos de sucesso... E fica uma pergunta...Será que é realmente necessário ter uma especificação formal e prévia do software?

Outro aspecto para pensarmos...Um projeto Open Source cria uma organização virtual, aglutinando indivíduos com conhecimentos diversos, mas correlatos ao escopo do projeto. Por exemplo, para um projeto de sistema operacional, os indivíduos tem que conhecer as características internas à sistemas deste tipo, como schedulers de processador, file management, etc. Como o processo é livre (cada um escolhe onde vai colaborar) e não existe uma metodologia ou padrão de desenvolvimento a seguir, as regras são mínimas. Mas apesar de mínimas, elas orientam o processo e a própria comunidade se auto-ajusta para colocar as pessoas que mais conhecem determinados aspectos do software aos respectivos componentes. Um subproduto da organização virtual, globalizada e sem barreiras hierárquicas é um maior impulso à inovação. Não existem os tradicionais limitadores impostos pelo padrão de pensamento vigente em uma determinada empresa.

Portanto, analisando com mais detalhes as comunidades Open Source vemos os embriões de uma nova forma de trabalho. É uma organização virtual, aglutinada por interesses comuns, desenvolvendo suas atividades de forma aberta, participativa e colaborativa. Criam suas regras sociais e modelos de governança, criam seus próprios métodos de trabalho e tudo isso com a liberdade que não vemos nos modelos tradicionais de engenharia de software. Querem um exemplo? Vejam as regras de etiqueta de uso do Bugzilla, em <https://bugzilla.mozilla.org/page.cgi?id=etiquette.html>. Aliás, eu imagino que para se ter liberdade no uso do produto (usar, distribuir, etc), deve-se ter liberdade no seu processo de criação.

Bem, a idéia deste post é instigar...Será que não estamos gastamos muito tempo debatendo o resultado final do modelo Open Source (acesso ao código fonte, modelos de licenciamento, etc...) e estamos deixando de lado as mudanças mais profundas que ele traz? Acho que temos aí um bom assunto para discussões!

Open Source: balanço do ano de 2007 (Novembro)

Estamos chegando ao final de mais um ano. Este ano participei de dezenas de eventos sobre Open Source e creio que vale a pena fazer um balanço do ano...Do que vi e ouvi nestes eventos, debates, reuniões, fóruns, etc.

Está ficando claro que Open Source é muito mais uma tecnologia social que simplesmente programação de software. Sua essência é colaboração, criação de comunidades e redes sociais. Open Source pode ser definido como um modelo técnico e social de desenvolvimento de artefatos de software baseado no desenvolvimento colaborativo.

Software é uma forma de representar conhecimento... E um novo sistema de riqueza baseado em conhecimento ainda é um território escassamente mapeado. O conhecimento, ao contrário do petróleo ou do carvão, por exemplo, quanto mais usamos, mais criamos. Esta é uma diferença significativa nos princípios fundamentais da economia, que sempre foi definida como a ciência da alocação dos recursos escassos. Mas, conhecimento é inesgotável!

Por isso no início do movimento Open Source houve muita desinformação e mesmo negação da sua importância.

Também está bem nítido que o discurso ideológico da sua fase inicial e romântica está ficando em segundo plano. O Open Source não é antagônico ao software proprietário, mas permite uma coexistência entre os diversos e alternativos modelos de negócio.

Um aspecto interessante das comunidades que gravitam em torno do Open Source é a possibilidade destas serem globalizadas, representando interesses de múltiplas culturas, línguas ou etnias. Aliás, muitas delas já o são. Um exemplo prático de colaboração multicultural é o Wikipedia. A língua portuguesa está em oitavo lugar com mais de 340.000 artigos. Mas vocês vêem até verbetes em cherokee...Aliás, se quiserem conhecer um pouco desta língua acessem o YouTube e ouçam a canção Amazing Grace em cherokee, em <http://www.youtube.com/watch?v=UvYIjFtPQEk>.

O movimento Open Source incentiva a criação de redes, inclusive a cooperação entre projetos. É muito comum partes de determinados softwares de um projeto serem inseridos em outros projetos. Cria-se muitas vezes processos de co-evolução, com os projetos evoluindo de forma interdependente. O resultado dos projetos não beneficia apenas a uma empresa ou indivíduo, mas a toda uma comunidade interessada no uso daqueles softwares.

Ok, e que podemos concluir de tudo isso? Como evoluímos de um ano para outro?

Vou me atrever a criar um modelo de medição de maturidade de uso do Open Source, algo como o CMM para Open Source. Que tal chamarmos de Open Source Maturity Model? De maneira geral, podemos colocar as empresas (usuárias e desenvolvedoras de software) em um gráfico de evolução ou maturidade, considerando seu comprometimento com Open Source. Bem, esta é uma simples proposta para discussões...A idéia do Open Source é

começar um assunto e incrementá-lo de forma colaborativa. Vamos ver se vai dar certo...

O primeiro estágio é a negação, que chamarei de estágio zero (0). Neste estágio a empresa luta contra o Open Source, ou por desinformação ou por sentir-se ameaçada em seu modelo de negócio.

Depois aproxima-se e faz as primeiras tentativas, sendo usuária de alguns poucos softwares Open Source. Muitas vezes usa softwares Open Source sem saber...Neste estágio (estágio 1) não existe contribuição para a comunidade. A empresa é apenas usuária. Sua intenção é reduzir custos em relação às alternativas proprietárias.

O próximo estágio (2), que chamaremos de colaborador, a empresa começa a usufruir mais intensamente dos conceitos de Open Source, fazendo modificações e evoluções em softwares e devolvendo estas modificações para a comunidade. Mas ainda são esforços sem integração, feitos de forma ad-hoc. Não existe uma estratégia real de uso e exploração dos conceitos do Open Source.

Vamos ao estágio 3, onde já existe uma estratégia definida. A empresa assume um papel proativo no desenvolvimento de uma comunidade, inclusive alocando profissionais de sua equipe ao projeto (o software não é mais desenvolvido ou evoluído dentro de casa, mas de forma independente, com a comunidade estruturada em fundações, por exemplo) e incentivando o ecossistema em torno dos seus softwares.

E finalmente podemos passar ao estágio 4 onde a empresa participa ativamente de diversos projetos e Open Source é parte integrante e essencial da estratégia e modelo de negócios. Um exemplo de empresa neste estágio mais avançado? A IBM!

Bom, espero que o texto seja debatido e melhorado...É apenas uma proposição. Sugiro como fonte adicional de estudo sobre Open Source que vocês acessem o portal da comunidade Open Source do MIT (<http://opensource.mit.edu/>). A lista de papers disponíveis é excelente.

E um comentário final...Em quase todos os eventos vejo os mesmos gurus...Claro que eles tem muito a dizer, mas não seria interessante também vermos e ouvirmos representantes da Linux Foundation, Eclipse Foundation, Apache Software Foundation e de diversas outras comunidades de grande reputação? Afinal são estes projetos que estão fazendo o Open Source se disseminar!

Explorando o conceito de Open Source (Dezembro)

Outro dia, em um destes bate-papos para descontrair reuniões, surgiu uma conversa de como muitos cursos de graduação em ciência da computação (pelo menos no universo das universidades que o grupo em volta da mesa tem contato) não estão explorando adequadamente os conceitos de Open Source. E algumas idéias interessantes surgiram...

Por exemplo, um professor de disciplinas como banco de dados ou compiladores pode elaborar algumas aulas teóricas e depois pedir aos alunos, para que em grupo, façam download de algum destes softwares Open Source (existem inúmeros disponíveis na Web) e implementem um novo algoritmo, escrevendo o código e o testando no próprio compilador ou banco de dados. E o que é melhor, como tanto a versão original como a modificada pelos alunos estarão disponíveis, pode-se fazer benchmarks e comparar se os algoritmos desenvolvidos pelos alunos são realmente melhores.

E que tal outra sugestão gerada na conversa? Um professor chega para a turma e diz: “gente, neste semestre vamos escrever em conjunto um livro sobre redes Wi-Fi. Aqui estão os tópicos principais. Vou escrever a introdução e vocês, em grupo, vão escrever os demais capítulos”. No final do semestre, o que teremos? Provavelmente um bom esboço de livro, que poderá ser a base da turma do próximo semestre e assim sucessivamente.

E mais, sendo um livro digital (de preferência licenciado por Creative Commons) poderá incluir mecanismos de hipertextos, figuras, entrevistas em vídeo, podcasts, exemplos de código, etc, uma vez que não existe a limitação física do papel...Um livro digital nunca termina...sempre tem uma inovação, uma nova versão da tecnologia, uma nova entrevista, etc. Ideal para cursos de graduação em ciência da computação, pois faz com que os alunos usem muitos recursos tecnológicos inovadores e no final produzam um material que será útil a todos.

OK, e outra sugestão...Não precisamos ficar restritos a usar os conceitos de colaboração apenas aos cursos de graduação em ciência da computação. Outros cursos e mesmo escolas de primeiro e segundo grau poderiam desenvolver trabalhos que se materializariam em verbetes para a Wikipedia. Se as escolas e cursos de graduação pedissem aos seus alunos para escreverem, em grupo, verbetes para a Wikipedia, em poucos anos duplicaríamos ou triplicaríamos os atuais 344.000 verbetes que estão em português. Teríamos uma bela enciclopédia em português, gratuita e disponível a todos.

Enfim, existe um provérbio chinês que diz: “existem três coisas que não voltam atrás: a flecha lançada, a palavra proferida e a oportunidade perdida”. Adiciono uma quarta: “o email enviado”...Não devemos perder mais esta oportunidade.

Novas fronteiras para o Open Source (Dezembro)

O movimento Open Source, antes limitado ao stack mais básico de software (sistemas operacionais e middleware), começa a aparecer e ser aceito nas camadas mais altas, onde estão os pacotes aplicativos.

Já existem algumas soluções que, se por um lado, ainda estão longe de oferecer as extensas funcionalidades dos produtos proprietários, por outro, atendem a uma gama bastante ampla de usuários, que não precisam de todas estas funcionalidades. De maneira geral os usuários destas soluções são empresas de pequeno porte, que são muito suscetíveis a custos.

Algumas pesquisas tem mostrado que os principais fatores de decisão que levam a escolha das alternativas Open Source são o baixo custo de aquisição, não ficarem sujeitos ao “vendor lock-in”, alta qualidade do código e facilidade de modificar o produto para atender as necessidades específicas (pelo acesso ao código fonte). Mas demandam também alguns riscos inerentes a suporte, manutenção e atualizações. Atualizações são um bom exemplo: se o nível de modificações for muito elevado, o pacote acaba sendo inteiramente customizado, ficando fora das evoluções criadas pela comunidade. A empresa passa a ser a única responsável por sua evolução, gerando a mesma situação de um sistema criado internamente.

Os primeiros projetos Open Source de pacotes aplicativos surgiram no campo do CRM e ERP, mas agora vemos em outros setores como em ETL, BI e mesmo ESB.

Vamos dar uma rápida olhada neste cenário.

Começando com CRM, existem alguns produtos interessantes como o SugarCRM (www.sugarcrm.com), vtiger (<http://www.vtiger.com/>), Centraview (<http://www.centraview.com/>) e outros.

No campo dos ERP, já há algum tempo conhecemos alguns projetos Open Source como o Compiere (www.compiere.com), OpenBravo (www.openbravo.com) e outros, mas que até hoje estavam fora da tela do radar da imensa maioria dos CIOs. Mas no ultimo evento “CIO08 The Year Ahead Conference”, organizado pela respeitada publicação CIO Magazine (www.cio.com), foram apresentados diversos cases de implementações de ERP Open Source.

Não me parece que o mundo dos ERP vai se transformar em Open Source, mas com certeza veremos mais e mais cases de uso, principalmente em empresas de menor porte. Os ERP Open Source ainda não estão na lista prioritária dos CIOs, mas segundo pesquisas do Gartner Group, entre empresas que estão usando Open Source, pelo menos 12% já usavam ERP Open Source e outras 14% pretendiam implementar estas soluções em um prazo de até um ano. De qualquer maneira, os ERP Open Source já estão deixando de ser curiosidades tecnológicas para se tornarem alternativas a serem plenamente consideradas.

O Open Source também já começa se infiltrar no outrora fechado mundo dos produtos ETL

(Extract, Transform and Load). Algumas das alternativas que podem ser analisadas são Clover.ETL (<http://www.cloveretl.org/>), o KETL (<http://www.ketl.org/>), o Apatar (www.apatar.com) e o francês Talend (<http://www.talend.com/>).

No segmento de BI (Business Intelligence), também outrora domínio exclusivo do modelo proprietário já vemos algumas alternativas interessantes como o Pentaho (<http://www.pentaho.com/>). Os ESB (Enterprise Service Bus), coração do modelo SOA também já foram infiltrados por projetos Open Source, como o MuleSource (<http://www.mulesoft.com/>).

Bem, isto tudo significa que o mundo dos pacotes será todo Open Source? Na minha opinião, não! Veremos sim, as alternativas Open Source e proprietárias convivendo. Um exemplo? O projeto Open Source de uma comunidade criada em torno do conhecido Salesforce.com., para desenvolver uma API aberta e respectivos Web services, projeto que pode ser visto em (<http://sourceforge.net/projects/sforce-app-dl>). A IBM, por exemplo, é uma empresa que está implementando uma bem sucedida estratégia de sinergia entre os seus softwares comercializados no modelo proprietário e as alternativas Open Source. Basta ver os exemplos da sinergia entre os projetos da comunidade Apache a família WebSphere, entre a comunidade Eclipse e os produtos Rational e Lotus e assim por diante.

Um outro ponto importante: uma pesquisa ao SourceForge ([//sourceforge.net](http://sourceforge.net)) nos retorna dezenas de projetos Open Source CRM, ERP, ETL... e nem todos estes projetos vão dar certo. Muitos, na minha opinião, serão absorvidos por outros ou desaparecerão com o tempo, com sua comunidade se dispersando para colaborar em outros projetos que lhes sejam mais atrativos. Portanto, para selecionar uma alternativa Open Source não esqueçam de avaliar cuidadosamente o ecossistema em torno do projeto (tamanho e atividade da comunidade, referências na Web, livros, treinamentos e consultores disponíveis, etc).

Mas, com certeza as alternativas Open Source vão ter seu espaço no mercado. Vale a pena prestar mais atenção a eles!

2008

Em 2008 o modelo Open Source já estava bem disseminado e muitos governos começaram a incentivar com mais ênfase sua utilização. Um survey das iniciativas governamentais em curso na época pode ser visto no relatório Government Open Source Policies, do Center for Strategic and International Studies, acessível em http://csis.org/files/media/csis/pubs/0807218_government_opensource_policies.pdf.

As ações dos governos, geralmente os maiores consumidores de TI e portanto influenciadores da indústria, incluíam preferências na aquisição de softwares, incentivos fiscais e financiamento para empresas que desenvolvessem softwares Open Source. Alguns exemplos internacionais forma Singapura que concedia descontos em impostos para empresas que adotassem Linux e Israel que oferecia financiamentos de até 100.000 US\$ para empresas start-up que desenvolvessem seu código em Open Source.

2008s também foi um ano bastante intenso com relação aos padrões de documentos. Em maio a ABNT aprovou oficialmente o ODF, como norma NBR ISO/IEC 26300:2008. Em agosto foi assinado o Protocolo de Brasília, que formalizou a clara intenção de órgãos públicos brasileiros na adoção de formatos abertos de documentos, na troca de documentos eletrônicos entre seus signatários, cuja lista contempla instituições de porte significativo.

Licenças Open Source: qual a melhor? (Fevereiro)

Esta semana estive almoçando com um amigo, que tem uma pequena empresa de software e está propenso a licenciar seu produto como Open Source. Portanto, a conversa derivou para modelos de licenciamento e acho que vale a pena colocar os pontos principais da conversa aqui.

Meu ponto de vista sobre o assunto é o seguinte...Sem receita, seja direta ou indireta, nenhum projeto de software consegue prosperar. Apesar do software ser intangível, nós (pessoas) precisamos comer coisas sólidas e beber coisas líquidas, que são tangíveis...E as empresas precisam manter estas pessoas e serem cada vez mais lucrativas e sólidas. Portanto, temos que pensar em modelos econômicos que dêem sustentação aos negócios de softwares.

Para mim Open Source é um modelo de desenvolvimento de software e que pode gerar modelos de negócio bem diferentes dos modelos tradicionais, baseados em venda de licenças de uso do software. E minha tese é a seguinte: você deve primeiro definir seus modelos de negócio e de receita, e só então escolher o modelo de licenciamento Open Source.

Bem, mas ainda temos um probleminha...Existem muitas licenças de Open Source. Se vocês forem ao site da OSI (Open Source Initiative) e acessarem os URL <http://www.opensource.org/licenses/alphabetical> ou <http://www.opensource.org/licenses/category> verão uma extensa lista de licenças. E escolher a que melhor se adeque não é uma tarefa muito simples...

Vamos fatiar o elefante...Podemos colocar as licenças em uma linha imaginária, onde em um extremo está a licença GPL, mais radical no sentido que obriga a todo e qualquer arquivo de código que esteja ligado a um arquivo GPL a também se tornar GPL. Muitos a chamam de licença virótica por causa deste efeito “contagioso”. Esta licença não permite muita flexibilidade em modelos de negócio, restringindo a receita a basicamente serviços, como consultoria, educação e empacotamento do software. Dá dinheiro? Claro, existem empresas que ganham dinheiro nestas atividades, como as distribuidoras Linux.

Podemos, agora, colocar no outro extremo as licenças BSD (Berkeley Software Distribution), que nos permite pegar o código fonte e criar um produto proprietário. Não existe inclusive obrigatoriedade de retornar as modificações (ou trabalhos derivados) à comunidade. Um exemplo é a licença do Apache (<http://www.apache.org/licenses/LICENSE-2.0>).

E, existem as licenças mais ponderadas, que ficam no meio. Estas licenças obrigam que o arquivo que contém código licenciado mantenha a mesma licença, mas arquivos que não contém o código podem ser licenciados de outras formas. Ou seja, o código com esta licença não “contamina” os demais códigos. Um exemplo desta licença é a MPL (Mozilla Public Licence, que pode ser vista em <http://www.mozilla.org/MPL/>). Na minha opinião estas licenças são flexíveis o suficiente para permitir a criação de diversos modelos de negócio e ao mesmo tempo estimular a contribuição da comunidade.

Qual a melhor? A resposta é simples: depende do modelo de negócios que você quer adotar! Ah, e se tiver dúvidas (e terá), consulte um advogado...

Mudando as regras no desenvolvimento de sistemas (Fevereiro)

É indiscutível que o modelo Open Source já passou da fase de curiosidade tecnológica. Já faz parte do mundo real de TI.

Mas, se avaliarmos o grau de evolução e maturidade do Open Source nas empresas, podemos criar três grandes patamares. O primeiro, na qual imensa maioria das empresas se encontra é a de simples usuário das ferramentas Open Source. O segundo patamar, onde pouquíssimas empresas estão é o de modificar o código fonte e contribuir com a comunidade. São casos raros no Brasil. Eu, pelo menos não conheço nenhum webmaster que tenha feito modificações no código do Apache para implementar em sua empresa. Claro que devem existir, mas eu não conheço estes casos. Até queria conhecer!

O terceiro patamar é o de desenvolver projetos baseados no modelo Open Source. Este é o patamar mais maduro e evoluído de adoção do Open Source e aqui no Brasil, a exceção de alguns raros casos em empresas de governo, ainda é inexistente.

Usar Open Source não demanda maiores mudanças nos processos da empresa. De maneira geral uma inovação técnica, como adotar um software Open Source é uma decisão que não envolve a gestão de TI da empresa. Muitos CIOs nem sabem quanto de Open Source tem dentro de casa...Tende a ser um processo bottom-up, com muitos CIOs simplesmente formalizando, a posteriori, o uso do Open Source na empresa.

Já modificar código fonte já demanda um comprometimento da gestão de TI, pois envolve maiores riscos. E desenvolver projetos em Open Source é sim, uma decisão gerencial, pois envolve mudanças significativas no processo de desenvolvimento de sistemas e gerenciamento de projetos.

E os benefícios com Open Source? Bem, o simples uso já traz bastante ganhos, como redução nos custos de licenciamento. Entretanto, pode-se conseguir grandes benefícios ao se adotar o modelo Open Source no desenvolvimento de projetos, seja abrindo o projeto à comunidade como um todo ou em consócio fechado e restrito a algumas empresas. Alguns ganhos que podem ser contabilizados quando o projeto é aberto: ritmo mais rápido de inovação (idéias vindas de fora), testes mais completos (a comunidade tem mais olhos que a equipe interna) e menor custo de desenvolvimento (divide-se o custo do projeto com outros). Claro que não são todos projetos que podem ser desenvolvidos de forma aberta, mas aqueles sistemas que não implicam em vantagem competitiva (todas as empresas fazem da mesma maneira...) poderiam ser construídos assim.

Existe um case interessante de desenvolvimento colaborativo que é a empresa americana CSI (Collaborative Software Initiative). O site é <http://www.csinitiative.com/>. A empresa começou quando Stuart Cohen, ex-CEO do Open Source Development Labs foi sondado por diversos bancos e seguradoras que pretendiam dividir os custos de desenvolvimento de aplicações não-competitivas, desenvolvendo-as em Open Source e de forma colaborativa, terceirizando as subsequentes atividades de suporte e manutenção. A iniciativa já recebeu um milhão de dólares da OVP Venture Partners.

Por que não adotar o modelo Open Source nos seus projetos internos? Creio que aos poucos veremos este movimento começar a acontecer, à medida que o modelo Open Source seja mais e mais conhecido.

FISL 9.0 e a maturidade do Linux e Open Source (Abril)

Esta semana, de 17 a 19 de abril de 2008, começa em Porto Alegre, o FISL 9.0. Infelizmente, sim, infelizmente mesmo, desta vez não poderei estar presente, por ter nas mesmas datas, algumas reuniões já agendadas. A IBM estará muito bem representada pelo pessoal do LTC (Linux Technology Center).

Conversando com alguns jornalistas sobre o evento, lembrei o fato do movimento Open Source e o Linux já estarem bem amadurecidos. Não são mais novidade, estranha às corporações, mas estão rapidamente se inserindo no mainstream corporativo.

Recente relatório publicado pelo IDC, chamado “The Role of Linux Servers and Commercial Workloads” mostra alguns números bem interessantes, como o que demonstra a solidez do ecossistema de negócios em torno do Linux (hardware, software e serviços) que totalizou 21 bilhões de dólares em 2007 e que deve crescer até 49 bilhões para 2011, principalmente pelo crescimento do Linux em servidores rodando aplicações comerciais como ERP, CRM, banco de dados e outras aplicações de negócio). O Linux não é mais apenas um nicho de workloads técnicos de infra-estrutura como file ou mail server.

O relatório do IDC aponta como principais oportunidades de serviços para este ecossistema as atividades de migração, integração e deployment. Educação e treinamento é visto como de menor oportunidade, mas geralmente é o ponto de entrada para os serviços de maior valor agregado como consultoria e integração. O IDC estima que estes serviços para Linux e Open Source deverão crescer cima da média do próprio mercado de serviços. Chama atenção para o fato que as receitas de software em torno das plataformas Linux já somam 10 bilhões de dólares, cerca de 4% do total da indústria, que é de 242 bilhões de dólares. Mas, estima que até 2011 deverá crescer para mais de 9%, representando 31 bilhões de um mercado que deverá se situar no patamar dos 330 bilhões de dólares.

A conclusão do relatório diz claramente “Linux continues to drive market shifts in the industry, and early adoption patterns have given way to mainstream deployment that includes a growing portion of business-oriented workloads including ERP, CRM and other line-of-business solutions. The phenomenon of Linux as being seen by business customers primarily as a low-cost infrastructure solution is being increasingly displaced by business deployment of Linux”. E mais ainda “The Linux ecosystem has strong long-term prospects, with the overall ecosystem spend projected to increase from \$21 billion in 2007 to \$49 billion in 2011. The shifts highlighted in this paper will help drive that trend forward at a healthy rate, as users increasingly use Linux as a key business solution for today’s IT challenges”.

A Linux Foundation também publicou um paper muito interessante, com números sobre o desenvolvimento do kernel, “How Fast is Going, Who is Doing It, What They are Doing, and Who is Sponsoring IT”.

Lendo vocês vão saber que o kernel já conta com quase nove milhões de linhas de código e a média de lançamento de releases se situa agora em torno dos 2,7 meses. São 2,83 patches

por hora. Analisando a atividade em torno do kernel chegamos a uma impressionante estatística que a cada dia são adicionadas 3621 novas linhas de código, 1150 são removidas e 1425 modificadas.

Quem faz este trabalho? Do 2.6.11 ao 2.6.24 foram 3678 desenvolvedores, sejam autônomos como de 271 companhias. Os dez contribuidores mais ativos contribuíram com quase 15% e os trinta mais, com cerca de 30%. A relação com seus nomes está no relatório.

Vemos também que entre as empresas contribuidoras, a Red Hat aparece em primeiro lugar no número de contribuições (11,2%), seguido pela Novell (8,9%) e IBM (8,3%). Sim, somos a terceira empresa em volume de contribuição com código para o kernel. Entre as demais vemos Intel (4,1%), Oracle (1,3%), Google (1,1%) e HP (0,9%). O maior volume de contribuições ao kernel, quase 70%, vieram de desenvolvedores de alguma forma pagos por empresas e cerca de 30%, mais precisamente, 26,8%, vieram de colaboradores “none” e “unknown”, que são colaborações individuais, sem patrocínio de alguma empresa ou que a empresa não pode ser identificada.

E, já vemos também colaborações de empresas que antes não imaginávamos que pudessem contribuir, como a Volkswagen, que contribuiu para o kernel 2.6.25 com a implementação do protocolo PF_CAN, para comunicações em ambientes que sofrem fortes interferências, como automóveis.

A conclusão disto tudo? Linux é sem sombra de dúvidas um excelente sistema operacional e um dos mais bem sucedidos exemplos de projetos Open Source. Sua comunidade é vibrante e demonstra de forma inequívoca que colaboradores individuais e empresas podem criar uma comunidade com ampla sinergia. Desenvolvimento colaborativo por excelência. Definitivamente, Linux não é obra de hobistas, mas de desenvolvedores competentes.

Em resumo, na minha opinião, apenas desinformados ainda tem medo de adotar Linux e Open Source!

Impacto Econômico do Open Source (1) (Abril)

Há pouco acabei de ler um abrangente relatório sobre o impacto econômico do Open Source na inovação e competitividade no setor de IT da União Européia. O Título é “Economic Impact of FLOSS on Innovation and Competitiveness of the EU ICT sector”. Vale a pena ler o documento com atenção. É um calhamaço de mais de 280 páginas, mas com um belíssimo conteúdo.

Bem, estive pensando bastante no assunto e gostaria de fazer alguns comentários sobre Open Source, principalmente aproveitando que o último FISL foi um grande sucesso. Open Source já está definitivamente afetando (aliás, já afetou) toda a indústria de software. Permite criar novos modelos de negócio e expande o próprio mercado de software, disponibilizando soluções a custos antes inatingíveis por muitas empresas.

Algumas empresas, como a IBM, foram pioneiras em entender e adotar este modelo de forma complementar e sinérgica, enquanto outras tentaram, em vão, lutar contra. Tem uma frase de Steve Ballmer da Microsoft que é emblemática desta visão antagônica, quando ele disse que “Linux é um câncer”, como vocês podem ler em http://www.theregister.co.uk/2001/06/02/ballmer_linux_is_a_cancer/. Mas, pelas últimas notícias, aparentemente até a Microsoft está se curvando ao Open Source...

Mas, o relatório me chamou atenção para alguns pontos que quero debater com vocês.

Primeiro, Open Source já não está mais confinado ao Linux, mas abrange uma gama imensa de produtos de software. E um dos maiores influenciadores de seu uso tem sido os governos. Relatório do CSIS (Center for Strategic and International Studies) mostra que em agosto de 2007 existiam 268 iniciativas governamentais de apoio ao uso de Open Source, sejam em discussão ou já aprovadas.

Agora, um fator de extrema importância e que muitas vezes não aparece nos debates sobre Open Source é o seu impacto econômico. O efeito econômico da indústria de software não pode ser medido simplesmente pelos empregos e impostos gerados diretamente. O software provoca um profundo impacto em praticamente todos os segmentos da economia. No nível mais básico, automatiza tarefas repetitivas que antes requeriam grande investimento de tempo e dinheiro. Com a automatização destas tarefas as empresas conseguem se concentrar no trabalho mais substantivo, ao invés do administrativo. A evolução tecnológica permite variar e sofisticar o trabalho, permitindo, por exemplo, que engenheiros utilizem recursos de “realidade virtual” para produzirem visões de automóveis, aeronaves e edifícios. A tecnologia da informação e o software são ferramentas essenciais para melhorias de eficiência das empresas e economias, aumentando sua produtividade e competitividade como um todo. A inovação tecnológica provocada pelo software pode afetar indústrias por inteiro, como o emblemático exemplo da indústria fonográfica que foi totalmente modificada pelo advento do P2P e da Internet.

A indústria de software se encaixa nas classificações pertinentes da economia de serviços. É invisível, embora seus efeitos sejam sentidos; é indivisível, pois não funciona em partes;

e é intangível, pois não pode ser tocado ou estocado. O que é estocado são as mídias. Além disso, existe a dificuldade em estabelecer um valor real a ele.

A dinâmica do mercado de software é acelerada. As barreiras de entrada são baixas. É um mercado orientado a conhecimento, com pouca necessidade de capital. Um empreendimento pode ser criado apenas se seus fundadores possuem conhecimento técnico suficiente. Não é em absoluto um mercado intensivo em capital. É um mercado de grande concorrência, com as empresas estabelecidas enfrentando contínuos riscos de novos entrantes.

As baixas barreiras de entrada e a pouca necessidade de capital impulsionam o ritmo de inovação e modificações. Produtos inovadores são lançados continuamente, bem como as versões dos produtos existentes são liberadas em curtos intervalos de tempo entre si.

O Open Source potencializa toda esta dinâmica. A esmagadora maioria das start-ups da Web 2.0 são construídas em cima de soluções Open Source. Sem o modelo Open Source seriam impossíveis conseguirem um volume de investimento tal que as permitisse lançarem-se ao mercado.

O relatório da União Européia mostra um dado interessante. Ele estima o custo de produção de alguns softwares Open Source e os compara com o hipotético desenvolvimento pelo tradicional modelo proprietário. Por exemplo, para dados de 2006, ele mostra que o OpenOffice teve um esforço de desenvolvimento de mais de 5 milhões de linhas de código, envolvendo mais de 79 mil homens-mês, a um custo estimado de 482 milhões de euros. O Mozilla (Firefox), com seus 2.430.000 LOC, demandou 25.000 homens-mês, a um custo de 154 milhões de euros. A questão é: sem Open Source estes softwares existiriam? Sem Open Source não teríamos hoje excelentes alternativas ao monopólio da Microsoft em suítes de escritório e navegadores. O mesmo quando falamos do Linux e seus mais de 9 milhões de LOC no kernel. Sem Open Source não teríamos um sistema operacional que fosse tão flexível que permitisse rodar desde celulares a supercomputadores.

O impacto econômico do modelo Open Source fica mais gritante quando comparamos o que seria o desenvolvimento de uma distribuição Debian (também com dados de 2006), com seus mais de 220 milhões de LOC, que se fosse desenvolvido por uma única empresa teria consumido 163.522 homens-ano a um custo estimado de 12 bilhões de euros (em 2005) e chegando, com as previsíveis evoluções e modernizações a 100 bilhões de euros em 2010. Alguns estudos mostram que 50% do código Open Source é completamente substituído em um período de cinco anos. Olhando apenas o esforço de codificação (excluindo debug, documentação, testes, etc), estriamos falando em 43.944 homens-ano correspondentes a 26000 desenvolvedores FTE (Full-Time equivalent, http://en.wikipedia.org/wiki/Full-time_equivalent).

Inimaginável pensar que existiria alguma empresa de software, principalmente uma start-up que se aventuraria no mercado para competir com softwares já dominantes, demandando este imenso volume de investimento. Sem Open Source o mundo do software seria outro, completamente dominado por monopólios.

Por outro lado o modelo Open Source abre perspectivas inovadoras de desenvolvimento colaborativo. O relatório mostra que a distribuição Debian analisada teve a contribuição de 986 empresas, que contribuíram com mais de 31 milhões de linhas de código, com esforço de 16.444 homens-ano, a um custo estimado de 1,2 bilhões de euros. Entre as empresas destacam-se a IBM, Sun Microsystems, RedHat, Novell e outras. Fica claro que Open Source é um movimento que agrega esforços da comunidade e de empresas.

Bem, o tema é apaixonante, e daria para escrever muitas e muitas páginas...E esta é a ameaça que faço a vocês...Vou escrevê-las sim, mas em posts separados por que ninguém aguenta ler textos extensos. Nos próximos dias vou continuar com “Open Source, a saga...”.

Impacto Econômico do Open Source (2) (Abril)

Continuando nosso estudo sobre os Impactos Econômicos do Open Source há um item que geralmente passa despercebido nos debates sobre este modelo: a questão da transferência de tecnologia.

Fazer parte de uma comunidade de Open Source e ter acesso a algoritmos sofisticados como os que estão embutidos em sistemas complexos como o Linux, equivale obter um nível de aprendizado superior ao que um estudante consegue em uma universidade média. Além disso, a constante troca de idéias e participação em fóruns de discussão amplia significativamente o escopo de conhecimentos.

Para os estudantes de ciência da computação, a participação em projetos Open Source abre skills não apenas de programação, mas também de teamwork, coordenação de projetos, compreensão de aspectos legais (regras e modelos de licenciamento) e melhor capacidade de comunicação em outras línguas.

Uma sugestão: as universidades de computação deveriam incentivar a participação de seus alunos em projetos Open Source. É um motivador e tanto ao velho e muitas vezes arcaico programa de aulas tradicionais. Aliás, as empresas também deveriam enfatizar que seus desenvolvedores participassem destes projetos. Estariam conseguindo para eles, de forma “grátis” uma maior capacitação, com enfoque muito mais prático que apenas frequentando cursos tradicionais.

Em termos globais, o movimento Open Source acaba sendo um duto de transferência de tecnologia dos países mais ricos (onde se concentram a maior parte dos colaboradores e empresas contribuintes) aos países menos desenvolvidos. É uma das formas benignas da globalização.

O Open Source permite também que países com menores recursos financeiros tenham acesso a softwares antes inatingíveis pelos seus altos preços. Imaginem um software para análises biológicas. Caso fosse adquirido um pacote específico, provavelmente muitas poucas universidades teriam acesso a ele. Mas, desenvolvido de forma colaborativa, com contribuição de empresas, universidades e voluntários de todo o mundo, uma parcela bem maior da sociedade teria acesso a esta tecnologia. Todos ganham!

Impacto Econômico do Open Source (3) (Abril)

Continuo com minha série de pensamentos sobre Open Source. Quero deixar claro que são idéias e propostas pessoais, não representando necessariamente a opinião da IBM e seus funcionários...

Open Source era visto nos seus primórdios (e isto há poucos anos...) como um exemplo típico da “gift economy”, onde o trabalho remunerado seria substituído por motivos mais sociais. É verdade que o espírito de colaboração é o cerne do movimento Open Source, mas em uma economia capitalista o dinheiro simplesmente não some. Ele se desloca de um lugar para outro.

Existem modelos de negócios construídos em cima do Open Source? Claro, desde as tradicionais ofertas de treinamento, consultoria e serviços de integração e distribuição, a outros mais dinâmicos e inovadores.

Por exemplo, é possível até fazer dinheiro com venda de licenças, como o exemplo do MySQL. Este modelo, “dual licence”, tem uma versão do software disponível sob GPL, sem custos e uma outra, licenciada, que permite outras empresas incorporarem o MySQL em seus produtos proprietários. Por que fazem isso? Ora, desenvolver a partir do zero um software de banco de dados é extremamente caro e a possibilidade de se embarcar um, já pronto e testado, é uma vantagem de tempo e dinheiro enorme. E por utilizarem uma licença não GPL, podem incorporar modificações específicas ao software que o tornam mais eficiente na interação com seu próprio produto e não são obrigadas a devolverem à comunidade tais modificações.

Claro que existem regras definidas para que este modelo funcione. A comunidade pode alterar a versão GPL e neste caso a MySQL não poderia exigir copyrights destas modificações. Como resolver a questão? A MySQL mantém seu próprio copro de desenvolvedores que são os únicos autorizados a incluir modificações na versão copyright. Também requisita que os contribuidores externos assinem contratos de copyright para modificações que considera úteis e adquire seus direitos de uso.

Começam a surgir outros modelos de negócio inovadores. Um exemplo é a ZEA Partners (www.zope-europe.org) que aglutina diversas pequenas empresas especializadas em suportar serviços de treinamento, suporte e desenvolvimento em Zope. Interessante que todo produto desenvolvido é disponibilizado em licenciamento permissivo (similar ao Apache Licencing), de modo que todos os demais membros ou mesmo outras empresas possam ter acesso. É o modelo colaborativo por excelência, pois os membros da rede ZEA também tem acesso a códigos gerado por empresas concorrentes.

O modelo Open Source abre também oportunidades para entrada em mercados de aplicativos antes inatingíveis pelos produtos comercializados por grandes empresas multinacionais. Um fabricante global de aplicativos como um ERP tende a se concentrar nos mercados de maior porte e rentabilidade, com menos investimentos em mercados menores, principalmente nos segmentos de empresas de pequeno porte, nos quais os custos

de localização nem sempre são compensadores. O resultado é que muitas vezes as inovações e atualizações tecnológicas aparecem com atraso nos mercados menores. Outro problema é a usabilidade (não apenas a tradução de telas), mas adaptação às características coloquiais de comunicação da língua nacional, que dificilmente são implementadas nos mercados menores.

Além disso, alguns tipos de aplicação demandam características totalmente diferentes. Um exemplo: aplicações estrangeiras de gestão hospitalar e clínicas não se adaptam facilmente às especificidades dos pequenos e carentes hospitais brasileiros. Atendem apenas (e muitas vezes parcialmente) aquela parcela mínima de hospitais classe primeiro mundo, que se contam em poucas dezenas. Outros exemplos? Gestão da administração escolar, gestão de consultórios dentários, e gestão de serviços públicos. Os processos dos órgãos públicos brasileiros são bem diferentes, inclusive nos aspectos culturais aos implementados em soluções estrangeiras. Além disso, surgem inúmeros espaços em setores onde pequenas empresas especializadas demandam aplicações específicas, como às encontradas em arranjos produtivos locais como têxtil ou calçados. Estes mercados não são atendidos de forma adequada por produtos estrangeiros (são mercados mínimos e de baixa rentabilidade).

Por outro lado, pequenas empresas de software nacionais não tem capacidade financeira, tecnológica e empresarial para atender a este mercado isoladamente. São poucas as empresas de software nacionais com capital suficiente para serem competitivas nacionalmente, como são a Totvs, Datasul e Senior. A solução é desenvolver aplicativos na forma colaborativa, Open Source, com contribuição de universidades que conhecem bem os processos dos setores de indústria e tem mão de obra tecnológica. De onde pode vir a receita destas redes de empresas? Que tal pensar em instalação, customização, integração e treinamento?

Esta rede social de negócios baseados em Open Source pode abrir novas oportunidades de mercado, hoje inexistentes. O software seria o mesmo, seja implementado em Rondônia ou Espírito Santo, a troca de expertise facilita o processo de implementação e as empresas concentram-se em fazer o aplicativo funcionar para atender ao negócio do cliente e não em escrever e depurar software.

O resultado final é um maior nível de informatização e eficiência das pequenas firmas nacionais (ainda bem limitada) e um maior e mais rico ecossistema para a indústria nacional de software. Lembrem-se. No mundo capitalista o dinheiro não some, apenas desloca-se de um lugar para o outro: da venda de licenças limitadas a um mercado relativamente restrito a um mercado bem maior de serviços e assinaturas. Será que não vale a pena pensar no assunto?

Impacto Econômico do Open Source (4) (Abril)

Este é o último post da série de opiniões pessoais sobre Open Source. E o tema que quero chamar atenção é inovação. No meu entender Open Source incentiva inovação e é um aspecto que nem sempre vejo comentado nas palestras sobre o assunto. O livre acesso ao código fonte é por si uma fonte de novas idéias. A razão é simples: software é uma tecnologia cumulativa, isto é, envolve inovação incremental e integração de várias invenções, algoritmos e técnicas. Com acesso ao fonte, podemos imaginar novas utilizações e mesmo criar novos algoritmos. E, claro, respeitando, as regras de licenciamento que estejam em vigor para o código fonte acessado.

Existe uma relação direta entre maior uso de software e aumento de produtividade empresarial e econômica. Assim, um maior incentivo à produção de software, através de modelos Open Source, que baixam tremendamente as barreiras de entrada (modelo colaborativo), aumentam as condições para criação de maior conhecimento e consequente maior produtividade em todos os setores da economia. E mais inovação implica em maior crescimento econômico.

Há pouco li um relatório que apontava que open source causaria prejuízos à indústria de software. Ora em um regime capitalista o dinheiro não some, ele se desloca de um ponto a outro do ecossistema. Talvez algumas empresas de software cujo modelo de negócios seja exclusivamente baseada em vendas de licenças saia perdendo, mas o software em si não. Vejam a tradicional indústria da música centrada nas grandes gravadoras. Ela está em crise, mas não a música, que está em plena efervescência cultural.

Ora, a chamada indústria de software representa apenas uma pequena parcela do total de software desenvolvido no mundo. Algumas estimativas apontam que 80% a 90% do valor do código gerado todo ano é feito por empresas (ou a seu pedido), para seu próprio uso, não estando inserida na economia da chamada indústria de software proprietário. Ora, adotar modelos colaborativos para estes softwares desenvolvidos internamente é um meio de acelerar a produtividade e o crescimento econômico das empresas.

Mesmo softwares desenvolvidos sob medida podem ser desenvolvidos por modelos colaborativos baseados em Open Source, seja por consórcios ou por grupos de empresas provedoras.

E quanto ao chamado software proprietário, aquele cujas licenças são comercializadas? Entendo que os modelos Open Source e Proprietários não sejam exclusivos e antagônicos. Existe espaço para ambos os modelos e acredito que muitas vezes mix destes modelos serão adotados pelo mercado. A razão é simples: existem conhecimentos específicos em um software (lembrem-se, software é conhecimento!) pelos quais o mercado reconhece valor e está disposto a pagar por este conhecimento. Estes são os softwares que serão comercializados, seja por licença seja como Software-as-a-Service. Agora, softwares que estão em fase de comoditização e com excesso de desempenho (funções muito acima das necessidades do mercado), não são mais reconhecidos como valor agregado pelo mercado,

que busca então alternativas mais baratas como as opções Open Source. Estes softwares, sim, serão canibalizados pelo modelo Open Source.

Portanto, devemos estudar o modelo Open Source pelo seu aspecto econômico e não apenas por questões ideológicas ou emocionais. Open Source, ao alavancar inovação vai acarretar, ao longo do tempo impactos positivos na economia como um todo.

Bem, tá legal...E a IBM, como se insere neste contexto? Há mais de dez anos atrás quando a IBM começou a falar deste assunto, para muita gente a surpresa era grande...A IBM envolvida com Linux e Open Source? Porque uma empresa que durante muitos anos foi considerada um bastião dos sistemas fechados (a era dos mainframes) entraria neste negócio, e ainda quando o movimento de Open Source era considerado um território inexplorado?

Em 2001, quando a IBM anunciou no Linux World um investimento de um bilhão de dólares em iniciativas em torno do Linux, o mercado compreendeu que a coisa era realmente séria. Um bilhão de dólares não é envolvimento, mas sim comprometimento!

Mas porque este comprometimento? Bem, entendemos que Open Source é uma inovação no processo de desenvolvimento de software, que permite construir novos e inovadores modelos de negócios. Open Source acelera a inovação. Tem uma frase muito interessante de Eric Raymond, em seu livro “The Cathedral and the Bazaar” que expressa bem esta idéia: “ I think Linus’s cleverest and most consequential hack was not the construction of the Linux kernel itself, but rather his invention of the Linux development model” .

O desenvolvimento colaborativo permite a construção de softwares fantásticos e abre novos mercados, antes inatingíveis. Com Open Source a indústria de software pode entrar no mercado da Cauda Longa (Long Tail), inacessível antes devido aos custos de produção, distribuição e comercialização de software pelos modelos tradicionais.

Os objetivos da IBM com Open Source são: aumentar o ritmo e velocidade das inovações, incentivando o “caldo cultural” de inovação nas comunidades (inteligência coletiva), ser um contribuidor ativo das comunidades e não apenas consumidor, capturar e transformar inovações Open Source em valor para nossos clientes e alavancar modelos de negócios baseados em Open Source para entrar em novos mercados e expandir oportunidades de negócio. É, portanto uma estratégia e não um simples oportunismo para explorar uma onda, ou uma reação atrasada e forçada pela pressão do mercado.

E o futuro? Que esperamos do Open Source? Bem, a estratégia da IBM com Open Source fica bem clara quando lemos um comentário de Sam Palmisano, gerente geral da IBM mundial, quando ele diz “Open Source is a method of tapping a community of experts to develop useful things. It began in software, but applies broadly, and is anything but anti-capitalist. It can raise quality at reduced costs, and vastly expands opportunities for profit. In a sense, open source fuels innovation much the way science fuels technology. Science is created by communities of experts, whose fundamental discoveries are typically made available to all, including individuals and companies that are able to capitalize on the new

knowledge in novel ways. For IBM, the open source model is familiar territory, given our long track record in the sciences”.

Código aberto: salvação ou suicídio? (Maio)

Logo depois de levantar o que seria o ultimo post de uma serie de observações e idéias sobre Open Source, estou no aeroporto esperando o vôo para o Rio quando vejo a última edição (versão brasileira) da Harvard Business Review, com um artigo que aborda exatamente o que vinha discutindo no blog...Muita coincidência para ser coincidência!

O artigo, um caso fictício que a HBR apresenta para debate tem o sugestivo título de “código aberto: salvação ou suicídio”, e descreve o dilema de uma empresa de games de grande sucesso que começa a enfrentar a onda do Open Source. Novas empresas, baseadas em open source surgiram, ameaçando o futuro, antes tão promissor para a fictícia empresa de games. E todas as novatas surgiram com código aberto...

O dilema da empresa é claro : “Devo aceitar a idéia do código aberto, liberar o acesso a minha propriedade intelectual a troco de nada?”. Tem uma frase no meio do texto que deve ser lida com atenção : “o código aberto é como uma grande onda...quem não sobe com ela se afoga”.

O texto termina com a questão para debate, “Martina, a presidenta da empresa, deve abrir o código do software de seu principal produto?”.

Quatro especialistas dão sua visão. Um deles, Jonathan Schwartz, presidente da Sun Microsystems afirma que “empresas fechadas precisam entender que, devido a suas decisões estratégicas,sua oportunidade de mercado é menor do que a de empresas abertas”. Ele compara a diferença nas vendas entre o iPhone da Apple, sistema fechado, com os da Nokia, que são aparelhos abertos a qualquer um que programe em Java.

Outro especialista, Eric Levin, da Techno Source, fabricante de brinquedos e jogos eletrônicos com sede em Hong Kong diz que é possível obter vantagens de ambas alternativas, com um meio termo, abrindo a plataforma para terceiros e adicionando recursos que estimulem a formação de uma comunidade.

Gary Pisano, professor da Harvard Business School lembra que antes de tomar a decisão, Martina precisa descobrir exatamente qual o poder de fogo de sua empresa na comunidade de código aberto. O software tem estrutura modular? Ela tem como criar e cultivar uma comunidade ativa?

E finalmente, temos a opinião de um advogado, Michael Bevilacqua, que aborda questões de violação de patentes.

O artigo merece ser lido por todos os que se interessam pelo estudo do movimento Open Source, e seus impactos econômicos. Na minha opinião pessoal, nenhuma empresa de software estará imune ao Open Source no longo prazo.

Open Innovation & Open Source (Junho)

Semana passada participei de um painel de debates em um seminário sobre Open Innovation, com o próprio Henry Chesbrough, autor do livro “Open Innovation: the new imperative for creating and profiting from technology”. Só para lembrar, o conceito de Open Innovation quebra o paradigma tradicional da P&D feito a portas fechadas, por uma única empresa. Este conceito trata a P&D como um sistema aberto onde tanto idéias externas e internas são debatidas e as melhores alternativas são selecionadas. No livro ele dedica um capítulo especial à IBM, onde descreve a transformação da empresa em sair do tradicional modelo de inovação fechado para o modelo Open Innovation. O livro é uma excelente fonte de referência para o assunto Open Innovation e deve ser lido por todos que estejam interessados em debater inovação em suas empresas.

A IBM indiscutivelmente é um case de sucesso na transformação para o modelo Open Innovation e vale a pena vocês lerem o capítulo que descreve o processo. Chesbrough diz que “IBM’s transformation demonstrates that even very large, very successful companies can learn new tricks.” E que “The Open Innovation approach requires IBM to focus on the value chain of its customers, rather than sticking with its traditional research heritage.” E também “Instead of reinventing wheels, IBM uses them to build new vehicles for its customers and makes money doing it.”.

Infelizmente, no evento, não pudemos debater em profundidade o assunto Open Source, mas quero discutir este tópico aqui no blog. O modelo Open Source é um exemplo claro do modelo de inovação aberta, pois envolve colaboração entre empresas, comunidade de desenvolvedores, clientes, etc. E este modelo tem impactado a indústria de software, obrigando a revisão dos antigos e tradicionais modelos de negócios, usados há pelo menos 25 a 30 anos.

O modelo Open Source difere do modelo tradicional em basicamente dois aspectos: o processo de desenvolvimento, que é essencialmente uma produção colaborativa e a filosofia de propriedade intelectual, que permite a livre distribuição do código fonte, bem como o direito de modificá-lo.

Open Innovation com Open Source pode se dar de várias formas. Podemos identificar o modelo de “Pooled R&D”, tipicamente representado pelos projetos Linux e Mozilla, onde diversas firmas (a IBM é uma delas) e a comunidade pesquisam e desenvolvem o software de forma colaborativa. Outro modelo é o Spinout representado pelo Eclipse. A IBM criou o projeto Eclipse doando código e propriedade intelectual e apoiando financeiramente a Eclipse Foundation.

A importância do Eclipse na indústria de software pode ser medido por alguns comentários publicados na mídia, como o InfoWorld de janeiro de 2006 que disse “..the Eclipse open source tools platform has come into its own, emerging as both an alternative to Microsoft...in the application development space and the de facto standard for developing Java”. Outros textos que valem a pena citar “Eclipse has won, what next for Eclipse”, dito pelo analista Carl Zetie do Forrester Research e “This year we find that six in ten

respondents use Eclipse as their primary IDE...”, publicado a partir de pesquisa com desenvolvedores, efetuada pelo Evans Data, em setembro de 2006.

Mas, porque o Eclipse faz tanto sucesso? Ele é um exemplo prático e bem sucedido do modelo de Open Innovation, onde empresas concorrentes podem criar redes de inovação, cooperando no desenvolvimento de softwares Open Source, que servirão de base para produtos específicos (proprietários), com os quais concorrerão no mercado. O Eclipse é aberto a todos, sob as mesmas regras. Não existem regras que excluam ou minimizem a participação de quaisquer contribuidores, mesmo que eles sejam concorrentes diretos entre si. Todas as discussões e deliberações dos projetos são transparentes a todos, abertos e públicos. O processo que governa os projetos Eclipse é meritocracia: quanto mais você contribui, mais responsabilidade pode obter.

O sucesso do Eclipse, é portanto, não apenas fruto de uma comunidade Open Source vibrante e atuante, mas também de um ecossistema saudável onde empresas de software (concorrentes ou não) trabalham juntos na criação de uma plataforma que servirá de base para seus produtos e serviços. Em resumo, as empresas cooperam na construção da plataforma e competem em produtos específicos, construídos em cima desta mesma plataforma.

Mas existem outros modelos de Open Source baseado em Open Innovation, como o modelo de venda de componentes, como exemplificado pelo Apache e o KDE, e o baseado em estratégias “dual licence” como MySQL.

O modelo Open Source é um belo exemplo de como empresas podem atuar em ecossistemas complexos combinando inovações internas e externas e provavelmente poderá servir de inspiração à adoção do modelo de Open Innovation por outros setores de negócios.

Schumpeter, Destruição Criativa e Open Source (julho)

Outro dia em um destes papo-cabeça estava debatendo com alguns colegas o impacto do Open Source na indústria de software. O modelo que usei para mostrar que Open Source está e estará transformando a indústria de software é a análise dos modelos econômicos feita por Joseph Schumpeter (destruição criativa). A sua teoria do ciclo econômico propõe que, para que a economia saia de um estado de equilíbrio e entre em um processo de expansão é o surgimento de alguma inovação, que do ponto de vista econômico, altere consideravelmente as condições existentes de equilíbrio no mercado.

Ele cita como exemplos de inovações que alteram o estado de equilíbrio a introdução de um novo produto no mercado, a descoberta de um novo modelo de produção e/ou de comercialização, e a alteração da estrutura de mercado vigente. Ora Open Source é o próprio processo de destruição criativa em ação: é um novo modelo de produção (colaborativo) e comercialização (explorando a Internet), possibilitando uma estrutura de custos zero de licenciamento. Com esta estrutura de custos tendendo a zero pode-se criar novos modelos de negócio, ampliando as opções e oportunidades de mercado. Open Source abre a possibilidade de exploração de mercados antes inatingíveis ou inexistentes. Um exemplo são as iniciativas Web 2.0, construídas em sua maioria, em cima de tecnologias Open Source. Na minha opinião, dificilmente veríamos tantas start-ups Web 2.0 se as tecnologias que as movem não fossem Open Source.

Assim, Open Source está desafiando o status quo da indústria de software. Sua aceitação pelo mercado já é um fato inconteste. As empresas produtoras de software não podem ignorar este fenômeno. Claro, podem reagir de forma contrária e lutar contra até o último momento, ou entender e explorar de forma positiva esta transformação. Este último caso é exemplificado pela IBM, que trabalha em colaboração com a comunidade há muito tempo. Aqui no blog já debati diversas vezes a estratégia Open Source da IBM (vejam os posts sobre o assunto pesquisando as tags ou categories Open Source).

Para finalizar, recomendo a leitura de um paper muito interessante, “The Transformation of Open Source Software”, acessado em <http://www.idi.ntnu.no/~ericm/brian.misq.pdf> que mostra o processo de evolução e profissionalização do modelo Open Source.

Ecosistema Open Source: amadurecendo rápido! (Agosto)

O ecossistema em torno do Open Source já é maduro o suficiente para impactar a indústria e os usuários de software. Existe um crescente número de soluções de negócio baseadas em Open Source entregando valor real para as empresas. As organizações já olham e implementam softwares Open Source sem os receios de alguns anos atrás. O momento ideológico e radical já ficou para trás, e a razão e não a emoção estão direcionando as estratégias de adoção de Open Source.

Portanto, o cenário atual do Open Source, se fizermos uma rápido sumário vai nos mostrar:

- a) Os ecossistemas Open Source e proprietário/fechado estão convergindo cada vez mais, cada um aproveitando e explorando as melhores idéias do outro. Vemos empresas criadas no modelo Open Source adotando modelos de negócio considerados tradicionais, bem como empresas de softwares proprietários adotando os conceitos e ofertando soluções Open Source em seu portfólio.
- b) Os ecossistemas Open Source já movimentam volumes significativos de dinheiro e está claro que o tempo das comunidades 100% voluntariadas já passou. As comunidades estão muito mais abertas à colaboração e contribuição das empresas de software.
- c) Já existe um consenso que ambos os modelos (Open Source e proprietário) vão coexistir, pelo menos em um horizonte previsível. E que um mundo só Open Source, que destruiria o valor da tradicional indústria de software, ainda está muito longe de acontecer.

A IBM, indiscutivelmente é um case de sucesso de adoção estratégica de Open Source. A IBM vem adotando Open Source de forma abrangente e pragmática, com forte integração com as comunidades. Uma frase de Jeff Smith, VP de Open Source and Linux Middleware da IBM é emblemática: “Open Source is important and fundamental for us. Our goal is to build advantages from Open Source into IBM’s core businesses, rather than to build something interesting on the side”.

E, portanto, nada mais natural que volta e meia sejamos consultados por parceiros ou empresas de software (ISVs) quanto a viabilidade deles adotarem Open Source em sua estratégia de negócios.

Coletei algumas dicas, um pequeno check-list de “como ir para Open Source”, que foi construído informalmente ao longo das várias interações que tive com os executivos destas empresas. Gostaria de compartilhá-las com vocês.

- a) Primeiro estude os conceitos Open Source. Compreenda os prós e contras, entenda como as comunidades funcionam (olhe por exemplo as comunidades Eclipse, Apache e Linux), estude as diversas alternativas de licenças e analise os projetos Open Source de sucesso (e veja também alguns que não deram certo...).
- b) Analise a competição das alternativas Open Source no seu segmento de atuação. O seu software já enfrenta concorrência do Open Source? Estas alternativas são bem sucedidas, com comunidades atuantes? O resultado da análise é mostrar que caso você adote Open Source, se seu produto terá condições de conquistar uma comunidade de colaboradores com

massa crítica o suficiente para que o modelo Open Source de desenvolvimento seja adotado.

c) Valide se seu modelo de negócio não conflita com Open Source e que mudanças serão necessárias. Lembre-se que Open Source não é um Business Model por si, mas uma inovação no processo de desenvolvimento, distribuição, marketing e comercialização. Verifique se a mudança em seu modelo de negócio irá gerar mais ou menos receitas...A idéia é simples: quão rapidamente suas novas receitas baseadas em modelos Open Source compensarão a perda da receita dos clientes atuais advindas com o modelo tradicional?

d) Caso sejam mantidas versões Open Source e proprietárias, defina claramente os limites de funcionalidades de cada uma. Não esqueça que a versão Open Source deve ser atrativa o suficiente para atrair usuários e comunidades de colaboradores. Especifique que funcionalidades premium serão reservadas para a versão proprietária (se existir) e qual o valor percebido desta funcionalidade para o usuário investir seu dinheiro na aquisição de sua licença de uso.

e) Escolha um nome e um logo adequado, que diferencie a versão Open da proprietária.

f) Escolha uma licença adequada ao seu modelo de negócios. Não invente novidades, mas adote uma já existente. E não esqueça de verificar se o código do seu produto não contém componentes que invalidem a licença escolhida...

g) Tenha certeza que o código fonte a ser liberado está em boas condições...É um código espaguete monolítico ou é modular e bem condificado? Está bem documentado? Um código fonte que ninguém consiga entender vai desestimular a colaboração. Se seu código não estiver adequado, esqueça!

h) Crie uma infraestrutura adequada para apoiar a comunidade, com um website, wiki, FAQ, forums de debates, roadmap, informações de contato, regras de uso e comportamento da comunidade, etc. Não confunda a comunidade integrando o site da versão Open Source com a proprietária...São projetos diferenciados. Publique o código fonte em um diretório bem conhecido como SourceForge.

i) Anuncie publicamente o seu projeto Open Source. Apesar da força da divulgação via Internet, reserve um budget para press releases, ações de mídia, etc.

j) Seja paciente. Os resultados nem sempre virão no dia seguinte. Não esqueça que seu software terá que se destacar na multidão, competindo muitas vezes pelos mesmos desenvolvedores que podem já estar envolvidos em outros projetos. Mantenha contante pressão para que a comunidade cresça e se torne atuante. Publicar o software e deixá-lo de lado é tornar o software órfão...

k) Ir para Open Source não é uma aventura de curta duração, mas um compromisso sério e permanente. Exige muito esforço e comprometimento. É uma decisão estratégica e não meramente comercial.

E uma recomendação final. Leiam o livro “Producing Open Source Software”, de Karl Fogel, disponível for free via PDF em [//producingoss.com](http://producingoss.com).

Como será a indústria de software em 2020? (Setembro)

Fim de semana chuvoso, um bom vinho na varanda e de repente vem um pensamento: Como será indústria de software daqui a uns dez ou doze anos? Como será a indústria de software em 2020? Bem, como depois de algumas taças de vinho as idéias fluem muito mais livremente, vou colocar algumas opiniões pessoais, que não necessariamente coincidem com as do meu empregador (a IBM) ou meus colegas...

Vejo dois movimentos que já estão transformando decisivamente esta indústria, o Open Source e o Software-as-a-Service (SaaS), que, na minha opinião, em 2020 serão os modelos dominantes.

O modelo Open Source afeta diretamente a cadeia de valor da indústria pois atua nas mais importantes variáveis que entram na composição dos seus preços como os custos de desenvolvimento (diluídos pelo trabalho colaborativo) e marketing/comercialização (via Internet). Oferecendo alternativas “good enough”, custos de propriedade mais competitivos (em alguns casos os custos de aquisição tendem a zero) e modelos de negócio mais flexíveis, o resultado gerado pelo Open Source é uma pressão maior nas margens, obrigando a muitos produtos terem seus preços sensivelmente reduzidos. Um exemplo típico tem sido a contínua redução de preços de pacotes como o Office. Ah, “good enough” significa uma solução tecnológica que não necessariamente tenha todas as funcionalidades de um produto líder de mercado, mas que contém as funcionalidades que atendem a uma imensa maioria de usuários.

Software-as-a-Service é outro modelo disruptivo. Sua proposição de valor é funcionalidade oferecida e não a “propriedade” do produto. A idéia básica é que você na verdade não quer uma máquina de lavar roupa, mas quer a roupa lavada. SaaS oferece isso. Você não necessita instalar um pacote de CRM ou ERP, mas precisa das suas funcionalidades. O cliente não adquire licença de uso, mas paga uma taxa mensal baseada no número de funcionários que acessem o serviço.

O mercado vem dando sinais de grande receptividade a este modelo. Algumas estimativas apontam que SaaS pode chegar a 25% ou 30% do mercado total de software já nos próximos 3 a 4 anos. Outra estimativa aponta que já em 2010 pelo menos 65% das empresas americanas terão pelo menos uma aplicação rodando no modelo SaaS. Como estamos falando de um horizonte de uns dez anos, podemos imaginar que um percentual bem significativo do mercado de software será baseado em SaaS e Open Source por volta de 2020.

O resultado final é que a indústria de software precisará ser reinventada. Porque comprar uma licença de uso de um caríssimo software se existir uma solução “good enough” mais barata e que não precisa ser instalada em suas máquinas? A questão é que atrás destas mudanças estão novos modelos de negócio que provavelmente não terão margens de lucro tão altas quanto hoje. Hoje por exemplo, segundo dados da McKinsey, o EBITDA das maiores empresas de software situa-se entre 25% a 30%, enquanto que as de SaaS ficam em torno dos 13%. Uma das causas pode ser a ainda pouca escala do negócio. Com o

crescimento da base instalada e maior adoção do modelo, a lucratividade pode aumentar, mas dificilmente chegará aos valores praticados pela indústria de software atualmente. A dificuldade maior vai aparecer para as empresas já estabelecidas, que precisam mudar seu modelo de negócios e provavelmente sua estrutura organizacional, de vendas e de custos. E recriar o ecossistema de parceiros...Ou sejam, existem barreiras culturais e organizacionais a serem vencidas!

O modelo de negócios SaaS é bem diferente do modelo de licenças tradicional. No modelo tradicional a lucratividade vem das taxas anuais de manutenção e não necessariamente da venda de novas licenças. Por exemplo, a manutenção respondeu, no ano passado, por 58% da renda e 74% da lucratividade da Oracle.

Já a lucratividade do negócio SaaS depende de três variáveis básicas, muito similares ao do setor de celulares: quanto custa atrair um novo cliente (custo de aquisição), quanto estes clientes renderão com suas assinaturas (ou a receita média por usuário ou ARPU, que significa Average Revenue Per User), e com que frequência os assinantes vão embora e precisam ser substituídos (taxa de rotatividade ou churn rate). As operadoras de celular conhecem bem este jogo...

A transição para o modelo SaaS não é simples. Os custos de vendas e marketing ainda são muito altos. Um exemplo é que a empresa SaaS mais bem sucedida até o momento, a Salesforce.com gasta metade de suas receitas em vendas e marketing. E como o modelo ainda é novidade, a maioria dos clientes ainda está testando o serviço pela primeira vez e não existem garantias que ficarão muito tempo. No modelo tradicional a troca de um software é mais complexa e o aprisionamento do usuário é quase uma regra da indústria. Quantos usuários de ERP trocam de fornecedor? No SaaS a barreira de saída é muito mais baixa. Você poderá trocar muito mais facilmente de fornecedor.

A consequência é uma competição mais acirrada e preços menores. Resultado final: margens e lucratividades menores. Definitivamente que em 2020 a indústria de software deverá ter uma “cara” bem diferente da atual e as empresas lucrativas de hoje (como as fornecedoras de ERP) provavelmente estarão ganhando dinheiro com outros modelos de negócio (mais focados em serviços de consultoria e integração) ou simplesmente estarão fora do jogo. Ignorar e não reagir rápido à esta mudança de paradigma pode significar sair do mercado. Um exemplo é a Siebel que ignorou a entrada da salesforce.com e acabou sendo adquirida pela Oracle.

Latinoware: um evento fantástico! (Novembro)

Semana passada aconteceu o Latinoware, em Foz do Iguaçu. Nunca tinha ido a este evento antes e para mim foi uma grata surpresa. Muito bem organizado e com palestras e palestrantes de alto nível. Foi realmente um evento que me marcou e pretendo voltar outras vezes. E deixo aqui os parabéns aos organizadores.

Minha palestra debateu o Open Source como um exemplo prático do modelo Open Innovation. Este modelo tem sido muito debatido e um autor que vem falando muito sobre ele é Henry Chesbrough, que escreveu um livro que li duas vezes, e que recomendo enfaticamente, que é “Open Innovation: the new imperative for creating and profiting from technology”.

Mas, na apresentação mostrei alguns dados interessantes sobre o impacto econômico do Open Source. Recomendo a leitura do documento “Economic Impact of FLOSS on innovation and competitiveness of the EU ICT sector”. É um calhamaço de mais de 280 páginas, mas que todo interessado em analisar Open Source sob o ponto de vista de seu impacto na economia e na indústria de software deve ler. Open Source é um movimento que deve ser estudado profundamente. Não pode ser considerado apenas de forma simplista, como um movimento ideológico de hackers. Tem profundos impactos na indústria de software. Aliás, pode e já está transformando toda a indústria de software.

Alguns dados extraídos deste relatório, que mostra de forma clara o impacto do Open Source na economia e indústria de software na Europa e no mundo:

- a) Em market share está claro que Open Source está se disseminando muito rapidamente pelo mundo todo. Em alguns setores é líder ou está entre os três primeiros softwares em termos de utilização, como sistemas operacionais (Linux), web servers (Apache), Browsers (Firefox), databases (MySQL e PostgreSQL), suites de escritório (OpenOffice), linguagens de programação (Perl, PHP...).
- b) Europa e EUA são as regiões que concentram a maior parte dos desenvolvedores. Provavelmente Ásia e América Latina enfrentam o problema da língua, pois a maioria das comunidades dos softwares mais importantes dialoga em inglês. Na Ásia, Índia e China se destacam. O Brasil aparece com destaque no uso do Open Source em órgãos governamentais.
- c) Falando em impactos econômicos, o estudo da base instalada de Open Source teria custado às empresas, se fosse desenvolvido nos moldes tradicionais, cerca de 12 bilhões de euros. Este código está dobrando a cada 18/24 meses.
- d) Estima-se que o esforço de desenvolvimento e manutenção do Open Source equivale a uma força tarefa de 131.000 profissionais/ano. O estudo avalia esta contribuição da comunidade em 800 milhões de euros por ano.
- e) O estudo também estima que em 2010, o ecossistema Open Source vai alcançar cerca de 4% do PIB europeu!
- f) O relatório aponta como um efeito positivo do Open Source na Europa, o potencial de criação de pequenas empresas de software e serviços, principalmente porque na UE não existe tanto VC (venture capital) disponível, nem uma cultura de risco como existe nos

EUA. O Open Source está inserido na Estratégia de Lisboa (Agenda de Lisboa) como um dos alicerces para tornar a UE a “most competitive knowledge economy by 2010”.

g) O estudo também mostra o potencial do Open Source em incentivar inovação (destruição criativa), apontando o fato que a indústria de software tradicional tende muitas vezes a inovar de forma incremental, não disruptiva. Estima que pelo menos 36% de investimentos em P&D de software poderiam ser economizados com sinergia entre indústria e comunidade.

Outra leitura que recomendo e cujo conteúdo usei na palestra, e nos debates e conversas de corredor, foi o livro free, disponível em pdf, chamdo “Producing Open Source Software”, de Karl Fogel. O livro pode ser acessado em <http://producingoss.com/>.

São quase 180 páginas, mas que mostram de forma clara como criar um projeto bem sucedido de Open Source. Criar um projeto Open Source não é apenas abrir código fonte. Muitos fracassam. Por curiosidade, existe um site que mostra os projetos Open Source órfãos, que precisam de mantenedores. Vejam em http://www.unmaintained-free-software.org/wiki/Main_Page.

O livro debate alguns pontos muito importantes que muitas vezes não são levados em consideração quando alguém ou uma empresa inicia seu projeto Open Source. Por exemplo, ele aborda a questão da escolha do modelo de licenciamento, a infra-estrutura tecnológica necessária, a criação e manutenção de uma comunidade ativa e interessada, os modelos de negócios que podem ser desenvolvidos, o esquema de comunicação e divulgação, o modelo de empacotamento e distribuição, e assim por diante. Leitura obrigatória para quem estiver realmente interessado em criar projetos Open Source.

Conhecendo o processo de desenvolvimento do kernel do Linux (Novembro)

Na semana passada li um interessante paper escrito por Jonathan Corbet, “How to participate in the Linux Community”, disponível no site da Linux Foundation em <http://ldn.linuxfoundation.org/book/how-participate-linux-community>.

O documento descreve como funciona o processo de desenvolvimento do kernel do Linux e é uma excelente fonte de referência para quem estiver interessado em colaborar com a comunidade ou mesmo conhecer como funciona por dentro o modelo de governança de um bem sucedido projeto de Open Source.

O kernel do Linux tem mais de seis milhões de linhas de código e aglutina mais de 1000 contribuidores ativos. É, sem sombra de dúvidas, um dos maiores projetos de Open Source do mundo. O seu sucesso tem atraído mais e mais colaboradores, a maioria, inclusive pertencendo a empresas que querem participar do projeto. A questão é como entrar e ser um participante ativo da comunidade?

Cada comunidade Open Source tem suas próprias regras de conduta e modelos de governança. Entender como funciona a comunidade que desenvolve o kernel é essencial para ser um colaborador e também abre idéias para aprendermos como funciona um projeto Open Source.

O documento é dividido em oito seções e não é extenso: são 24 páginas, que são lidas bem rápido...Recomendo enfaticamente a todos interessados em entender o que é Open Source a lê-lo.

A primeira seção mostra a importância de se inserir todo código na mainline de códigos do kernel, para evitar-se a proliferação de forks e futuras “bombas relógio”. Por exemplo, um código inserido no mainline do kernel passa a ser visto, mantido e refinado por toda a comunidade, resultando em código de alta qualidade. Quando o código é mantido separado, não só não recebe contribuições externas, como corre o risco de funcionalidade similar ser incorporado por outro desenvolvedor ao mainline, fazendo com que o código isolado fique cada vez mais difícil e custoso de ser mantido. Este alerta é feito porque muitas empresas envolvidas em projetos de software embarcado tendem a acreditar que seu código, pela sua especificidade, deve ser mantido de forma isolada da árvore principal do kernel, o que é um grande erro. A seção também descreve as razões porque toda contribuição deve ser “assinada”, não se aceitando colaborações anônimas. É necessário para se evitar eventuais problemas de propriedade intelectual.

A seção dois é bem interessante, pois descreve como funciona o processo de desenvolvimento do kernel e a nomenclatura usada para designar as diversas versões. Explica também como é criado um release estável e o ciclo de vida dos patches. Descreve como um patch é inserido no kernel e cita alguns dados curiosos. Por exemplo, O responsável pela decisão de inserir ou não um patch no kernel é de Linus Torvalds. Mas quando se verifica que o kernel 2.6.25 teve mais de 12.000 patches, fica claro que é impossível uma única pessoa analisar e decidir sózinha. Linus, por exemplo, foi o

responsavel pela escolha de 250 patches ou cerca de 2% deste total. Os demais ficaram a cargo dos seus lugares-tenente, figuras que representam Linus na seleção dos patches em subsistemas, como gerenciamento de memória, drivers, networking, etc. A seção também descreve as regras de etiqueta da comunidade na troca de mensagens e uso das mailing lists.

Um item importante da seção é o que mostra como “getting started with kernel development”, ou seja como faço para começar a colaborar? Tem uma frase de Andrew Morton que é um excelente conselho: “The #1 project for all kernel beginners should surely be “make sure that the kernel runs perfectly at all times on all machines which you can lay your hands on”. Usually the way to do this is to work with others on getting things fixed up (this can require persistence!) but that’s fine. It’s a part of kernel development.”.

Outras seções descrevem o processo de planejamento da evolução do kernel (quem disse que Open Source não tem planejamento?), debatendo as etapas de avaliação, as discussões com a comunidade para definir o que e como será feito, os estilos de codificação (que garantem a qualidade do código), as ferramentas de depuração usadas, a documentação que deve ser gerada e assim por diante.

A seção seis mostra como deve ser feito o trabalho em colaboração. Cada patch é revisado por desenvolvedores que você não conhece, o que muitas vezes pode gerar algum desconforto. Existem algumas recomendações de como se portar nestas situações meio desagradáveis e de como evitar discussões desgastantes. Trabalhar em colaboração é um exercício que muitos desenvolvedores não estão acostumados. Para muita gente, ter um estranho revendo seu código e expondo as falhas em público, é algo no mínimo desconfortável...

Mas, enfim, a leitura deste documento é obrigatória para todos que estejam realmente interessados em conhecer mais a fundo o que é o movimento Open Source, e suas características de desenvolvimento colaborativo. Vale a pena este investimento de tempo!

Software Livre na Aeronáutica (Dezembro)

Na semana passada estive em Brasília participando do I Forum de Software Livre da Aeronáutica. Foi um evento muito legal, onde pude ver como as Forças Armadas estão investindo em Open Source. E meus parabéns à comissão organizadora, e ao major aviador Bernardo pelo belo resultado alcançado.

Como todo evento, as conversas de intervalo e cafézinho são muito instigantes e este evento não foi exceção. Uma das conversas foi sobre a sinergia entre cloud computing e Open Source. No meu entender, ambos tem uma relação simbiótica, onde cada um incrementa o crescimento do outro, e juntos tem o poder de provocar uma verdadeira disrupção na indústria de TI. Por que tenho esta opinião? Bem, Open Source está na base de alguns dos mais conhecidos exemplos de computação em nuvem como os do Google, salesforce e Amazon.

Qual a conexão entre cloud e Open Source? Ambos alteram de forma fundamental a economia da indústria de TI, transformando o modelo de negócios de venda para serviços. É uma mudança significativa. Vamos, por exemplo, olhar o software. De maneira geral, no modelo tradicional o software é “comprado” por uma licença de uso. “Comprado” pois não há transferência de propriedade, mas o usuário obtém o direito de seu uso pelo pagamento de um fee. A licença governa os termos e condições deste uso. E estes termos e condições são determinados pelo provedor do software. O foco das licenças de uso tende a ser as obrigações dos usuários...Já serviços são governados por acordos de nível de serviços (SLA, service level agreement) que definem claramente as obrigações que o fornecedor destes serviços aceita prestar, em troca de pagamento de um fee. Ao contrário das licenças de software, o foco do SLA são as obrigações do fornecedor.

Ora, um provedor de cloud para o mercado externo (o open cloud) não pode correr riscos de mudanças nos termos das licenças de software (imagine o nível de insatisfação gerado se ele for obrigado a repassar um eventual e inesperado aumento de custo do software de cloud para centenas de milhares de usuários...), e nem pode ficar refém do provedor de software, caso ele resolva também assumir o papel de provedor de cloud, competindo em melhores condições. O modelo Open Source minimiza estes riscos.

O evento também foi uma boa oportunidade para mostrar o que a IBM tem feito em Open Source. As primeiras iniciativas remontam a 1999 (lá se vão 9 anos!) e de lá para cá o comprometimento com Open Source tem crescido notavelmente. Estamos envolvidos em mais de 150 projetos e colaborando proativamente em iniciativas chave para a consolidação do movimento Open Source como Apache Software Foundation, Eclipse.org, Linux Foundation e Globus Alliance, entre outras. A IBM já doou um substancial volume de código e patentes para diversos projetos. Alguns exemplos? O banco de dados Cloudscape para o projeto Apache Derby e o VisualAge para o Eclipse. E recentemente código de storage management para o projeto Eclipse Aperi. O uso de Open Source também acelera a entrada em novos mercados, como por exemplo o “Project Zero”, de desenvolvimento ágil baseado em Web 2.0, integrando Dojo, Groovy e PHP. A IBM também é grande usuária de Open Source. Um exemplo é o projeto Big Green de consolidar seus sistemas internos, de

3900 servidores para 30 mainframes system Z, rodando basicamente Linux. É um percentual de consolidação de 130 x 1!

Diante deste quadro, é fácil prever que a IBM vai continuar investindo intensamente em Open Source.

E como estamos no final de 2008, uma pergunta que me fizeram foi como eu via a maturidade dos softwares Open Source e quais as perspectivas para 2009.

Bem, para classificar os softwares Open Source pensei, de forma um tanto simplificada, em três estágios: softwares maduros, com tecnologia já consolidada e amplamente aceitos pelo mercado, emergentes, com tecnologia ainda nas fases iniciais de evolução e pouco entendidos pelo mercado, e embrionários, ainda nos estágios experimentais.

Podemos pensar como alguns exemplos de softwares maduros o Linux, o Apache, o Eclipse, o MySQL e o OpenOffice. Como emergentes os softwares tipo ERP, CRM e ETL (que mantém base bastante pequena de usuários, ainda não tendo quebrado a barreira da desconfiança...), e como embrionários os softwares de BI. Bem, a lista deve ser preenchida por vocês...

E para 2009? Penso em uma maior adoção do Open Source, principalmente porque tempos de crise aceleram a procura por softwares mais econômicos. E esta maior adoção aumenta o interesse da comunidade em apoiar estes softwares, gerando um círculo virtuoso, com mais contribuidores aumentando as funcionalidades do software e este aumento de funcionalidades aumentando o número de usuários.

Como testemunhas convoco alguns analistas de indústria, como o Gartner que afirma: “By 2010, global 2000 IT organizations will use open-source products in 80% of infrastructure-focused software investments and 25% of business software investments. By 2010, 90% of global 2000 organizations will have formal open-source acquisition and management strategies”. E vai mais além: “By 2010, software companies that do not incorporate open source software (in some manner) into offered solutions, risk becoming uncompetitive because of the expense associated with in-house engineering. An open source software strategy could include community participation, licenses allowing freely available copying and distribution, subscription support, and commercial open source software”. O Gartner também afirma que em 2011 “open source software’s impact on enterprise application software will grow to US\$ 17 billion, with a five-year compound annual growth rate of 43%”.

2009/2010

Em 2009 e em 2010 o assunto Open Source já estava deixando as páginas da mídia para entrar na utilização prática. Na verdade existe uma relação diretamente inversa e proporcional entre aparições na mídia e uso de uma determinada tecnologia ou conceito pelas empresas. Quando o tema está quente na mídia ninguém está usando, a não ser uns poucos pioneiros. Quando sai da mídia é porque o mercado já adotou o modelo ou tecnologia e portanto já não desperta mais interesse jornalístico.

Open Source e o Profissional de TI (Janeiro)

O movimento Open Source vai se acelerar com a crise econômica. Budgets mais apertados e ênfase na redução de custos tornam mais atraentes as alternativas “low cost”.

Open Source (OS) já está maduro. Não é mais novidade e vemos soluções OS em todos os lugares. Na Internet, nos roteadores, nos data centers corporativos, em navegadores GPS, nos celulares...Estão na base da maioria das iniciativas Web 2.0 e redes sociais e nas linguagens de programação dinâmicas (PHP, Perl, etc). É a base tecnológica do Google, da Amazon, da Wikipedia...Seu impacto na indústria de software, nas empresas e na carreira profissional não deve e nem pode ser ignorado.

Diante deste cenário, o que podemos esperar do Open Source para os próximos anos? Bem, embora futurologia seja uma área de alto risco, podemos (e estas são minhas opiniões pessoais) com alto grau de certeza afirmar que:

- a) Open Source vai se tornar mainstream nas organizações. Será um padrão “de facto” em alguns segmentos como infraestrutura (sistema operacional, middleware, banco de dados...), ferramentas de desenvolvimento, computação de alto desempenho (hoje cerca de 85% dos supercomputadores da lista Top500 são baseados em Linux) e na maioria dos softwares embarcados.
- b) Os ecossistemas de software serão construídos em cima da sinergia entre modelos de negócio baseados em Open Source e proprietários. Praticamente todos produtos de software proprietários vão incorporar, em maior ou menor grau, código Open Source.
- c) Parcela significativa das atividades profissionais de TI estarão diretamente relacionados com Open Source. Arquitetos e engenheiros de software deverão ter competência no modelo Open Source de desenvolvimento de software. Vejam bem, não estou falando em ser simples usuário de Linux ou OpenOffice, mas de conhecer modelos de desenvolvimento colaborativos, criar e manter comunidades, etc. Contribuir para a comunidade e obter certificações que comprovem skill em Open Source serão consideradas um “plus” pelos empregadores e profissionais.
- d) Modelos de inovação abertos (Open Innovation) serão cada vez mais comuns e adotados pela maioria das empresas. Open Source é um exemplo prático e bem sucedido de Open Innovation. As empresas estarão atuando de forma mais pró-ativa com as comunidades, não

apenas usando software Open Source mas colaborando com código e capital intelectual. Desenvolver código Open Source será visto como um investimento em P&D.

e) Open Source será visto como um campo específico da educação em ciência da computação e novas disciplinas serão adotadas pela academia. O diferencial do Open Source é o processo de desenvolvimento colaborativo de software e não o uso dos sistemas Open Source...Open Source é basicamente colaboração e inteligência coletiva.

Precisamos estudar e entender os motivadores e os inibidores para criação de processos colaborativos e para isso é necessário entender o que é uma cultura de participação. Um ponto importantíssimo neste contexto é criação e motivação de comunidades. É fundamental para o sucesso de qualquer projeto Open Source que uma comunidade ativa e atuante seja mobilizada. A comunidade é o coração dos projetos Open Source. Se for vibrante e ativa, os projetos decolam e evoluem.

O processo de desenvolvimento colaborativo do software Open Source também merece um estudo mais aprofundado. Apresenta características diferentes dos modelos de desenvolvimento comumente adotados nos softwares proprietários. Portanto esta nova disciplina poderia ser algo como “Desenvolvimento em Open Source: práticas de engenharia de software”. E outra disciplina essencial: modelos de negócio. Sem um ecossistema saudável, os projetos Open Source não serão sustentáveis. Deve existir receita em algum lugar da cadeia de valor e como obter esta receita serão os principais tópicos desta disciplina.

Estamos vivenciando um momento interessante. Open Source está se consolidando como parte integrante do portfólio de software das empresas. Já não está mais restrito à periferia de TI, rodando apenas em servidores de impressão ou de rede, mas está inserido no core do negócio. As novas aplicações que utilizam conceitos mash-up, exploram redes sociais e a Web 2.0 serão intensamente baseadas em softwares Open Source. E estas aplicações serão cada vez mais críticas ao sucesso dos negócios. A conclusão então é simples: ignorar Open Source não será nada saudável para a nossa carreira profissional.

Open Source no ensino de computação (Fevereiro)

Outro dia, aproveitando uma pedalada noturna na ciclovia da praia de Ipanema, fiquei imaginando como o modelo Open Source poderia ser adotado na academia. Na minha opinião, o modelo de ensino tradicional adotado hoje pelas universidades assume um processo de aprendizado formal, geralmente individualizado e centrado no professor (teacher-centric). Os alunos são medidos por testes e fazem, ao fim do ano o famoso TCC (trabalho de conclusão do curso) de forma individual. Colaboração e team-work não são valorizados nestes testes! São chamados de cola e combatidos...E na maioria das vezes os testes não colocam o problema no contexto, tendem a ser teóricos e fora da realidade do dia a dia do futuro profissional. Para mim este modelo deve ser repensado!

Mas, não é só. Existe outro grande desafio no ensino de disciplinas de computação. A evolução tecnológica e a demanda de conhecimento evolui muito rapidamente. Imaginem um curso de graduação de quatro anos em ciência da computação. Metade do que o aluno aprende no primeiro ano estará obsoleto lá pelo terceiro ano. Os mecanismos de atualização dos cursos atuais ainda estão, em sua maioria, adaptados aos tempos pré-Internet e não conseguem acompanhar na velocidade adequada a evolução tecnológica.

Olhemos agora as comunidades Open Source. Elas são orgânicas ou sejam, auto organizadas, e incentivam processos informais de aprendizado em grupo. Podemos classificá-las como “learner-centric”. Na minha opinião, os projetos Open Source são um belo exemplo de ecossistemas de aprendizado, pois as comunidades Open Source conseguem prover e distribuir, de forma sustentável, o conhecimento necessário para a produção de software de boa qualidade. Além disso, do ponto de vista do aprendizado, o compartilhamento de informações transcende os skills de programação, havendo extensa troca de idéias entre os membros das comunidades em assuntos diversos como patentes, licenças, habilidades gerenciais e principalmente trabalho em equipe.

Se analisarmos suas principais características, veremos que muitas delas são muito desejáveis para um processo de ensino mais prático em computação.

Querem alguns exemplos?

- a) Conteúdo gerado pelo usuário. Porque os estudantes não podem contribuir proativamente para a criação e evolução do material do curso, através de wikis, código fonte, blogs, etc? O engajamento ativo dos estudantes aumenta sua motivação e abre perspectivas inovadoras para o conteúdo do curso.
- b) Atividade real. Fazerem os alunos contribuírem com código real para uma comunidade Open Source existente ou a ser criada pelos próprios alunos, é um trabalho útil e uma experiência profissional sem preço. Eles passam a ter contato com outros profissionais e estudantes (do mundo inteiro) e aumentam sua percepção e prática do que é desenvolver software de forma colaborativa. O compartilhamento de informações com outros estudantes e profissionais é altamente benéfico.
- c) Uso intenso de tecnologias. Participar ativamente de uma comunidade Open Source significa usar tecnologias intensamente, que variam de wikis, listas de discussão e chats, até ferramentas de geração e gerenciamento de código. Uso prático e real.

Um aspecto importante que faço questão de enfatizar é o potencial de aprendizado prático que se tem quando se trabalha em projetos reais Open Source. Pelo que vejo e ouço, conversando com CIOs e mesmo dentro da IBM, é que existe uma defasagem grande entre o aprendizado na maioria dos cursos de computação e a necessidade do mercado. Muito do que o aluno aprende nas escolas não é crítico ao dia a dia nas empresas, faltando a eles um maior experiência prática.

Ok, mas como adotar o modelo Open Source? Talvez o primeiro passo seja fazer com que algumas disciplinas passem a demandar contato direto com projetos Open Source. Estas disciplinas, e aí podemos falar em inúmeras atividades, como desenvolvimento de programas, aprendizado em banco de dados, sistemas operacionais, etc, exigirão que os alunos contribuam, de forma colaborativa, para comunidades Open Source. E nem precisam ser atividades de geração de código. Podem ser contribuições para os wikis das comunidades e mesmo até contribuições para o Wikipedia.

Um ponto importante. À medida que a geração Y começar a entrar nas universidades, vão querer usar no seu processo de aprendizado o que já fazem de forma comum no seu dia a dia, como trabalhar em equipe, usar intensamente tecnologias Web 2.0 e assim por diante. Os mecanismos atuais de ensino estão defasados diante desta nova demanda.

Bem, as condições básicas para se implementar o modelo Open Source na academia existem. Existem milhares de projetos Open Source que necessitam de maior colaboração para evoluírem. Também pode ser criada um novo projeto, especificamente para determinada disciplina... Demanda por mais colaboradores existe. As tecnologias adotadas pelos projetos Open Source são, obviamente, Open Source, o que significa que não existe custo adicional de compra de licenças para as universidades. O resultado para o aprendizado dos alunos será altamente positiva e a experiência que eles obterão nestas disciplinas será bastante útil para sua vida profissional. Portanto, o que falta para isto acontecer?

Open Source, SaaS e Cloud Computing: mudando a indústria de TI (Março)

A semana passada foi bastante exaustiva, com quatro palestras diferentes, em 3 cidades deste imenso país: Rio, São Paulo e Recife. Em uma destas palestras, me fizeram uma pergunta interessante, que gerou um bom debate e gostaria de compartilhar minha opinião pessoal sobre o assunto com vocês. A pergunta foi “com a crescente popularização do modelo SaaS, como fica Open Source?”.

Na minha opinião, o modelo SaaS já está saindo do “se” para “como”, impulsionado até pela crise de crédito, quando as empresas procuram trocar capex (capital expenses) por opex (operating expenses).

Na prática, SaaS e Open Source compartilham o mesmo modelo econômico, de baixo custo de capital e custos operacionais variáveis. Isto gera sinergia entre ambos os modelos e um impulsiona o outro. Os mesmos argumentos que atraem os usuários para o Open Source são usados pelos provedores de softwares como serviços. Que argumentos são esses? Simplesmente não haver necessidade de aquisição prévia de licenças de uso antes de usar o software. No SaaS você paga pelo que consumiu de recursos. No Open Source, o software também é visto como serviços e as receitas das empresas envolvidas neste setor são obtidas por serviços prestados, como por exemplo, empacotamento e distribuição de um conjunto de softwares, como uma distribuição Linux.

A computação em nuvem também será um acelerador do Open Source. A combinação de uma infra-estrutura “pay-per-use” associado com uso de softwares abertos vai reduzir significativamente as necessidades de capital e os custos de desenvolvimento de aplicações, e acelerar o time to market. É um cenário que vai permitir às pequenas e médias empresas entrarem mais rapidamente no mundo da Tecnologia da Informação. Portanto, para mim, acho que Open Source, SaaS e Cloud Computing vão criar um interrelacionamento e gerar sinergias, um impulsionando o outro. O resultado final será um outro modelo computacional, que vai mudar em muito o atual cenário da indústria de TI.

Após o debate, aproveitei para reler alguns pedaços do livro “A Cauda Longa” de Chris Anderson e comecei a pensar na relação deste conceito com a atual transformação da indústria de TI, com o crescente interesse pelo Open Source, SaaS e Cloud Computing.

O conceito de Cauda Longa propõe que determinados negócios podem obter uma parcela significativa de sua receita pela venda cumulativa de grande número de itens, cada um dos quais vendidos em pequenas quantidades. Isto é possível porque a Internet abre oportunidades de acesso que antes não existiam. É um modelo diferente do mercado de massa, onde poucos artigos são vendidos em quantidades muito grandes. Na indústria de livros, música e de mídia faz todo o sentido. Por exemplo, a Amazon reporta que parcela significativa de sua receita vem de produtos da Cauda Longa que não estão disponíveis (e jamais estariam) nas livrarias tradicionais, limitadas pelos caros espaços físicos das lojas.

E como Open Source, SaaS e Cloud Computing vão afetar a indústria de software? Nestes modelos, o custo de capital é substituído por custos operacionais.

Softwares que tem seu projeto de desenvolvimento cerceado pelo pequeno tamanho do seu mercado potencial (seu custo de produção não gerava retorno financeiro suficiente) podem agora, se desenvolvidos em Open Source e operados em nuvens computacionais, entrar no mercado. Os custos de comercialização destes softwares também tendem a zero, pois não é necessário hordas de vendedores, mas simples downloads e marketing viral (blogs e outros meios de disseminação de informação). A receita dos desenvolvedores dos softwares Open Source será obtida pelo seu uso (pay-as-you-use), típico do modelo SaaS. A imensa maioria das empresas não vai investir tempo e dinheiro modificando código, a não ser quando absolutamente necessário.

Será muito mais pragmático e lucrativo para qualquer empresa pagar pelo uso de um software que esteja hospedado em uma nuvem computacional. Afinal, não queremos uma máquina de lavar e sim, a roupa lavada...

Temos, portanto, um vasto campo para explorar o mercado da Cauda Longa no software.

Então, isto tudo significa que o mercado de software tradicional, baseado em licenças vai morrer? Na minha opinião, não! Pelo menos no horizonte visível...Acredito que conviveremos em um contexto onde os modelos de vendas de licença e software como serviços vão compartilhar os palcos por algum tempo ainda...

Cinco anos de Open Source (Abril)

Em 2004 publiquei um livro sobre Open Source, chamado “Software Livre, Potencialidades e Modelos de Negócio”, editado pela Brasport. Ele foi fruto de estudos que desenvolvia na época, aqui na IBM. De lá para cá me envolvi mais e mais no assunto Open Source e fazendo uma retrospectiva destes cinco anos vi que muita coisa mudou, muita coisa evoluiu. Algumas previsões que havia escrito aconteceram e outras não. Mas, faz parte do jogo...

No mercado de TI as mudanças ocorrem por diversos fatores, que isolados não significam muita coisa, mas quando interrelacionados potencializam as mudanças. Olhar para trás e rever o que pensávamos que iria ocorrer e ver o que realmente ocorreu nos ajuda a entender o que provavelmente vai acontecer mais à frente.

Em 2004, algumas empresas clamavam aos céus que Open Source era “coisa do diabo”, acabaria com a própria indústria de software e assim por diante. Enfim, a indústria continua forte, mas em transição.

Bem, o que estamos vendo hoje? Open Source já está maduro. Não é mais novidade e os vemos em todos os lugares. Na Internet, nos roteadores, nos data centers corporativos, em navegadores GPS, nos celulares...Estão na base da maioria das iniciativas Web 2.0 e redes sociais e nas linguagens de programação dinâmicas (PHP, Perl, etc). É a base tecnológica do Google, da Amazon, da Wikipedia...Seu impacto na indústria de software, nas empresas e na carreira profissional não deve e nem pode ser ignorado.

Open Source, ao lado de outros movimentos como SaaS, está mudando os modelos de negócio da indústria de software. Os usuários já tem outra percepção do valor do Open Source, não existem mais dúvidas quando a sua viabilidade. Aliás, open source está embutido em muito dos softwares comercializados no mercado. O exemplo da IBM é emblemático: temos, entre outros casos, o Eclipse embutido em produtos da família Rational e o Apache embutido no WebSphere.

A comunidade também já entende que a sinergia com a indústria é extremamente positiva. A IBM, por exemplo, participa de centenas de projetos Open Source e mantém, inclusive, um laboratório específico para desenvolver código para o Linux, o Linux Technology Center (LTC). Está claro que indústria e comunidade podem e devem atuar de forma conjunta.

Alguns estudos já mostram que de 50% a 80% dos membros mais ativos das principais comunidade de Open Source atuam de forma direta ou indireta junto à empresas de software. O LTC, por exemplo, tem mais de 600 profissionais engajados em Linux.

Já está sendo bastante comum empresas de software colaborarem em projetos de Open Source ou mesmo abrirem projetos por si mesmos, geralmente adotando modelos de licenças dual-mode. Nesta modalidade existe uma versão free, aberta à comunidade e uma

outra, “premium”, de código fechado, com funcionalidades adicionais e níveis de suporte mais refinados.

Mas, deve ficar claro que o tradicional open source criado por desenvolvedores voluntários não vai acabar. A inovação e a criatividade da talentosa comunidade de desenvolvedores em todo o mundo vai continuar criando novos e inovadores softwares. Mas, ao invés de procurarem se isolar, vão aceitar e até mesmo buscar maiores sinergias com a indústria. Na minha opinião, veremos cada vez mais a comunidade e a indústria trabalhando em conjunto, cada um contribuindo com suas expertises. O resultado será uma indústria de software muito mais eficiente e com custos de propriedade menores.

Aliás, o modelo de desenvolvimento de software colaborativo, típico do Open Source, está sendo visto como o paradigma de desenvolvimento para os próximos anos. A IBM, recentemente liberou uma tecnologia orientada a este modelo, chamando de Jazz.

O Jazz, é fruto da experiência obtida com o Eclipse e seu modelo de desenvolvimento colaborativo e aberto. O Jazz Project, traz para o desenvolvimento de software proprietário a experiência de desenvolvimento aberto, participativo e colaborativo, típico dos projetos Open Source e bem exemplificado pelo Eclipse.

O projeto Jazz (www.jazz.net) é uma iniciativa conjunta da brand Rational e do IBM Research, e sua proposta é desenvolver uma plataforma que alavanque a criação de tecnologias que permitam criar ambientes de desenvolvimento colaborativo, integrando equipes dispersas geograficamente.

Existe um forte relacionamento entre o Eclipse e o Jazz, embora ambos tenham propostas distintas: o Eclipse tem como objetivo o aumento da produtividade individual do desenvolvedor, enquanto o Jazz busca otimizar a produtividade de equipes de desenvolvimento distribuídas. O primeiro produto gerado por este projeto é o Rational Team Concert, ferramenta projetada para suportar desenvolvimento colaborativo.

O projeto Jazz coloca em prática no processo de desenvolvimento de software de uma empresa os estilos e práticas participativas e colaborativas do Open Source. Estamos chamando este processo de “Collaborative Application Lifecycle Management”, que pode ser definido como “new paradigm for transforming the software lifecycle so that it is more collaborative, transparent and productive”.

Aliás, visitando o site jazz.net vocês terão acesso a uma seção Community, onde verão cases de projetos Open Source e acadêmicos construídos em cima da plataforma Jazz (Academic & Open Source Use), e onde está definido que “Rational Team Concert 1.0 is available for use free of charge to qualified open source and academic institutions”. Aqui uma sugestão para a academia brasileira: usar o Jazz e o Rational Team Concert em seus cursos de engenharia de software! Por que não começar a educar hoje os estudantes de engenharia de software nas tecnologias que estarão sendo demandadas amanhã?

Mas, continuando com as “memórias do Open Source”...Observo que a disseminação do modelo SaaS e computação em nuvem deve acelerar o uso do Open Source.

Aliás, já aparecem as primeiras experiências de computação em nuvem baseado em Open Source. Um é o projeto Eucalyptus, desenvolvido pela Universidade da Califórnia, em Santa Barbara, EUA. Este projeto, ainda acadêmico, emula o interface de computação em nuvem EC2 da Amazon e pode ser visto em <http://open.eucalyptus.com/>. Um outro projeto Open Source para computação em nuvem é o Enomalism (www.enomally.com). Este último permite que você construa um ambiente similar ao do Amazon, dentro de seu próprio data center. Existe também um toolkit chamado Nimbus (<http://workspace.globus.org/>), que permite transformar um cluster em um ambiente de computação em nuvem.

Os objetivos destes projetos são exploratórios, ou seja, permitir que pesquisadores trabalhem diretamente com tecnologias de computação em nuvem, coisa difícil de fazer quando a maioria das tecnologias são fechadas e proprietárias.

Existe também uma simbiose entre computação em nuvem e Open Source. Se visualizarmos um massivo data center com milhares de servidores comoditizados e um ambiente virtualizado, parece claro que o uso de softwares de código aberto permite reduzir mais ainda os custos operacionais e reduzir a dependência dos provedores de nuvem dos fornecedores de software. Open Source já é usado amplamente por várias das implementações de computação em nuvem, como o Linux (base das nuvens do Google, Salesforce e Amazon), o Xen (usado pela Amazon), o Hadoop e mesmo os pioneiros projetos de código aberto para computação em nuvem como os Eucalyptus e o Enomalism, que vimos acima, que usam Linux e Xen.

Além disso, me parece claro que o movimento Open Source vai se acelerar com a crise econômica. Budgets mais apertados e ênfase na redução de custos tornam mais atraentes as alternativas “low cost”.

O resultado final é que a simbiose entre computação em nuvem e Open Source vai mudar de forma significativa a indústria de software como um todo.

Open Source na maioria (Agosto)

Volta e meia dou entrevistas para a mídia falando de Open Source. E uma das perguntas mais frequentes é sobre “Quanto a IBM obtém de receita com Open Source?”. Ora, quando se fala no modelo tradicional de comercialização de softwares, esta pergunta tem uma resposta fácil: basta ver quantas licenças foram comercializadas e qual o preço médio delas. Mas, com Open Source é diferente. É muito difícil capturar com precisão o volume de receitas. Muito da receita de Open Source é obtido de forma indireta. Um exemplo é o Google que fornece gratuitamente softwares como Android e outros, para alavancar receita com anúncios. Vejamos também a IBM, que apoia diversos projetos como o Linux, Eclipse e outros, alavancando receitas indiretas, como mais servidores, serviços e mesmo softwares não Open Source.

Portanto, precisamos reformular a questão. A receita direta e contabilizável de um determinado software Open Source não implica em medida direta do sucesso ou fracasso do seu impacto econômico. Devemos, para esta análise, olhar o ecossistema como um todo. Um engano comum é comparar as receitas de um determinado software comercializado na modalidade de vendas de licenças com a receita obtida pelos distribuidores de softwares Open Source. Ora as distribuições pagas de softwares Open Source, de maneira similar ao SaaS, são comercializados pelo modelo de negócios de assinaturas, com a receita sendo distribuída ao longo de vários anos e não concentrada em um único pagamento. Assim, comparar receitas obtidas por modelos de negócio diferentes é comparar laranjas com bananas.

Além disso, não existe correlação entre a receita direta obtida com determinado software e seu uso pela sociedade. É muito difícil medir com precisão o uso de um software Open Source. Podemos contabilizar os downloads registrados a partir de um determinado site associado ao software em questão. Mas, a partir daí, como é perfeitamente possível e até mesmo incentivada sua livre distribuição, fica difícil contabilizar as inúmeras outras cópias que circularão pela Web.

Entretanto, é indiscutível que Open Source está se disseminando rapidamente. Os seus principais apelos para o mercado são bastante motivadores: não demanda desembolso prévio para licença de uso (troca capex ou custo de capital por opex, que é custo operacional), menor custo de propriedade, evita aprisionamento forçado por parte de fornecedores e maior facilidade de customização, pelo livre acesso ao código fonte. Também observamos que sua disseminação não é homogênea por todos os segmentos de software. Sua utilização é muito mais ampla em sistemas operacionais, web servers e bancos de dados, mas ainda restrita em outros setores, como ERP e business intelligence.

Mas, Open Source não cresce não apenas no campo do uso tradicional de software, que são os aplicativos comerciais. Vemos sua disseminação se acelerando à medida que a Web se dissemina (muito do código que existe rodando na Web 2.0 e redes sociais é baseado em linguagens dinâmicas em Open Source como PHP, Python e Ruby) e veremos muito código Open Source sendo a base de sensores, atuadores, set top boxes da TV digital, netbooks,

celulares e outros novos equipamentos. Open Source também está na base tecnológica de muitas infraestruturas de cloud computing.

O termo “Open Source vendor” que era considerada uma contradição no primeiro momento, começa a se popularizar de forma bem rápida. À medida que o mercado Open Source amadurece, fica claro que depender única e exclusivamente de comunidades de desenvolvedores voluntários não atende adequadamente às demandas empresariais. Os usuários corporativos exigem níveis de serviço e suporte que a maioria das comunidades “loosely coupled”, típicas do Open Source, não conseguem atender. Abriu-se espaço para o surgimento de novos negócios, intermediários entre as empresas e as comunidades. O que estes novos negócios (“Open Source vendors”) oferecem é um suporte comercial similar ao dos negócios tradicionais da indústria de software. Algumas destas empresas tornaram-se grandes corporações, como a Red Hat, que em 2008 conseguiu receitas de mais de meio bilhão de dólares.

A cadeia de valor do Open Source é constituída de diversos atores como a comunidade (que contribui com código); os “Open Source vendors” que oferecem suporte de segundo nível, localizam software e desenvolvem e comercializam distribuições e edições dos softwares; Open Source VARs que oferecem suporte de primeiro nível, implementação e treinamento; e finalmente os próprios usuários dos softwares Open Source, que usam o software, identificam bugs e também contribuem com código para a comunidade.

Uma diferença fundamental entre os modelos de comercialização do Open Source e os modelos de venda de licença tradicionais é que no Open Source o processo de aquisição de software é direcionado pelo próprio usuário, que pode fazer download do software e testá-lo, sem intervenção do fornecedor. Após os testes ele pode continuar usando uma versão free ou contratar assinatura da versão comercial, que na verdade é muito mais um contrato de serviços. Na prática, a razão de downloads para assinaturas pagas é de pelo menos 1.000 para um. Ou seja, para cada 1.000 downloads, um contrato de assinatura é assinado. No modelo tradicional este nível de conversão de experimentos para contratos seria simplesmente inaceitável e levaria a empresa de software à falência. No Open Source é perfeitamente válido, pois a maior parte das ações de pré-vendas é feita pelo próprio usuário do software. O usuário é quem bate na porta do vendedor quando está interessado na assinatura do software e não o contrário! Uma característica interessante deste modelo é que a maioria dos “Open Source vendors” tem muito mais profissionais em áreas técnicas que em vendas, enquanto que no modelo de comercialização tradicional, a força de vendas é proporcionalmente muito maior.

Uma consequência da maior disseminação do Open Source é que começamos a ver empresas de software, que até a pouco tempo eram hostis ao movimento já se envolvendo ou no mínimo se tornando neutras em relação ao conceito. E para o futuro? Um insight, sujeito a correção de rumo na sua intensidade e velocidade: as empresas de software que estavam fora do Open Source vão integrar suas soluções com estes softwares. Este movimento deve se acelerar com a rápida disseminação do SaaS. A natureza do modelo de negócios SaaS demanda que as licenças de software a serem pagas a terceiros pelos provedores de soluções SaaS sejam mínimas e o Open Source se encaixa muito bem neste

contexto. Muitas destas empresas vão seguir os passos da IBM e se engajar nos projetos de Open Source, inclusive até mesmo colaborando ativamente com a comunidade. É provável que nos próximos anos o modelo de negócios dominante da indústria seja híbrido, com Open Source e venda de licenças integrados na comercialização das stacks de software.

Open Source no ensino da computação (Setembro)

Um tema que me entusiasma é o uso de open source/software livre nos cursos de formação de profissionais de TI. Já havia abordado o assunto em um post anterior e volto a ele hoje com algumas outras idéias que gostaria de compartilhar com vocês.

O uso de open source na formação dos futuros profissionais de computação nos traz diversos benefícios como:

- a) Possibilita que os estudantes adquiram experiência prática com desenvolvimento de software, necessária quando de eventuais futuras contratações, pois desenvolvem código real que será avaliado e até mesmo colocado em operação;
- b) Possibilita que eles compreendam, na prática, a importância dos princípios de engenharia de software;
- c) Possibilita que eles trabalhem em colaboração com profissionais já experientes e com estudantes de outras instituições;
- d) Aprendem que programação não é uma tarefa isolada, mas colaborativa;
- e) Aprendem a trabalhar em projetos de razoável (e até mesmo alta) complexidade, abrindo a visão prática para as dificuldades do engajamento em projetos destes portes;
- f) Aprendem a trabalhar em projetos que tem vida útil longa, não ficando mais restritos a projetos individuais, que duram apenas o semestre de aulas;
- g) Mantém os currículos atualizados, pois estarão envolvidos em projetos atuais, usando técnicas e tecnologias modernas.

Mas, em quais projetos os estudantes deveriam se engajar? Não faltam projetos. O diretório Sourceforge registra mais de 170.000 projetos e mais de 1,7 milhões de membros. Claro que existem projetos open source muito conhecidos como o Linux, Firefox e o BROffice, mas que tal pensarmos em olhar software também como uma tecnologia que pode e deve beneficiar diretamente a humanidade e a sociedade? Pesquisando o assunto descobri um projeto muito interessante que pode servir de base para o uso do open source no ensino da computação no Brasil. É o projeto HFOSS (Humanitarian Free and Open Source Software), que pode ser visto em www.hfoss.org. Este projeto, por sua vez, foi baseado nas idéias de um projeto open source para gerenciamento de desastres, desenvolvido no Sri Lanka, chamado de Sahana (<http://www.sahana.lk/>). Este projeto foi iniciado por desenvolvedores voluntários, para ajudar na gestão de atividades de apoio em desastres, como o tsunami ocorrido naquele país em dezembro de 2004.

Engajar estudantes de computação na criação, manutenção e evolução de projetos de software humanitários, além dos benefícios que vimos acima, contribui para que eles

percebam que desenvolver software é muito mais que codificar linhas de código. Eles poderão ver os resultados de seu trabalho refletidos diretamente na sociedade. Aprendem também que os desenvolvedores devem ter uma visão mais abrangente dos problemas e desafios da sociedade e das empresas, para poderem projetar softwares mais adequados ao mundo real. Este, aliás é um dos desafios que os cursos de computação enfrentam: na maioria das vezes estão descolados das realidades do mundo real, pois os estudantes desenvolvem projetos hipotéticos de classe, sem conexão com os problemas das empresas e da sociedade.

Na nossa realidade que tipo de projetos podemos desenvolver? Olhemos por exemplo a gestão de recursos importantes como a área de saúde. Estima-se que pelo menos 30% dos que se gasta nestes hospitais é desperdiçado de diversas maneiras, por absoluta falta de gestão. Portanto, sistemas de gestão hospitalar, desenvolvidos em software livre, por estudantes de computação trariam como resultado uma melhoria significativa nestes processos. Além disso, ia ficar claro que os estudantes de computação teriam que aprender o que são sistemas de saúde, conversar com usuários, como seus pares e professores da área médica, enfim, entenderem que programação é muito mais que simplesmente escrever linhas de código.

Outros setores importantes da nossa sociedade poderiam ser beneficiados. Por exemplo, o Brasil perde cerca de 1/3 de sua produção de soja por falta de tecnologia adequada. Cerca de 40% dos alimentos produzidos aqui são perdidos por problemas de armazenamento, manuseio e transporte, que poderiam ser minorados com softwares adequados. No próprio sistema educacional, a falta de sistemas de gestão faz com que tenhamos uma ineficiência muito grande, que não será resolvida apenas com construção de mais escolas. A maioria das faculdades, sejam públicas ou privadas, não usam softwares modernos de gestão. Usam sistemas desenvolvidos internamente, com muitas limitações e ineficiências. Imaginemos então as escolas públicas espalhadas pelo país!

E que tal pensar em adicionar recursos de acessibilidade aos softwares livres já existentes? Ou criar novos? E porque não usar plataformas como Android para desenvolver soluções para celulares? Afinal no Brasil já temos mais de 160 milhões de celulares e muitos deles permitem acesso à Internet.

E por que não adotar projetos tipo HFOSS no ensino da computação? O modelo open source cria uma disrupção no modelo tradicional de desenvolvimento de software. Mas este modelo não é debatido ou ensinado nos cursos. Claro que desenvolver projetos práticos, como os propostos aqui, de softwares humanitários para ajudar a minimizar nossos problemas sociais, implica em uma transformação na maneira de se ensinar. Os professores e os alunos deverão conhecer e ter experiência prática com linguagens como PHP, Python e Ruby (além, é claro de Java) e deverão se sentir confortáveis no uso de blogs e wikis. Usarão Facebook, Orkut e Twitter como usam o celular. Aprenderão a trabalhar de forma colaborativa...enfim, estamos falando de uma disrupção no modelo de ensino.

Open Source 2.0 no Latinoware2009 (Outubro)

Semana que vem estarei no Latinoware 2009 fazendo uma palestra sobre oportunidades de negócios em Open Source.

Open Source já está deixando de ser notícia de mídia. Não que o assunto tenha se esgotado, mas é que não é mais novidade. Open Source já está no dia a dia das empresas. E um dos negócios em torno do Open Source que está crescendo bastante são exatamente os serviços. Neste modelo, não se ganha dinheiro diretamente com venda de licenças, mas principalmente com serviços. Segundo o IDC, neste ano de 2009 o mercado mundial de serviços em torno do Open Source deverá atingir a casa dos sete bilhões de dólares, um crescimento de 25% sobre o ano anterior. Em 2013, as previsões são que alcancem 12,7 bilhões de dólares! A receita de serviços acumulada entre 2008 e 2013 será de mais de 54 bilhões de dólares. Realmente, estamos falando de muito dinheiro.

O que estes numeros representam? Indiscutivelmente que Open Source já criou um ecossistema forte e saudável e que não tem como ser ignorado.

Open Source já está atingindo a maturidade e as discussões ideológicas estão sendo deixadas de lado. Lembro que em 2004 e 2005 as discussões se centravam na luta entre o bem e o mal, com o mal simbolizado pelo software proprietário. Pouquíssimas discussões eram pragmáticas...Não se falava em mercado, das dificuldades das empresas em adotar Open Source. Muitas iniciativas eram, no âmbito governamental, definidas por decreto, sem uma visão clara e consistente dos desafios a serem enfrentados. Na época, a IBM era a única das grandes empresas de software a acreditar e investir em Open Source. Já em 2001, a IBM anunciava investimentos de um bilhão de dólares em iniciativas ligadas ao Linux. Foi um marco para a indústria de software, sinalizando que Open Source era coisa séria.

Hoje o contexto é diferente. Muitos projetos de software Open Source já estão amadurecidos. O Linux já tem 18 anos, o MySQL 14 anos, o PostgreSQL é de 1996, o Apache surgiu em 1995, o Eclipse em 2001, Mozilla em 1998 e o OpenOffice em 2000.

A indústria de software, salvo ainda raras exceções já entendeu que Open Source é irreversível. E muitas empresas de software misturam código Open Source com seu código proprietário, criando soluções híbridas. Estas alternativas abrem espaço para produtos inovadores como os ofertados pela Black Duck Software (<http://www.blackducksoftware.com/>) e OpenLogic (<http://www.openlogic.com/>) que vasculham código e validam se existe alguma violação de patentes.

Também já vemos Open Source entrando em outros setores de software, como um sistema de sistema de trouble ticket, o OTRS (<http://www.otrs.org/>), e sistemas de shopping cart para comércio eletrônico como o osCommerce (<http://www.oscommerce.com/>) ou o Zen Cart (<http://www.zen-cart.com/>). Existem soluções para integração de dados e ETL, como o Talend (<http://www.talend.com/>) e diversos outros softwares.

Mas, voltando aos serviços, o segmento de maior crescimento é exatamente o de integração. Curiosamente, os de menor oportunidade de negócios são as atividades de educação e treinamento. Por outro lado, os treinamentos abrem portas para os serviços de maior valor agregado como consultoria e integração.

OK, e que sugestões podemos fazer para empreendedores que queiram entrar neste setor? Apenas duas: Gerar dinheiro com software Open Source é diferente do modelo proprietário. Não existem licenças e portanto devem buscar receita em serviços ou obter ganhos indiretos. E não esquecer que integração entre sistemas proprietários e Open Source é a demanda de maior oportunidade. Assim, não tem sentido uma empresa ser especializada 100% em Open Source.

Adicionalmente sugiro prestar atenção à rápida disseminação da chamada Internet das coisas, onde sensores e outros dispositivos estarão atuando em tempo real, integrados à infraestrutura física do planeta. As oportunidades de novos negócios usando tecnologias Open Source serão absolutamente imensas.

Finalizando, vejam a interessante iniciativa do projeto colaborativo e global, chamado The Open Source Census, que pretende fazer o censo global da adoção de Open Source. O site é <https://www.ossensus.org/>.

Ganhando dinheiro com OpenSource (Outubro)

Latinoware 2009. Respirando 24 horas Open Source...E aqui e ali algumas conversas interessantes. Um extrato delas: Open Source é um modelo de desenvolvimento, entrega e suporte de software. Não é inerentemente anti-comercial e por incrível que pareça ainda não é plenamente compreendido por muitos empresários e empresas de software no Brasil. Bem, não é para menos, pois não existe uma definição legal de Open Source. Nenhuma organização tem o direito desta “marca” e nem menos governa o conceito. Existe, na verdade, uma entidade chamada Open Source Institute (OSI) que criou uma lista de atributos do que vem a ser Open Source, que geraram uma definição (www.opensource.org), que vem sendo adotada por consenso.

Volta e meia converso com alguns empresários da indústria de software e sinto um certo receio deles em se aproximarem do Open Source. Pensam logo nos riscos de usar Open Source e que também perderão dinheiro, pois sua receita hoje vem da venda de licenças.

Mas, será mesmo que perderão dinheiro? Ou talvez valha a pena pensar “fora da caixa” e identificar outras alternativas no uso deste modelo em seus negócios?

Vamos citar alguns exemplos. A IBM investe no desenvolvimento do Linux com seu LTC (Linux Technology Center) porque consegue com isso não apenas ter maior compreensão do Linux, tornando os seus próprios produtos mais eficientes neste ambiente, como também influencia a comunidade com sua experiência. Mas, lembrem-se um software Open Source não pertence a nenhuma empresa. Neste modelo, nenhuma entidade mantém controle sobre as propriedades intelectuais que fazem o software.

Mas, investindo em tornar o Linux mais eficiente permite que a IBM amplie sua base de venda de servidores e softwares proprietários, engajando-se em um mercado que cresce mais rápido que qualquer outro ambiente operacional. Como um empresário pode agir de forma similar? Imagine que seu código seja baseado em Java ou PHP. Se ele colaborar com estes projetos, estará criando relacionamentos mais estreitos com suas comunidades.

O empresário de software também pode pensar em usar Open Source embutido em seu produto. Neste caso usaria seu software de código fechado ao lado de alguns componentes Open Source. Claro que para isso deve escolher projetos Open Source maduros, que apresentem uma base de código estável e já provada em campo. Buscar um código novo, com comunidade pequena e incerta, traz embutido o risco do empresário ter que assumir a responsabilidade pela manutenção e evolução do código. Pode ser um ônus insuportável. Esta opção parece ser interessante e algumas estimativas, como a do Gartner, apontam que por volta de 2013 cerca de 80% dos produtos de software fechados embutirão elementos Open Source. Como benefício, evita-se escrever código que já está (e bem) escrito, podendo-se concentrar esforços e investimentos nos códigos que vão gerar valor para seus clientes.

Uma outra alternativa é transformar o seu código em Open Source. Entretanto, neste caso existe a questão de como monetarizar o software. Como gerar dinheiro com os downloads gratuitos? De maneira geral existe uma diferença de uma ordem de magnitude entre downloads (o usuário está simplesmente investigando o software), uso operacional e pagamento de algum valor por, digamos, upgrades. Existe uma rule-of-thumb que diz que para cada 1 milhão de downloads temos 100.000 usuários e que destes apenas 1.000 pagarão pelo software. Como obter dinheiro em uma relação de 1.000 x 1?

Antes de mais nada precisa-se criar uma imensa base de usuários. Coisa nada fácil. Mas, existe uma alternativa que é a criação do modelo dual-licence. Neste modelo cria-se uma versão aberta e uma fechada, com algum atrativo que gere valor para o cliente. Este, por sua vez, concorda em pagar por este valor agregado. Um exemplo comum é a criação de uma versão “premium” ou “enterprise” com mais recursos, que demandam pagamento para seu uso.

E, existe, é claro, a opção de gerar retorno financeiro com venda de serviços e treinamento.

Enfim, são diversas maneiras de uma empresa de software se aproximar do modelo Open Source, além de usar Linux nos seus servidores e Eclipse no desenvolvimento de seu código...Não é uma aposta de tudo ou nada, mas que deve ter uma estratégia bem definida e a escolha de modelos de aproximação que agreguem valor para seu negócio. Neste caso, Open Source será um bom negócio.

Open Source nos Smartphones (Dezembro)

Este ano participei de vários eventos de Open Source. E na imensa maioria deles, nas conversas de cafézinho, surgia a clássica pergunta: “voce acha que um dia o Linux vai substituir o Windows nos PCs?”. Para mim esta é uma questão de pouca importância. Os PCs fazem parte de uma fase da evolução da TI e devemos prestar atenção no que está vindo aí. Não no retrovisor.

E o que vem por aí? Computação em Nuvem e os smartphones! Vamos começar pelos smartphones. Estes dispositivos ainda são menos de 20% dos celulares fabricados no mundo, mas em pouco mais de cinco anos já serão a maioria. São verdadeiros e poderosos computadores, e estão evoluindo muito rapidamente. Há dez anos o iPhone era futurologia. Daqui a dez anos seus recursos serão básicos em qualquer celular. E nem me atrevo a dizer o que será o smartphone de 2019...

O que roda nestes dispositivos? Já está claro que as plataformas Open Source dominam este cenário. Android, Maemo, Symbian (se tornando Open Source, sob licença Eclipse Public Licence agora em 2010) e outros sabores de Linux já estão próximos de deter 60% do mercado. Os sistemas proprietários RIM (BlackBerry OS) e Apple ficam com cerca de 30% e o Windows Mobile com a pequena parcela de 10%.

O Android, na minha opinião será o “flavor” Linux mais consistente destes dispositivos. Já está sendo adotado por diversos fabricantes como Motorola, Samsung, SonyEricsson, LG e HTC, e possui um catálogo com milhares de aplicativos, o Android Market (<http://www.android.com/market/>).

Este movimento em direção à Open Source deve mudar bastante as regras do jogo no cenário de smartphones. Primeiro, a ativa participação de comunidades de desenvolvedores e o fato de não ser mais necessário pagar-se por royalties, como no modelo adotado pelo Windows Mobile, deve acelerar o ciclo de inovações e lançamentos de novos modelos. O valor comercial de sistemas operacionais para computação móvel vai cair bastante. Isto vai afetar em muito a Microsoft, que tem seu modelo de negócios neste cenário baseado na venda de licenças e royalties. Algumas estimativas apontam que os royalties do Windows Mobile situam-se entre 15 e 25 dólares por unidade vendida. Na minha opinião, o Windows Mobile vai ser relegado a nichos de mercado, ficando espremido entre a forte e crescente presença de sistemas Linux-based de um lado, e por alguns sistemas proprietários, como o da Apple e seu iPhone, de outro.

Mas, o cenário da computação nos dispositivos “cliente” ainda vai mudar mais. Há poucos meses o Google anunciou um novo sistema operacional para PCs e netbooks, open source, baseado no kernel do Linux, chamado de Chrome OS. Este sistema deve, em princípio, estar disponível a partir do segundo semestre do ano que vem.

O Chrome OS colide diretamente com o modelo estabelecido da indústria de sistemas operacionais para PCs, dominado hoje pela Microsoft com seu Windows. O Chrome OS é orientado ao modelo de computação em nuvem e na verdade sua arquitetura, baseado nas

poucas informações disponíveis, me parece o browser Chrome rodando um sistema de janelas em cima de um kernel Linux. Ou seja, o Chrome OS é uma plataforma para o browser Chrome e suas aplicações. O Google afirmou que seu código será Open Source e eu acredito que será baseado na licença GPLv2. Usando o kernel do Linux, o Chrome aproveita todo o desenvolvimento já feito em drivers e a sua imensa comunidade de desenvolvedores.

O Chrome OS será inicialmente centrado nos netbooks, um mercado que cresce a cada dia. Os netbooks são máquinas voltadas a operarem em nuvens computacionais. Uma parcela significativa da base instalada de netbooks já roda Linux e este número deve aumentar à medida que mais e mais destas máquinas equipadas com processadores ARM entrarem no mercado.

O que o Chrome nos sinaliza? Que nos próximos anos assistiremos a uma batalha interessante pelo mercado de PCs e netbooks, quando dois modelos de negócio diferentes estarão em choque. O modelo tradicional, como o da Microsoft e o do Google. A rede de valor do Google é baseado na economia do grátis, onde o software é meio para vender mais anúncios. Os exemplos estão se encaixando: o Chrome OS, o browser Chrome, o Google Gears, Google Apps e o Google AppEngine. Esta rede de valor é diferente do modelo da Microsoft, que obtém sua receita exatamente da venda de licenças de seus softwares.

Um ponto chave para o sucesso do Google será a aceitação do modelo de computação em nuvem. Este modelo computacional vai se consolidar nos próximos anos e muitas das atuais limitações e restrições serão minimizadas ou até mesmo eliminadas.

Portanto, voltando a pergunta inicial, na minha opinião, em menos de dez anos, o mercado de dispositivos de acesso à Internet (que será basicamente constituído por smartphones e netbooks) estará sendo dominado pelo modelo de Computação em Nuvem, Open Source e sistemas Linux-based. O ciclo atual, dominado pelo PC e o Windows será cada vez menos importante. Será visto pelo espelho retrovisor...

Artigos publicados na revista Espirito Livre em 2010

Em 2009 comecei a colaborar com a excelente revista eletrônica Espirito Livre (<http://www.revista.espiritolivres.org/>). Das colunas que escrevi em 2009 e 2010, selecionei alguns dos textos publicados em 2010 para compartilhar com vocês aqui no blogbook.

Impacto Econômico do Open Source

Um assunto que merece nossa atenção e que nem sempre é debatido com profundidade nos eventos e artigos específicos sobre Open Source é o seu impacto econômico. Entretanto, existe um estudo muito interessante e bem detalhado, escrito em 2006, mas ainda bem atual, que é o relatório “Economic Impact of FLOSS on innovation and competitiveness of the EU ICT sector”.

Open Source é um movimento que deve ser estudado profundamente. Não pode ser considerado apenas de forma simplista, como um movimento ideológico de hackers. Tem profundos impactos na indústria de software. Aliás, pode e já está transformando toda a indústria de software.

Alguns dados extraídos deste relatório, que mostra de forma clara o impacto do Open Source na economia e indústria de software na Europa e no mundo:

- a) Em market share está claro que Open Source está se disseminando muito rapidamente pelo mundo todo. Em alguns setores é líder ou está entre os três ou quatro primeiros softwares em termos de utilização, como sistemas operacionais (Linux), web servers (Apache), browsers (Firefox), databases (MySQL e PostgreSQL), suites de escritório (OpenOffice), linguagens de programação (Perl, PHP...).
- b) Europa e EUA são as regiões que concentram a maior parte dos desenvolvedores. Provavelmente Ásia e América Latina enfrentam o problema da língua, pois a maioria das comunidades dos softwares mais importantes dialoga em inglês. Na Ásia, Índia e China se destacam. O Brasil aparece com destaque no uso do Open Source em órgãos governamentais.
- c) Falando em impactos econômicos, o estudo da base instalada de Open Source teria custado às empresas européias, se fosse desenvolvido nos moldes tradicionais, cerca de 12 bilhões de euros. Este valor foi calculado à época e devemos lembrar que o volume de código aberto está dobrando a cada 18/24 meses.
- d) Estima-se que o esforço de desenvolvimento e manutenção do Open Source (os grandes projetos de software tem cerca de 50% de seu código base substituído a cada cinco anos) equivale a uma força tarefa de 131.000 profissionais/ano. O estudo avalia esta contribuição da comunidade em 800 milhões de euros por ano.
- e) O estudo também estima que em fins deste ano de 2010, o ecossistema Open Source vai alcançar cerca de 4% do PIB europeu!
- f) O estudo também mostrou o potencial da informatização por empresas e administrações governamentais de pequeno porte (como prefeituras), que não têm capital para maiores investimentos em aquisições de software. Com Open Source as

barreiras de entrada caem significativamente, abrindo maiores oportunidades para estas empresas se informatizarem.

- g) O relatório aponta como um efeito positivo do Open Source na Europa, o potencial de geração de empregos e criação de pequenas empresas de software e serviços, principalmente porque na UE não existe tanto VC (venture capital) disponível, nem uma cultura de risco como existe nos EUA.
- h) Um aspecto destacado pelo estudo é a oportunidade de desenvolvimento de skills, que incluem além da óbvia habilidade de programação, conhecimentos dificilmente aprendidos nos cursos formais de ciência da computação como aspectos jurídicos (licenças e patentes/copyrights de software), trabalho colaborativo em equipes e coordenação de comunidades. Uma das pesquisas do estudo mostrou que 78% dos desenvolvedores se juntavam à comunidades Open Source para aprender e desenvolver novos skills.
- i) O estudo também mostra o potencial do Open Source em incentivar inovação (destruição criativa), apontando o fato que a indústria de software tradicional tende muitas vezes a inovar de forma incremental, não disruptiva. Estima que pelo menos 36% de investimentos em P&D de software poderiam ser economizados com uma maior sinergia entre a indústria e as comunidades Open Source.
- j) Um importante aspecto destacado pelo relatório foi o crescente uso do Open Source na produção de produtos e serviços onde software é embarcado, como equipamentos médicos, eletroeletrônicos, celulares, automotivos e outros. Nestes produtos o software é meio e não fim, e o uso de Open Source como o sistema operacional Linux reduz o seu custo de desenvolvimento e acelera a sua entrada no mercado. O estudo exemplificou mostrando como a Nokia ganhou tempo com sua plataforma Maemo, baseada na distribuição Debian. Analisando um releasase específico, denominado Mistral, liberado em agosto de 2006 para o projeto experimental Tablet 770, eles viram que a Nokia precisou desenvolver apenas 200.000 linhas de código (1,5% das mais de 15 milhões de linhas da plataforma Maemo), ganhando com isso tempo e poupando investimentos que puderam ser aplicados em novos desenvolvimentos. Caso tivesse que desenvolver tudo do zero a Nokia precisaria alocar 12.000 desenvolvedores/ano e mais de 870 milhões de euros.

Além dos dados deste estudo outros números nos chamam a atenção. Somente no ano passado foram criados mais de 19.000 projetos Open Source. Se somarmos o valor de todas as centenas de milhares de projetos Open Source que existem hoje em dia, com seus bilhões de linhas de código, chegaríamos ao estonteante valor de 400 bilhões de dólares.

O que estes numeros demonstram claramente é que Open Source não deve ser encarado de forma simplista como um movimento restrito a discussões ideológicas, onde a decisão de ir para Open Source é apenas porque é Open Source, mas como um meio eficaz de proporcionar desenvolvimento econômico.

Aliás, o modelo Open Source começa a se espalhar por outros setores econômicos. Recentemente a GlaxoSmithKline PLC abriu à comunidade pesquisadora os modelos de 13.500 componentes químicos, que segundo ela, seriam capazes de inibir o parasita que causa a malária. A expectativa da Glaxo é que o compartilhamento de informações e a pesquisa conjunta (conceito Open Source), que deram certo no setor de software levem os

cientistas a descobrir uma droga mais rapidamente que se trabalhassem isoladamente. É uma quebra de paradigmas, pois a indústria farmacêutica guarda como segredo de estado as fórmulas de seus remédios, assim como a indústria de software sempre manteve a sete chaves o código fonte de seus produtos.

A iniciativa é pragmática, pois abre as fórmulas para doenças, como a malária, que afligem populações pobres e que os remédios “proprietários” desenvolvidos para tratá-las não prometem retornos muito altos. O desenvolvimento de um remédio é um processo caro, de erro e acerto, que envolve identificar quais componentes químicos produzem certos efeitos num determinado alvo biológico. No ano passado a Glaxo testou dois milhões de compostos, selecionando as 13.500 moléculas, que segundo ela, potencialmente apresentam algum efeito. Mas, reduzir a lista de compostos a um número bem mais limitado de itens que possam levar ao desenvolvimento de um remédio é um processo caro e complexo. Qualquer componente que se prove promissor levará anos de pesquisa e investimentos para se transformar em um remédio contra a malária.

A proposta Open Source da Glaxo sinaliza que a empresa não patenteará qualquer remédio contra a malária que possa ser descoberto a partir destes componentes e espera que outros pesquisadores doem sua propriedade intelectual para um pool de patentes voltado a doenças negligenciadas, como a malária.

Contribuindo para o Kernel do Linux

Outro dia estava conversando com um amigo sobre o Linux, sistema que fará 20 anos no ano que vem. O Linux já representa hoje uma força econômica considerável, com um ecossistema estimado em 25 bilhões de dólares. Está em praticamente todos os lugares. Quando fazemos uma pesquisa no Google ou lemos um livro no Kindle é o Linux que roda nos bastidores.

O Linux demonstrou de forma inequívoca o potencial do desenvolvimento de sistemas de forma colaborativa, que é o cerne do modelo Open Source. Seria praticamente impossível para qualquer empresa de software, sózinha, criar um sistema operacional de seu porte e complexidade. Estima-se que o custo de desenvolvimento de uma distribuição como a Fedora 9, com seus mais de 204 milhões de linhas de código corresponda a quase 11 bilhões de dólares. Para chegar ao kernel 2.6.30 (mais de 11 milhões de linhas de código) o investimento seria de mais de 1,4 bilhão de dólares.

Para se ter uma idéia do volume de trabalho em cima do kernel, nos ultimos 4 anos e meio, a média foi de 6.422 novas linhas de código adicionadas por dia, além de outras 1.687 alteradas e 3.285 removidas. Do 2.6.24 ao 2.6.30 a média subiu para 10.923 linhas de código adicionadas por dia. Quem teria cacife financeiro para sustentar, por si, um empreendimento bilionário destes?

Portanto, o Linux é uma força no presente. É usado não apenas em web servers e print servers, mas a cada dia vemos mais e mais aplicações core das empresas operando em plataformas Linux. A IBM, por exemplo, consolidou seu ambiente de computação interno (seus sistemas internos) em plataformas mainframe System Z, rodando Linux.

Mas, e o futuro? Com as mudanças que já estão acontecendo, como a crescente disseminação da computação móvel e o modelo de computação em nuvem (Cloud Computing), como o Linux se posiciona?

É uma resposta fácil. O Linux, que não conseguiu muito espaço nos desktops está se posicionando como plataforma dominante na computação móvel, como smartphones, netbooks e tablets. No seu último ano fiscal que se encerrou em 30 de junho do ano passado, a Microsoft pela primeira vez reconheceu em seu relatório para os acionistas que os sistemas Linux para máquinas cliente, basicamente netbooks, da Red Hat e Canonical seriam ameaças ao seu negócio. Anteriormente, só havia reconhecido a ameaça do Linux nos servidores.

E quanto aos servidores? Nestas máquinas o Linux está altamente alinhado com as tendências de virtualização e cloud computing. Alguns pontos positivos sobre o Linux chamam a atenção, quando falamos em cloud. Primeiro, o Linux opera em praticamente qualquer plataforma de hardware, o que facilita o provisionamento, alocação e gerenciamento de recursos computacionais em nuvem. Podemos criar desde uma nuvem baseada em plataforma x86 como o Google até nuvens em mainframes IBM, aproveitando o alto throughput e facilidade de virtualização destas máquinas. Falando em mainframes, agora em maio, fez dez anos que o Linux roda nestas máquinas.

O custo de licenciamento é outro fator interessante. Embora existam distribuições licenciadas, um provedor de infraestrutura em nuvem, pelo grande número de servidores que deverá dispor (falamos aqui em milhares ou dezenas de milhares de máquinas), poderá adotar, pela escala, versões Linux não comerciais. Virtualização é outro plus do Linux, com diversas tecnologias disponíveis como Xen (base da arquitetura de cloud da Amazon) e KVM.

Hoje, se olharmos Linux em cloud já vemos seu uso como base tecnológica da nuvem do Google, da Amazon, do Force.com do Salesforce e de startups como 3Tera (recentemente adquirida pela CA), Elasta e Mosso, entre outros. E com certeza seu uso se alastrará pelas futuras ofertas de nuvens.

Então, porque não colaborar com esta força, contribuindo para a comunidade Linux? Recomendo a leitura de um interessante paper escrito por Jonathan Corbet, “How to participate in the Linux Community”, disponível no site da Linux Foundation em <http://ldn.linuxfoundation.org/book/how-participate-linux-community>.

Modelos de Negócio em Open Source

Outro dia mantive um animado debate com alguns amigos sobre como ganhar dinheiro com Open Source. O assunto era identificar quais os modelos de negócio que podem realmente fazer um projeto Open Source ser sustentável economicamente. Um negócio, qualquer que seja ele, tem um grande desafio: como gerar receitas para se manter e crescer. Um projeto Open Source pode ser mantido de várias maneiras, algumas indiretas, outras que permitem geração direta de receitas. Nas indiretas, os ganhos são advindos de atividades ao redor do software, como treinamento, suporte e consultoria, ou mesmo, incentivando a geração de receitas obtidas por outros produtos, como venda de componentes de software ou add-ons ao produto original, bem como venda de hardware com o software embarcado. Entretanto, as estatísticas tem mostrado que os usuários que pagam por serviços especiais, como assinatura com direito à suporte, são um minoria diante do universo de usuários. Alguns números mostram que em média, apenas 0,1% dos downloads geram um contrato de suporte assinado. Mas, manter a versão gratuita é altamente benéfico pois gera um ecossistema que gravita em torno do software, abrindo espaço para serviços de treinamento e consultoria. Aliás, um ecossistema saudável e atuante é fundamental para a sobrevivência dos projetos Open Source. Sem um suporte ativo e comprometido da comunidade, o projeto tende a cair no ostracismo.

Como receita direta um modelo que tem sido adotado por alguns projetos e que aparentemente tem dado certo é o dual licencing. Neste modelo o software é distribuído em duas versões, uma baseada em licenças livres como GPL e outra, proprietária, com o usuário adquirindo o direito de uso do software através de um licenciamento comercial. A versão proprietária geralmente apresenta algumas diferenças de funcionalidade, que justificariam o interesse do usuário em adquirir uma licença de uso. Outras vezes, o interesse é demandado pela característica do software, que permite que sua versão proprietária seja embarcada em outro software, não conflitando com os interesses comerciais da empresa produtora deste produto. Caso típico são os softwares de banco de dados, que tendem a ser embarcados em outras aplicações, estas sim, comerciais. Desenvolver a partir do zero um software de banco de dados é extremamente caro e a possibilidade de se embarcar um, já pronto e testado, é uma vantagem de tempo e dinheiro enorme. Um exemplo deste modelo é o MySQL, que agora faz parte da Oracle. Claro que existem regras definidas para que o modelo “dual licencing” funcione. A comunidade pode alterar a versão GPL e neste caso a MySQL não poderia exigir copyrights destas modificações. Como resolver a questão? A MySQL mantém seu próprio corpo de desenvolvedores que são os únicos autorizados a incluir modificações na versão copyright. Também requisita que os contribuidores externos assinem contratos de copyright para modificações que considera úteis e adquire seus direitos de uso.

O modelo Open Source abre também oportunidades para entrada em mercados de aplicativos antes inatingíveis. Um fabricante global de software tende a se concentrar nos mercados de maior porte e rentabilidade, com menos investimentos em mercados menores, nos quais os custos de localização nem sempre são compensadores. O resultado é que muitas vezes as inovações e atualizações tecnológicas são introduzidas com atraso nos mercados menores. Outro problema é a usabilidade (não apenas a tradução de telas), mas

adaptação às características coloquiais de comunicação da língua nacional, que dificilmente são implementadas nos mercados menores.

Além disso, alguns tipos de aplicação demandam características totalmente diferentes. Um exemplo: aplicações estrangeiras de gestão hospitalar e clínicas não se adaptam facilmente às especificidades dos pequenos e carentes hospitais brasileiros. Atendem apenas (e muitas vezes parcialmente) aquela parcela mínima de hospitais classe primeiro mundo, que se contam em poucas dezenas. Outros exemplos? Gestão de serviços públicos. Os processos dos órgãos públicos brasileiros são bem diferentes, inclusive nos aspectos culturais aos implementados em soluções estrangeiras. Surgem também inúmeros espaços em setores onde pequenas empresas especializadas demandam aplicações específicas, como às encontradas em arranjos produtivos locais como têxtil ou calçados. Estes mercados não são atendidos de forma adequada por produtos estrangeiros, pois são mercados mínimos e de baixa rentabilidade. É um mercado não alcançado pelas grandes empresas de software e de alto potencial para empresas menores.

Por outro lado, as pequenas empresas nacionais não tem capacidade financeira, tecnológica e empresarial para atender a este mercado isoladamente. São poucas as empresas nacionais com capital suficiente para serem competitivas nacionalmente. O setor de software no Brasil é composto por 94% de micro, pequenas e médias empresas. Uma solução é desenvolver aplicativos na forma colaborativa, no modelo Open Source, com contribuição ativa de universidades que conhecem bem os processos dos setores de indústria e tem mão de obra tecnológica. Neste modelo os custos de desenvolvimento se reduzem pois são compartilhados e as empresas podem atuar de forma integrada na prestação de serviços. Na prática criam uma rede de pequenas empresas, com cada membro conquistando clientes em suas regiões de origem e desenvolvendo parcerias estreitas com os outros membros, que tenham expertise complementar, suportando-se uns aos outros. Assim, as pequenas empresas regionais, sem capital para se tornarem nacionais, se associam em rede para criar uma “empresa virtual” especializada.

De onde pode vir a receita destas redes de empresas? Que tal pensar em componentes específicos, instalação, customização, integração e treinamento? E, porque também não em Software-as-a-Service?

Esta rede social de negócios baseados em Open Source pode abrir novas oportunidades de mercado, hoje inexistentes. O software é o mesmo, seja implementado em Rondônia ou no Espírito Santo, e a troca de expertise facilita o processo de implementação e as empresas concentram-se em fazer o aplicativo funcionar para atender ao negócio do cliente e não em escrever e depurar software.

O resultado final é um maior nível de informatização e eficiência das pequenas firmas nacionais (ainda muito limitada) e um maior e mais rico ecossistema para a indústria nacional de software. Lembrem-se. No mundo capitalista o dinheiro não some, apenas desloca-se de um lugar para o outro: da venda de licenças limitadas a um mercado relativamente restrito a um mercado bem maior de serviços e assinaturas (Software-as-a-Service).

Mas, uma vez definido o modelo de negócio, qual poderia ser o preço praticado pela versão comercial? Indiscutivelmente que o preço tem muito a ver com a categoria do software. Um software comoditizado, como uma suíte de escritórios, vai disputar um mercado altamente sensível a preços e portanto, tem que oferecer vantagens de preços bem inferiores às versões proprietárias existentes. Por outro lado, se for um software inovador, pode pensar em preços maiores.

Um aspecto importante a ser avaliado quando da definição dos preços para as opções Open Source é o “switching cost”, que embute os custos do usuário ao trocar de uma versão proprietária para a Open Source. Entre estes custos temos treinamento, conversão, integração com outros softwares, suporte e mesmo necessidade de aquisição de produtos proprietários que complementem funções faltantes.

Acabamos chegando a um consenso. Não existe uma resposta única, mas algumas perguntas podem ajudar a definir qual o melhor caminho para um projeto Open Source ser sustentável:

- a) Como o software vai se posicionar no mercado? É um produto comoditizado que vai disputar o espaço com algum produto comercial dominante ou é um software inovador, sem concorrência?
- b) O software atende a um mercado de nicho ou ao mercado horizontal? O tamanho do mercado potencial e a abrangência da comunidade que vai gravitar em torno dele é dependente desta questão.
- c) O projeto tem estrutura por trás que permite gerar receita com treinamento e suporte?
- d) O modelo SaaS é adequado?
- e) O projeto pode ser sustentado por outras receitas indiretas, como venda de hardware?
- f) Quais são os objetivos de negócio dos líderes do projeto Open Source?

Concluimos que existem inúmeras oportunidades de negócio com Open Source e as fontes de receita podem ser indiretas ou diretas. Cada caso é um caso, mas as oportunidades estão aí para os empreendedores de plantão. Talvez uma das principais barreiras ainda seja o mito que Open Source não gera dinheiro. Puro engano.

Open Source Potential Index

Estudei atentamente um trabalho muito interessante desenvolvido pela Red Hat e pelo Georgia Institute of Technology, chamado Open Source Software Potential Index ou OSPI. A idéia básica deste índice é comparar diversos países com relação às suas atividades atuais e potenciais quanto ao uso de Open Source. O resultado pode ser visto em www.redhat.com/about/where-is-open-source/activity onde passando o mouse pelo mapa do globo vemos o posicionamento de cada país e podemos fazer algumas comparações. O mapa mostra cada país pelos critérios Activity e Environment. Activity visualiza fatores concretos como políticas de incentivo ao Open Source e número de usuários, e Environment é mais subjetivo pois tenta apontar condições que permitam ao Open Source florescer.

Medir o uso de Open Source não é uma tarefa fácil, pois nem sempre os dados estão disponíveis. Até mesmo medir o uso de determinado software não é simples, pois ao contrário de um software comercial onde podemos obter do fornecedor o número exato de cópias vendidas, em Open Source os downloads podem ser feitos não apenas a partir dos sites base, mas de inúmeros outros. Cópia de cópia é uma atividade comum no mundo Open Source.

Com base neste índice, podemos avaliar quão bom é o posicionamento de um determinado país em relação ao Open Source e definir ações que possam contribuir para um aumento na velocidade e amplitude de sua adoção.

Analisando o mapa vemos que o Brasil está em uma posição relativamente boa, em 12º lugar. Interessante que na frente do Brasil estão, à exceção da Austrália (4º) e EUA (9º), apenas países europeus. Os três primeiros são a França (1º), Espanha e Alemanha. A Europa considera Open Source estratégico e recentemente, na edição 16 da Espirito Livre abordei esta importância analisando um estudo muito interessante e bem detalhado, escrito em 2006, mas ainda bem atual, que é o relatório “Economic Impact of FLOSS on innovation and competitiveness of the EU ICT sector”.

Mas, ao detalharmos a composição deste índice, observamos que ele é composto pela avaliação do uso de Open Source em três dimensões: governo, empresas e academia/comunidades.

No item governo, o Brasil está em terceiro lugar. Apenas França e Espanha estão à nossa frente. A atuação dos governos varia de país para país, e inclui diferentes ações e estratégias que incluem principalmente iniciativas regulatórias e políticas públicas de incentivo. Alguns países adotam Open Source por necessidade de sobrevivência como Irã e Síria, que não tem outras alternativas, uma vez que estão limitados pelo embargo comercial que os impedem de utilizar a maioria dos softwares comerciais.

Por outro lado, quando analisamos o posicionamento do Brasil nos fatores academia (14º) e ambiente empresarial (43º) vemos que ainda existe muito espaço para uma maior adoção do Open Source. O uso de Open Source pelas empresas privadas brasileiras, sejam usuárias ou desenvolvedoras de software ainda é relativamente restrito. O Linux está bem disseminado, já existem muitos usos em aplicações críticas, mas ainda muitas empresas restringem seu uso

a atividades periféricas, como servidores de impressão ou email. Outros softwares Open Source também ainda não estão amplamente disseminados nas médias e grandes empresas. Na minha opinião, olhando as empresas desenvolvedoras, me parece que o uso ainda restrito do Open Source deve-se ao desconhecimento e a idéia ainda arraigada que Open Source e softwares comerciais são antagônicos.

Um exemplo prático de sinergia entre o modelo comercial e o Open Source é o da IBM. É indiscutível que o famoso anúncio de um bilhão de dólares de investimentos em Linux no Linux World de 2001 foi um marco no movimento Open Source. Sinalizou de forma inequívoca ao mercado que Open Source era sério. Mas o que mais a IBM vem fazendo com relação ao modelo Open Source? Por exemplo:

- a) Doou propriedade intelectual a projetos Open Source. Entre outras ações, a IBM doou código do banco de dados Cloudscape (500.000 linhas de código Java) para a Apache Software Foundation, para alavancar o projeto Derby (<http://db.apache.org/derby>). Também doou o código do Visual Age para a Eclipse Foundation, código que se tornou a base do Eclipse IDE.
- b) Incorporou softwares Open Source em seus produtos. A IBM investe nas tecnologias de Open Source (O LTC – Linux Technology Center – compreende mais de 600 desenvolvedores que escrevem código e o doam para a comunidade Linux) e também incorpora estas tecnologias em seus produtos. Alguns exemplos são a inclusão do Apache no WebSphere e do Eclipse nas ferramentas Rational.

Os objetivos da IBM com Open Source são: aumentar o ritmo e velocidade das inovações, incentivando o “caldo cultural” de inovação nas comunidades (inteligência coletiva), ser um contribuidor ativo das comunidades e não apenas consumidor, capturar e transformar inovações Open Source em valor para seus clientes (como exemplos o uso do Linux nos hardwares IBM e a tecnologia Rational fundamentada no Eclipse) e alavancar modelos de negócios baseados em Open Source para entrar em novos mercados e expandir oportunidades de negócio. É, portanto uma estratégia de negócios e não um simples oportunismo ou uma reação forçada pela pressão do mercado.

Assim, quando vemos que uma empresa do porte e importância para o mercado de TI que é a IBM entrar forte no movimento Open Source, fica claro que este movimento não é brincadeira. Deve, portanto, ser encarado como uma estratégia de negócio. As empresas de software brasileiras deveriam olhar com mais atenção as potencialidades do modelo Open Source e inseri-las em seus modelos de negócio.

Na academia, acredito que ainda exista muito espaço para disseminar Open Source. A imensa maioria dos cursos de computação aborda Open Source de forma tangencial, apenas “aprendendo Linux”. Não insere em suas ementas a participação ativa dos alunos em comunidades desenvolvendo ou depurando código. E, pior, alguns cursos que se dizem de pós-graduação em software livre somente ensinam a usar softwares como Linux e Firefox, e nem de perto passam pela experiência de envolver seus alunos nas comunidades Open Source.

Na minha opinião o uso de Open Source na formação dos futuros profissionais de computação nos traz diversos benefícios como:

- a) Possibilita que os estudantes adquiram experiência prática com desenvolvimento de software, necessária quando de eventuais futuras contratações, pois desenvolvem código real que será avaliado e até mesmo colocado em operação;
- b) Possibilita que eles compreendam, na prática, a importância dos princípios de engenharia de software;
- c) Possibilita que eles trabalhem em colaboração com profissionais já experientes e com estudantes de outras instituições;
- d) Aprendem que programação não é uma tarefa isolada, mas colaborativa;
- e) Aprendem a trabalhar em projetos de razoável (e até mesmo alta) complexidade, abrindo a visão prática para as dificuldades do engajamento em projetos destes portes;
- f) Aprendem a trabalhar em projetos que tem vida útil longa, não ficando mais restritos a projetos individuais, que duram apenas o semestre de aulas;
- g) Mantém os currículos atualizados, pois estarão envolvidos em projetos atuais, usando técnicas e tecnologias modernas.

Portanto, uma rápida análise do OSPI nos mostra que ainda temos um longo caminho a percorrer. Mas, recomendo a vocês visitarem o site e tirarem suas próprias conclusões.

Dez anos de Linux no Mainframe

17 de maio de 2000. Nesste dia, há dez anos era formalmente anunciado o Linux S/390, o Linux nos mainframes. Na época, imaginar um sistema operacional “estranho no ninho” no mainframe era quase uma heresia e até a data do anúncio formal, o Linux S/390 era um projeto quase secreto, pouco divulgado dentro da própria IBM.

Mas sua história é interessante e vale a pena recordar alguns marcos. O projeto começou de forma voluntária em 1998, no laboratório da IBM em Boeblingen, na Alemanha. Não estava no budget e nem era oficialmente autorizado. Mas, no final do ano, a IBM passou a olhar o movimento Open Source e o Linux com outros olhos, quando Sam Palmisano, hoje CEO e na época senior VP, disse em uma entrevista em outubro: “The Internet has taught us all the importance of moving early, the advantage of being a first-mover. We want to be riding that Linux momentum at the front, no trailing it.”.

Em dezembro a IBM publica uma coleção de patches e adições ao então kernel 2.2.13 para o mainframe, abrindo o sistema para avaliação do mercado, criando um alvoroço no mercado e na comunidade Linux. Chega o ano 2000 e as coisas avançam bem rápido. Em janeiro o Linux S/390 é disponibilizado para demonstrações públicas a partir do mainframe do Marist College, nos EUA e em pouco tempo registram-se 4.000 downloads. No mês seguinte, no LinuxWorld Expo, em New York, Linus Torvalds em seu keynote speech menciona o Linux no mainframe. E em 17 de maio é anunciado formalmente o Linux S/390, merecendo inclusive anúncio de página inteira no Wall Street Journal. Neste mesmo ano consegue-se uma experiência incrível: um mainframe consegue rodar mais de 41.000 instâncias do Linux em um único LPAR, sob VM. Dentro da IBM, colocar o Linux no mainframe sinalizou diversas mudanças. Uma delas afetou inclusive as questões de propriedade intelectual, pois os drivers dos periféricos do mainframe eram IP (intelectual property) da IBM e na primeira versão do Linux S/390 tiveram que ser disponibilizados apenas em código objeto. Posteriormente a IBM cedeu os direitos de licenciamento e o código hoje está aberto e disponível sob licença GPL.

No ano seguinte a IBM faz o famoso anúncio, no LinuxWorld de New York, de investir um bilhão de dólares no desenvolvimento do Linux. Este anúncio foi um marco para o movimento do Open Source e do Linux, pois sinalizou claramente para as empresas que Linux era coisa séria. Neste evento, o mainframe z900 rodando Linux ganhou o prêmio de “Best Hardware”. Torna-se também disponível a primeira distribuição oficial do Linux para mainframe, o SUSE Enterprise Linux Server 7. A SUSE esteve envolvida desde o início no projeto do Linux S/390, inclusive porque seu escritório ficava em Nuremberg, há menos de duas horas de carro do laboratório de Boeblingen.

Em 2002 e 2003 mais novidades. Empresas de software como SAP e Oracle anunciaram suporte de seus softwares para rodarem no Linux em mainframes e no fim de 2003 contabilizava-se mais de 250 produtos de software, inclusive o Lotus Notes. A Red Hat também anuncia sua primeira distribuição para mainframes, com o Red Hat Enterprise Linux 3.

O uso do Linux em mainframe crescia rápido e em 2007 a IBM anunciava oficialmente seu projeto Big Green, com objetivo de consolidar seus sistemas internos, que operavam em 3.000 servidores distribuídos, em cerca de 30 mainframes rodando Linux. Sinalizou claramente que se uma empresa global de 100 bilhões de dólares consegue colocar a maioria de seus sistemas internos em Linux, qualquer outra empresa poderá fazer a mesma coisa. Os resultados do Big Green são impressionantes: conseguiu-se reduzir o consumo de energia em 80% e de espaço físico em 85%. Na prática obteve-se duas vezes mais capacidade computacional sem demandar aumento no consumo de energia.

Além disso provou-se que é perfeitamente possível rodar-se grandes sistemas em Linux. O Projeto “Blue Insight”, sistema de BI interno da IBM é um bom exemplo. Consolidou dezenas de BIs departamentais e mais de 60 bases de dados, em um modelo de “Private Cloud Computing”, baseado em Cognos/Linux/mainframe, suportando 1 petabyte ou mil terabytes de dados e atendendo mais de 200.000 usuários. A sua base computacional é um mainframe com 48 processadores, sendo 32 para produção e 16 para desenvolvimentos e testes, capaz de suportar 10.000 transações por segundo. Um maior detalhamento deste projeto pode ser visto em <http://www.clipper.com/research/TCG2009050.pdf>.

Bem, já que falamos Cloud Computing, os mainframes incorporam naturalmente muitos dos atributos que são necessários a uma nuvem, como capacidade escalonável, elasticidade (você pode criar e desligar máquinas virtuais sem necessidade de adquirir hardware), resiliência e segurança. E sem falar em virtualização, que faz parte dos mainframes desde 1967! Acessando

http://www.ibm.com/systems/z/news/announcement/20090915_annnc.html podemos visualizar alguns cases de uso de Cloud Computing em mainframe.

Os dados atuais mostram que portar o Linux para o mainframe foi uma decisão acertada. Hoje mais de 3150 aplicações estão homologadas para rodarem em Linux nos mainframes. Além das distribuições oficiais Red Hat e SUSE, vemos outras distribuições gratuitas (e não oficiais) como Slackware (www.slack390.org) e Debian (<http://www.debian.org/ports/s390/>) também rodando em mainframes.

De 2004 para 2009 o ritmo de crescimento dos MIPS dedicados ao Linux foi de 43% ao ano. Cerca de 16% da base instalada de MIPS em mainframes estão rodando Linux. 70% dos “Top Clients” de mainframes rodam instâncias Linux em suas máquinas. Os clientes que rodam Linux em seus mainframes ultrapassam a marca dos 1300.

Mas, diante de tantos numeros é curioso que ainda vemos percepções errôneas com relação ao mainframe. Uma recente conversa sobre Linux nos mainframes, com um colega meu, CIO de uma grande corporação foi emblemática desta percepção de muitos executivos quanto aos mainframes. Ele, como muitos outros profissionais é da geração formada durante o movimento de downsizing, que consagrou o modelo cliente-servidor e que considerava “politicamente correto” desligar o mainframe. Nesta época, início dos anos 90, todo e qualquer projeto de consultoria recomendava a mesma coisa: “troquem os caríssimos mainframes pelos baratos servidores distribuídos.”. Claro que muitas destas decisões de troca foram acertadas, mas muitas outras se revelaram totalmente inadequadas. O custo de propriedade de ambientes distribuídos se mostrou muito mais caro que se imaginava.

Há tempos fiz um pequeno exercício onde analisei informalmente uma empresa que houvesse desligado o mainframe no início dos anos 90 e quanto gastou em TI com seu ambiente distribuído, considerando todos os fatores que envolvem o TCO. Comparei com a alternativa de manter os sistemas no mainframe (evoluindo com novos modelos e tecnologias) e claro, com um conseqüente uso bem mais restrito de servidores distribuídos. O ambiente 100% distribuído consumiu em 15 anos mais dinheiro que a alternativa de se manter um ambiente mixto.

É verdade que o custo de hardware há 15 anos atrás era destacadamente o maior do orçamento de TI. Hoje não é mais. Mas a percepção quanto ao mainframe continua. Fiquei surpreso em ver o quanto de desconhecimento da evolução dos mainframes ele tinha. E, tenho certeza, não é só ele...Para muitos o mainframe é apenas um velho repositório de aplicações Cobol e PL1, tripulado por profissionais à beira da aposentadoria...Bem, ele se mostrou interessado quando falei que parcela substancial do crescimento de receita da IBM com mainframes vem de novos workloads, principalmente Linux.

Mostrei que no contexto atual, o movimento de consolidação e virtualização dos data centers estão abrindo novas portas para o mainframe. O TCO para se gerenciar um parque de centenas ou milhares de servidores é altíssimo, acrescido agora da variável energia, que vem aumentando ainda mais os budgets das áreas de TI.

O mainframe pode ser usado para consolidar centenas de servidores Linux x86 distribuídos pelos diversos cantos da empresa em uma única máquina. Um recurso que ajuda muito é a tecnologia IFL (Integrated Facility for Linux), que surgiu em setembro de 2000. Os IFL são processadores do mainframe especialmente dedicados a rodar Linux, com ou sem z/VM. São processadores iguais aos demais, mas por microcódigo eles só aceitam rodar workloads Linux. Estima-se que hoje existam mais de 4.600 processadores IFL ativos.

O Linux no mainframe consegue explorar todas as características únicas do hardware como sua reconhecida confiabilidade, disponibilizada por features como processadores redundantes, elevados níveis de detecção e correção de erros e conectividade inter-server de altíssima velocidade. O nível de disponibilidade do ambiente operacional do mainframe é muito maior que dos sistemas distribuídos. E abrir uma máquina virtual Linux em um mainframe leva alguns minutos, enquanto que comprar, instalar e configurar um servidor distribuído pode levar semanas.

Bem, questionado quanto a citar alguns aspectos positivos da consolidação lembrei a ele: menos pontos de falha, eliminação da latência de rede, menos componentes de hardware e software para gerenciar, uso mais eficiente dos recursos computacionais, melhor gestão de cargas mixtas batch e transacionais, maior facilidade de diagnósticos e determinação/correção de erros, recovery/rollback muito mais eficiente... O resultado? Um melhor TCO!

Interessante que uma vez criada uma percepção, torna-se difícil mudar as idéias. Olhar o mainframe sob outra ótica é uma mudança de paradigmas e mudar paradigmas não é fácil. Paradigma é como as pessoas vêem o mundo, ele explica o próprio mundo e o torna mais

compreensível e previsível. Mudar isso exige, antes de mais nada, quebrar percepções arraigadas. Um estudo de TCO pode mostrar que talvez as idéias e hábitos que tem mantido a TI da companhia nos últimos anos não sejam tão mais válidos assim...Resumo da história: em 2020 iremos comemorar os 20 anos do Linux nos mainframes!

Gerando receita com Open Source

Volta e meia dou entrevistas para a mídia falando de Open Source. E uma das perguntas mais frequentes é sobre “Quanto a IBM obtém de receita com Open Source?”. Ora, quando se fala no modelo tradicional de comercialização de softwares, esta pergunta tem uma resposta fácil: basta ver quantas licenças foram comercializadas e qual o preço médio delas. Mas, com Open Source é diferente. É muito difícil capturar com precisão o volume de receitas. Muito da receita de Open Source é obtido de forma indireta. Um exemplo é o Google que fornece gratuitamente softwares como Android e outros, para alavancar receita com anúncios. Vejamos também a IBM, que apoia diversos projetos como o Linux, Eclipse e outros, alavancando receitas indiretas, como mais servidores, serviços e mesmo softwares não Open Source.

Portanto, precisamos reformular a questão. A receita direta e contabilizável de um determinado software Open Source não implica em medida direta do sucesso ou fracasso do seu impacto econômico. Devemos, para esta análise, olhar o ecossistema como um todo. Um engano comum é comparar as receitas de um determinado software comercializado na modalidade de vendas de licenças com a receita obtida pelos distribuidores de softwares Open Source. Ora as distribuições pagas de softwares Open Source, de maneira similar ao SaaS, são comercializados pelo modelo de negócios de assinaturas, com a receita sendo distribuída ao longo de vários anos e não concentrada em um único pagamento. Assim, comparar receitas obtidas por modelos de negócio diferentes é comparar laranjas com bananas.

Além disso, não existe correlação entre a receita direta obtida com determinado software e seu uso pela sociedade. É muito difícil medir com precisão o uso de um software Open Source. Podemos contabilizar os downloads registrados a partir de um determinado site associado ao software em questão. Mas, a partir daí, como é perfeitamente possível e até mesmo incentivada sua livre distribuição, fica difícil contabilizar as inúmeras outras cópias que circularão pela Web.

Entretanto, é indiscutível que Open Source está se disseminando rapidamente. Os seus principais apelos para o mercado são bastante motivadores: não demanda desembolso prévio para licença de uso (troca capex ou custo de capital por opex, que é custo operacional), menor custo de propriedade, evita aprisionamento forçado por parte de fornecedores e maior facilidade de customização, pelo livre acesso ao código fonte. Também observamos que sua disseminação não é homogênea por todos os segmentos de software. Sua utilização é muito mais ampla em sistemas operacionais, web servers e bancos de dados, mas ainda restrita em outros setores, como ERP e business intelligence.

Mas, Open Source não cresce não apenas no campo do uso tradicional de software, que são os aplicativos comerciais. Vemos sua disseminação se acelerando à medida que a Web se dissemina (muito do código que existe rodando na Web 2.0 e redes sociais é baseado em linguagens dinâmicas em Open Source como PHP, Python e Ruby) e veremos muito código Open Source sendo a base de sensores, atuadores, set top boxes da TV digital, netbooks, celulares e outros novos equipamentos. Open Source também está na base tecnológica de muitas infraestruturas de cloud computing.

O termo “Open Source vendor” que era considerada uma contradição no primeiro momento, começa a se popularizar de forma bem rápida. À medida que o mercado Open Source amadurece, fica claro que depender única e exclusivamente de comunidades de desenvolvedores voluntários não atende adequadamente às demandas empresariais. Os usuários corporativos exigem níveis de serviço e suporte que a maioria das comunidades “loosely coupled”, típicas do Open Source, não conseguem atender. Abriu-se espaço para o surgimento de novos negócios, intermediários entre as empresas e as comunidades. O que estes novos negócios (“Open Source vendors”) oferecem é um suporte comercial similar ao dos negócios tradicionais da indústria de software. Algumas destas empresas tornaram-se grandes corporações, como a Red Hat, que em 2008 conseguiu receitas de mais de meio bilhão de dólares.

A cadeia de valor do Open Source é constituída de diversos atores como a comunidade (que contribui com código); os “Open Source vendors” que oferecem suporte de segundo nível, localizam software e desenvolvem e comercializam distribuições e edições dos softwares; Open Source VARs que oferecem suporte de primeiro nível, implementação e treinamento; e finalmente os próprios usuários dos softwares Open Source, que usam o software, identificam bugs e também contribuem com código para a comunidade.

Uma diferença fundamental entre os modelos de comercialização do Open Source e os modelos de venda de licença tradicionais é que no Open Source o processo de aquisição de software é direcionado pelo próprio usuário, que pode fazer download do software e testá-lo, sem intervenção do fornecedor. Após os testes ele pode continuar usando uma versão free ou contratar assinatura da versão comercial, que na verdade é muito mais um contrato de serviços. Na prática, a razão de downloads para assinaturas pagas é de pelo menos 1.000 para um. Ou seja, para cada 1.000 downloads, um contrato de assinatura é assinado. No modelo tradicional este nível de conversão de experimentos para contratos seria simplesmente inaceitável e levaria a empresa de software à falência. No Open Source é perfeitamente válido, pois a maior parte das ações de pré-vendas é feita pelo próprio usuário do software. O usuário é quem bate na porta do vendedor quando está interessado na assinatura do software e não o contrário! Uma característica interessante deste modelo é que a maioria dos “Open Source vendors” tem muito mais profissionais em áreas técnicas que em vendas, enquanto que no modelo de comercialização tradicional, a força de vendas é proporcionalmente muito maior.

Uma consequência da maior disseminação do Open Source é que começamos a ver empresas de software, que até a pouco tempo eram hostis ao movimento já se envolvendo ou no mínimo se tornando neutras em relação ao conceito. E para o futuro? Um insight, sujeito a correção de rumo na sua intensidade e velocidade: as empresas de software que estavam fora do Open Source vão integrar suas soluções com estes softwares. Este movimento deve se acelerar com a rápida disseminação do SaaS. A natureza do modelo de negócios SaaS demanda que as licenças de software a serem pagas a terceiros pelos provedores de soluções SaaS sejam mínimas e o Open Source se encaixa muito bem neste contexto. Muitas destas empresas vão seguir os passos da IBM e se engajar nos projetos de Open Source, inclusive até mesmo colaborando ativamente com a comunidade. É provável

que nos próximos anos o modelo de negócios dominante da indústria seja híbrido, com Open Source e venda de licenças integrados na comercialização das stacks de software.

Open Source e Serviços

Open Source já está deixando de ser notícia de mídia. Não que o assunto tenha se esgotado, mas é que não é mais novidade. Open Source já está no dia a dia das empresas. E um dos negócios em torno do Open Source que está crescendo bastante são exatamente os serviços. Neste modelo, não se ganha dinheiro diretamente com venda de licenças, mas principalmente com serviços. Segundo o IDC, neste ano de 2009 o mercado mundial de serviços em torno do Open Source deverá atingir a casa dos sete bilhões de dólares, um crescimento de 25% sobre o ano anterior. Em 2013, as previsões são que alcancem 12,7 bilhões de dólares! A receita de serviços acumulada entre 2008 e 2013 será de mais de 54 bilhões de dólares. Realmente, estamos falando de muito dinheiro.

O que estes numeros representam? Indiscutivelmente que Open Source já criou um ecossistema forte e saudável e que não tem como ser ignorado.

Open Source já está atingindo a maturidade e as discussões ideológicas estão sendo deixadas de lado. Lembro que em 2004 e 2005 as discussões se centravam na luta entre o bem e o mal, com o mal simbolizado pelo software proprietário. Pouquíssimas discussões eram pragmáticas...Não se falava em mercado, das dificuldades das empresas em adotar Open Source. Muitas iniciativas eram, no âmbito governamental, definidas por decreto, sem uma visão clara e consistente dos desafios a serem enfrentados. Na época, a IBM era a única das grandes empresas de software a acreditar e investir em Open Source. Já em 2001, a IBM anunciava investimentos de um bilhão de dolares em iniciativas ligadas ao Linux. Foi um marco para a indústria de software, sinalizando que Open Source era coisa séria.

Hoje o contexto é diferente. Muitos projetos de software Open Source já estão amadurecidos. O Linux já tem 18 anos, o MySQL 14 anos, o PostgreSQL é de 1996, o Apache surgiu em 1995, o Eclipse em 2001, Mozilla em 1998 e o OpenOffice em 2000. A indústria de software, salvo ainda raras exceções já entendeu que Open Source é irreversível. E muitas empresas de software misturam código Open Source com seu código proprietário, criando soluções híbridas. Estas alternativas abrem espaço para produtos inovadores como os ofertados pela Black Duck Software (<http://www.blackducksoftware.com/>) e OpenLogic (<http://www.openlogic.com/>) que vasculham código e validam se existe alguma violação de patentes.

Também já vemos Open Source entrando em outros setores de software, como um sistema de sistema de trouble ticket, o OTRS (<http://www.otrs.org/>), e sistemas de shopping cart para comércio eletrônico como o osCommerce (<http://www.oscommerce.com/>) ou o Zen Cart (<http://www.zen-cart.com/>). Existem soluções para integração de dados e ETL, como o Talend (<http://www.talend.com/>) e diversos outros softwares.

Mas, voltando aos serviços, o segmento de maior crescimento é exatamente o de integração. Curiosamente, os de menor oportunidade de negócios são as atividades de educação e treinamento. Por outro lado, os treinamentos abrem portas para os serviços de maior valor agregado como consultoria e integração.

OK, e que sugestões podemos fazer para empreendedores que queiram entrar neste setor? Apenas duas: Gerar dinheiro com software Open Source é diferente do modelo proprietário.

Não existem licenças e portanto devem buscar receita em serviços ou obter ganhos indiretos. E não esquecer que integração entre sistemas proprietários e Open Source é a demanda de maior oportunidade. Assim, não tem sentido uma empresa ser especializada 100% em Open Source.

Adicionalmente sugiro prestar atenção à rápida disseminação da chamada Internet das coisas, onde sensores e outros dispositivos estarão atuando em tempo real, integrados à infraestrutura física do planeta. As oportunidades de novos negócios usando tecnologias Open Source serão absolutamente imensas.

Torne-se um pesquisador-cidadão

Imaginem este cenário: você, mesmo sem conhecimentos profundos de medicina e biologia pode participar ativamente de uma pesquisa científica que busque drogas que combatam o câncer, a AIDS ou o dengue. Ou sem profundos conhecimentos de física e química pode participar de uma pesquisa que analisa as propriedades de nanotubos de carbono como base de filtros para gerar água potável. Sim, você pode ser um pesquisador cidadão, simplesmente fazendo parte do World Community Grid ou WCG.

O WCG é um projeto inovador criado pela IBM que usa uma massiva grade (grid) de PCs como base para solucionar complexos problemas computacionais. Este conceito popularizou-se com o sucesso do projeto [SETI@Home](http://setiathome.ssl.berkeley.edu/) (<http://setiathome.ssl.berkeley.edu/>), de busca por vida extraterrena.

Este modelo, chamado de computação filantrópica ou voluntária é baseado na idéia de voluntariado, onde um usuário toma a decisão deliberada de ceder ciclos ociosos de seu PC para contribuir com uma determinada organização a executar uma tarefa, seja pesquisa por vida extraterrena, cura do câncer, pesquisa climatológica ou qualquer outra iniciativa.

A idéia por trás deste projeto é simples. Parte do princípio que em 95% do tempo um PC está ocioso. Imagine aquele tempo em que você está com a máquina ligada e falando no telefone. Ou quando sai para um café ou almoço? O PC está ocioso, com um screensaver que nada faz de útil.

Para iniciar a colaboração, após o usuário se cadastrar como voluntário no site específico, um pequeno programa é transferido (*downloaded*) para o seu PC. Este programa é o responsável pela comunicação via Internet com o servidor central, bem como pela utilização dos ciclos ociosos do PC para execução da tarefa computacional solicitada.

A sua característica principal é que demanda pouca interação com o servidor central, resumindo-se a baixar novos dados ou devolver dados já processados. A maior demanda é pelos ciclos de processador da máquina, que opera a computação independente do servidor central.

A aplicação também não deve interferir com utilização diária do PC e apenas consome ciclos de processador quando o computador estiver inativo. De maneira geral ela opera como um “screen saver”.

Para entendermos melhor como funciona um projeto de “ computação filantrópica” vamos analisar com mais detalhes o World Community Grid (www.worldcommunitygrid.org) .

O WCG foi criado em fins de 2004 e desde então meio milhão de voluntários de quase todos os países do mundo (o Brasil está em 7º lugar no ranking de voluntários mundiais) já contribuíram com o equivalente a 328.000 anos de computação de um PC. Hoje são cerca de 1,5 milhão de PCs que contribuem com seu tempo ocioso para diversas pesquisas e esta capacidade computacional coloca o WCG como um dos Top-10 supercomputadores.

Mas, pensem...1,5 milhão é apenas um milionésimo dos PCs e laptops espalhados pelo mundo. Se imaginarmos multiplicar por dez este numero e chegar a 1% da base instalada mundial teremos um dos maiores supercomputadores do mundo. E se multiplicarmos por cem? Com certeza estaremos falando de uma capacidade equivalente a muitos PetaFLOPS.

Embora a idéia do WCG tenha vindo do SETI@Home, seu projeto se propõe a tratar de pesquisas que afetem mais diretamente a sociedade humana. O Brasil já teve uma pesquisa concluída no WCG, de comparação de genomas, pesquisa esta desenvolvida pela Fiocruz, cujo resultado pode ser visto em http://www.worldcommunitygrid.org/about_us/viewNewsArticle.do?articleId=132. Uma segunda pesquisa brasileira, de busca por drogas que combatam a esquistossomose, está se iniciando agora (<http://www-03.ibm.com/press/us/en/pressrelease/32422.wss>). Uma pesquisa para ser submetida e aceita pelo board do WCG deve preencher alguns requisitos, um deles é que os resultados sejam de domínio público. O formulário de submissão encontra-se em <http://www.worldcommunitygrid.org/research/viewSubmitAProposal.do>. O uso do WCG é inteiramente gratuito.

Uma outra característica é que a aplicação que vai sustentar a pesquisa deve poder ser paralelizada e demanda muita computação. Recomendo a leitura do paper “Large Scale Execution of a Bioinformatic Application on a Volunteer Grid”, que descreve o processo de portar para o WCG um projeto de pesquisa para distrofia muscular, em <http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2007/RR2007-49.pdf> como roteiro a ser seguido por pesquisadores interessados em usar o WCG.

A tecnologia por trás do WCG é a mesma do SETI@Home que é o software Open Source denominado BOINC (Berkeley Open Infrastructure for Network Computing) da Universidade de Berkeley, EUA (<http://boinc.berkeley.edu>). O BOINC é um sistema que permite que um usuário possa operar simultaneamente diversos projetos de computação filantrópica.

Além do WCG, o BOINC é usado como plataforma para diversos outros projetos de computação filantrópica em áreas tão diversas como física, biologia molecular, medicina, astronomia, climatologia, matemática e estudo de jogos. Uma lista destes projetos pode ser encontrada em http://www.boinc-wiki.info/BOINC_Powered_Project.

A arquitetura do BOINC é simples e fortemente inspirada no [SETI@Home](http://setiathome.berkeley.edu). No lado servidor há um banco de dados relacional que armazena as informações referentes a cada projeto, como usuários cadastrados, unidades de trabalho disponíveis, unidades de trabalho enviadas e processadas e assim por diante. Cada projeto também tem um serviço back-end, responsável por distribuir as unidades de trabalho e tratar os resultados recebidos. Os servidores de dados são os responsáveis pela distribuição dos arquivos de dados e seu recebimento. Os serviços de escalonamento controlam o fluxo de entrega das unidades de trabalho aos usuários, conforme a produtividade de cada um. A interface com o usuário é feita via Web.

No lado cliente o BOINC é representado por um código central (o núcleo), que é comum a todos os projetos, com as interfaces com o sistema servidor, e os códigos específicos de cada projeto, como os algoritmos para resolução de problemas.

As características principais do projeto BOINC são:

- a) Permitir a operação de aplicações já existentes, escritas em linguagens de programação comuns, como Fortran, C e C++, com pouca ou nenhuma modificação;
- b) Oferecer um maior grau de segurança e proteção contra vírus, inclusive com uso de assinaturas digitais baseadas em criptografia de chaves públicas;
- c) Permitir o uso de diversos servidores. O programa cliente, ao identificar que um servidor se encontra fora do ar, automaticamente tentará acesso a servidores alternativos;
- d) Implementar ferramentas de monitoração, que visualizarão diversas variáveis como carga de processador e tráfego de rede, simplificando as tarefas de diagnósticos de problemas;
- e) Disponibilizar o código fonte, com distribuição sob licenças públicas. O servidor BOINC é livremente distribuído, apenas com restrições de não poder ser usado em produtos comerciais. Por sua vez, os projetos que rodam sob BOINC não precisam necessariamente ter seu código fonte aberto;
- f) Suportar grandes volumes de dados, inclusive com acesso a múltiplos servidores. Os usuários podem especificar limites de utilização de disco e banda de passagem na rede.

O projeto BOINC, como qualquer projeto de computação filantrópica, opera com o servidor enviando dados aos programas cliente e estes, após executar as computações, retornam os resultados ao servidor. Entretanto, como projetos como o [SETI@Home](#) e os desenvolvidos pelo WCG identificaram, diversas situações podem ocorrer, como:

- a) O programa cliente executa corretamente a computação e retorna os resultados;
- b) O programa cliente computa erroneamente seus algoritmos e envia resultados errados ao servidor;
- c) O usuário não consegue estabelecer conexão com o servidor e portanto não consegue executar downloads ou uploads;
- d) A aplicação não funciona na máquina do usuário;
- e) O usuário nunca retorna o resultado de sua computação, inclusive por desistir do processo no meio do caminho.

As questões de segurança, uma vez que vários projetos podem estar executando na mesma máquina, mereceram atenção. Para impedir falsificação de resultados, o BOINC utiliza redundância para diminuir a chance e ocorrências. Cada unidade de trabalho é distribuída para múltiplos clientes, os resultados gerados por eles são comparados e são aceitos apenas aqueles nos quais o consenso é obtido. Eventualmente, novos clientes são acionados para executar as mesmas computações, quando não for alcançado o consenso com as computações que tenham chegado ao servidor.

O BOINC gerencia a maior parte das tarefas de redundância, embora a aplicação deva se envolver com os processos de validação e assimilação. Validação acontece quando um número suficiente de respostas chega ao servidor (foi alcançado um número mínimo de respostas, considerado como quorum mínimo) e deve ser avaliado se existe consenso. O método de comparar resultados e a política para determinar se há ou não consenso deve ser fornecido pela aplicação no servidor.

Assimilação é o mecanismo no qual o projeto é notificado do término, com sucesso ou não, de uma unidade de trabalho. Quando é obtido consenso, os resultados são enviados para processamento complementar pela aplicação no servidor. Se a unidade de trabalho não conseguir ser executada com sucesso, por exemplo, por não haver quorum ou não ter sido obtido consenso, a aplicação no servidor é acionada para determinar ações alternativas.

Para eliminar distribuição forjada de aplicações ou a possibilidade de um cracker distribuir um código cliente falso (um vírus) no lugar de código válido, o BOINC utiliza assinatura de código. Cada projeto possui um par de chaves criptográficas que são utilizadas para autenticar os programas que distribui.

O ambiente de computação voluntária apresenta características especiais. Uma delas é que a disponibilidade das máquinas é extremamente dinâmica, com usuários entrando e saindo sem aviso prévio, e por isso o BOINC implementa uma API para função de checkpoint, que permite que o estado de execução da aplicação seja salvo e retomado mais adiante, no mesmo ponto em que foi interrompido. A aplicação deve estar ciente dos momentos do checkpoint. Isto significa que ela deve conter código que indique explicitamente os pontos no qual o estado de execução deva ser salvo. Também é de responsabilidade da aplicação decidir o que deve ser salvo para permitir retomada posterior.

BOINC foi projetado para suportar aplicações que demandam grande capacidade computacional e que possam atrair um grande volume de voluntários. Entretanto, como a computação filantrópica depende de conexão via Internet, com usuários voluntários, é importante que os seguintes critérios sejam satisfeitos, para que a aplicação tenha sucesso:

- a) Apelo público: a aplicação deve atrair um grande número de voluntários. Para isso deve ser um projeto de apelo popular, que motive o voluntariado e implemente interfaces amigáveis, de fácil utilização por usuários não técnicos;
- b) Paralelismo independente: a aplicação deve ser divisível em partes que podem ser operadas em paralelo, com pouca ou nenhuma dependência entre elas;
- c) Razão dados/computação deve ser baixa. A conexão é efetuada pela Web, muitas vezes com usuários com modems de baixa velocidade. O tráfego de dados pela rede não pode ser elevado. Uma transferência demorada desestimula o voluntário, que está pagando pelo uso das linhas de acesso;

d) Tolerâncias à falhas. Os resultados obtidos de computadores de usuário não conhecidos, típicos da computação filantrópica, não podem ser considerados sempre corretos. Devem ser usados mecanismos de redundância para reduzir a probabilidade de erros. Entretanto se a aplicação exigir correção de 100% o não devemos considerar o uso de BOINC e do modelo de computação filantrópica.

O BOINC pode ser usado para criar plataformas de pesquisa. Os pesquisadores tem como opção submeter uma proposta ao World Community Grid, que já conta com um volume muito grande de voluntários ativos ou criar seu proprio ambiente de pesquisas. Eventualmente as universidades brasileiras podem pensar em criar uma plataforma BOINC que sirva de base para seus diversos projetos específicos. O poder computacional de um único PC é insuficiente para rodar complexos algoritmos que fazem parte da maioria das pesquisas acadêmicas, mas se juntarmos as centenas de milhares que estão nos laboratórios das universidades e nas casas dos seus alunos podem gerar esta capacidade computacional. Com certeza a imensa maioria das universidades brasileiras não tem budget para comprar um caríssimo supercomputador mas se juntarem seus esforços verão que tem um parque de PCs que podem simular este supercomputador, desde que estejam trabalhando em grade, em projetos baseados no BOINC.

Bibliografia Recomendada

Software Livre: Potencialidades e Modelos de Negócio, de Cezar Taurion, editora Brasport.

A Revolução do Software Livre, coletânea de textos de diversos profissionais que atuam em Open Source, editado pela Comunidade Sol (www.comunidadesol.org).

The Success of Open Source, de Steven Weber, Harvard University Press

Open Sources 2.0, de Chris Bona, Danese Cooper e Mark Stone, O'Reilly

The Business and Economics of Linux and Open Source, Martin Fink, Prentice Hall.

Producing Open Source Software: How to Run a Successful Free Software Project, Karl Fogel, O'Reilly

Understanding Open Source and Free Software Licensing, Andrew St. Laurent, O'Reilly

Só Por Prazer - Linux - Bastidores Da Sua Criação, Linus Torvalds, Campus

Autor

Cezar Taurion

Gerente de Novas Tecnologias Aplicadas/Technical Evangelist da IBM Brasil, é um profissional e estudioso de Tecnologia da Informação desde fins da década de 70. Com educação formal diversificada, em Economia, Ciência da Computação e Marketing de Serviços, e experiência profissional moldada pela passagem em empresas de porte mundial, Taurion tem participado ativamente de casos reais das mais diversas características e complexidades tanto no Brasil como no exterior, sempre buscando compreender e avaliar os impactos das inovações tecnológicas nas organizações e em seus processos de negócio.

Escreve constantemente sobre tecnologia da informação em publicações especializadas, além de apresentar palestras em eventos e conferências de renome. É autor de cinco livros que abordam assuntos como Open Source/Software Livre, Grid Computing, Software Embarcado e Cloud Computing, editados pela Brasport e já teve publicado um outro blogbook pela SingularDigital, o Inovação!

Cezar Taurion também mantém um dos blogs mais acessados da comunidade developerWorks (www.ibm.com/developerworks/blogs/page/ctaurion). Este blog, foi, inclusive o primeiro blog da developerWorks na América Latina. Mantém adicionalmente um blog sobre Cloud Computing em www.computingonclouds.wordpress.com.

Para contatos com o autor use ctaurion@br.ibm.com ou ctaurion@gmail.com. Também pode ser acessado no Facebook, LinkedIn e pelo Twitter em @ctaurion.