Vantagens Estratégicas do Software Livre para o Ambiente Corporativo

Nelson Corrêa de Toledo Ferraz

Orientador: Geraldo Coen

Monografia apresentada para a conclusão de curso
Master Business Information Systems

MBIS / PUC-SP

São Paulo / 2002

1.Introdução

A cada ano que passa, o software livre ganha mais força, movido pela paixão de milhares de desenvolvedores e milhões de usuários que acreditam em uma ética segundo a qual o conhecimento não deve permanecer oculto, mas ser compartilhado.

Este trabalho tem como objetivo mostrar, através de estudos de casos, que o software livre é uma fonte de vantagem estratégica para as empresas, trazendo benefícios práticos, independentemente de posições filosóficas ou princípios morais.

O trabalho começa apresentando alguns conceitos fundamentais relacionados ao tema, conta um pouco da história do software, e passa a apresentar os casos que servirão para demonstrar que as vantagens estratégicas oferecidas pelo software livre são um padrão constante, e não apenas casos isolados. Tentaremos também rebater algumas críticas, que sob a luz dos casos apresentados se revelarão como simples mitos[1].

1 Ou parte de uma estratégia para disseminar medo, incerteza e dúvida (*Fear, Uncertainity, Doubt* – FUD): http://www.geocities.com/SiliconValley/Hills/9267/fuddef.html

Concluiremos com uma visão do que ainda precisa ser feito e como a comunidade deve direcionar seus esforços para que o software livre torne-se uma opção cada vez mais viável para o ambiente corporativo.

Este trabalho está disponível sob a Licença de Documentação Livre GNU (GNU FDL)[2], oferecendo liberdade de leitura e distribuição, em qualquer meio, com ou sem modificações. Com isso temos o objetivo de disseminar a informação - que vale mais quando é compartilhada, e não quando permanece restrita.

Caso você encontre qualquer imprecisão, ou queira incluir mais informações sobre qualquer assunto abordado, envie seus comentários para a lista de discussão[3].

História

A Free Software Foundation define software livre como aquele “que vem com permissão para qualquer um copiar, usar e distribuir, com ou sem modificações, gratuitamente ou por um preço. Em particular, isso significa que o código-fonte deve estar disponível.”[4].

2 GNU Free Documentation License
http://www.gnu.org/licenses/fdl.html
3 http://listas.softwarelivre.org/cgi-bin/mailman/listinfo/estrategia
4 Categorias de Softwares Livres e Não-Livres
http://www.gnu.org/philosophy/categories.pt.html

Curiosamente houve uma época em que essas características foram a regra, e não a exceção. Entre os anos 50 e 60, por exemplo, praticamente ninguém considerava a hipótese de "vender" softwares. O motivo era simples: haviam tão poucos computadores no mundo, que o valor real estava na máquina em si, e não nos programas, que eram distribuídos gratuitamente, melhorados e compartilhados.

Essa foi a época dos computadores de milhões de dólares, que ocupavam salas inteiras, dos cartões perfurados, e dos primeiros hackers[5].

A década de 70 viu o surgimento de diversas tecnologias que formam a base da chamada Era da Informação: o microprocessador, a fibra ótica, o protocolo TCP/IP e, finalmente, o microcomputador[6].

O microcomputador, em especial, permitiu que pessoas comuns, entusiastas de microinformática (ou "hobbistas"), fizessem o que antes só era permitido a poucos: criar, modificar e trocar programas entre si.

No entanto, esse método de desenvolvimento, baseado na liberdade e na cooperação, não foi suficiente para atender toda a demanda que surgiu com a massificação da informática, até mesmo por que o contato entre estas pessoas era

5 Para duas definições do termo "hacker", consulte:
http://www.tuxedo.org/~esr/faqs/hacker-howto.html#what_is
http://www.tuxedo.org/~esr/jargon/html/entry/hacker.html

6 Castells, Manuel. "A Sociedade em Rede", pp. 64

dificultado por barreiras físicas e geográficas – limitações estas que a internet resolveu, muitos anos depois.

De qualquer forma, é importante dizer que o modelo comercial surgiu para preencher uma legítima necessidade do mercado e que uma das primeiras pessoas a perceber o valor deste mercado foi um jovem chamado Bill Gates, que em 1975 desenvolveu um interpretador BASIC para o Altair, um micro com apenas 4kb de memória.

Diferentemente de outros programas disponíveis na época, o Altair BASIC estava disponível apenas para venda, sem código-fonte. Para desgosto de seu criador, porém, a comunidade de programadores continuou trocando cópias desse programa entre si.

Gates considerou isso um "roubo", e expressou sua insatisfação em uma carta aberta aos hobbistas, escrita em 1976:

> ...As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid?
>
> Is this fair? One thing you don't do by stealing software is get back at MITS for some problem you may have had. MITS doesn't make money selling software. The royalty paid to us, the manual, the tape and the overhead make it a break-even operation. One thing you do do is prevent good software from being written. Who can afford to do professional work for nothing? What hobbyist can put 3-man years into programming, finding all bugs, documenting his product and distribute for free? The fact is, no one besides us has invested a lot of money in

hobby software.[7]

Na visão de Gates, usuários jamais conseguiriam softwares e documentação de qualidade sem um incentivo financeiro, e portanto os hobbistas não deveriam compartilhar, e sim comprar softwares.

Mas uma pessoa não concordava com esta visão: Richard Stallman iniciou uma luta pessoal, no início dos anos 80, contra as licenças que, cada vez mais, restringiam a liberdade dos usuários. Para ele, a arte da programação havia se transformado em um negócio lucrativo, porém imoral.

Stallman abandonou o MIT em 1983, com o objetivo específico de criar um sistema operacional completo e distribuí-lo para quem quisesse, gratuitamente. Naquele mesmo ano ele publicou o GNU Manifesto, onde declarava seus princípios:

> I consider that the golden rule requires that if I like a program I must share it with other people who like it. I cannot in good conscience sign a nondisclosure agreement or a software license agreement.
>
> So that I can continue to use computers without violating my principles, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free.[8]

Hoje, o sonho de Richard Stallman, de um sistema operacional 100% livre, está praticamente completo. O *kernel*, ou núcleo do sistema operacional, que era uma das principais peças

7 An Open Letter to Hobbyists by William Henry Gates III
http://www.blinkenlights.com/classiccmp/gateswhine.html

8 The GNU Manifesto
http://www.gnu.org/gnu/manifesto.html

que faltavam ao projeto GNU, vem sendo aprimorado desde 1991, por Linus Torvalds e uma equipe de desenvolvedores espalhados pelo mundo.

A cada ano o software livre ganha novos adeptos e mais força. A maioria dos servidores web do mundo rodam em Apache, a maior parte dos e-mails transmitidos passam pelo sendmail, e o GNU/Linux é o sistema operacional com maior crescimento no servidor.

Olhando para o passado, nós vemos que as liberdades defendidas por Stallman nada mais são do que direitos que por muito tempo havíamos esquecido, e que durante este período a indústria de softwares conseguiu prosperar erguendo barreiras artificiais, dividindo e conquistando usuários, duplicando esforços e criando produtos de baixa qualidade, que causam perdas de bilhões de dólares todos os anos[9].

9 O custo da baixa qualidade dos softwares será discutida em diversos capítulos deste trabalho

2.Conceitos fundamentais

Software Livre

Existem duas expressões em inglês para designar o que chamamos, nesse trabalho, de software livre: *Free Software* e *Open Source Software*.

O termo *Free Software* costuma causar alguma confusão em inglês, pois a palavra *Free* é frequentemente associada a "preço igual a zero", ou "grátis". Por causa desta confusão, é comum encontrar o seguinte comentário em textos que falam sobre software livre em inglês:

> 'Free software' is a matter of liberty, not price. To understand the concept, you should think of 'free' as in 'free speech', not as in 'free beer'.[10]

A *Free Software Foundation* define que um software pode ser considerado "livre" quando oferece quatro liberdades fundamentais:

- The freedom to run the program, for any purpose (freedom 0).

10 The Free Software Definition http://www.gnu.org/philosophy/free-sw.html

> - The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
> - The freedom to redistribute copies so you can help your neighbor (freedom 2).
> - The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. (freedom 3). Access to the source code is a precondition for this.[11]

A Open Source Initiative criou o termo *Open Source* com o objetivo de eliminar a ambiguidade entre liberdade e preço, e encorajar o uso do software livre no ambiente corporativo.

Infelizmente o termo *Open Source* também é ambíguo, pois não basta um programa ter o código fonte aberto para ser considerado *Open Source*[12].

Bruce Perens publicou uma série de requisitos para que um software possa ser classificado como *Open Source*, entre os quais podemos destacar:

- A licença não deve restringir a redistribuição do programa e de trabalhos derivados;
- O código-fonte deve estar disponível junto com o programa, ou deve estar disponível para download gratuitamente;
- A licença não deve discriminar pessoas ou grupos, nem restringir a forma de uso do programa (por exemplo, não deve restringir o uso comercial)[13].

Embora a expressão “software livre” seja considerada ambígua em inglês, ela é corretamente entendida em português e

11 Idem

12 Isto será melhor explicado no próximo item, em que falamos sobre os diversos tipos de softwares proprietários.

13 The Open Source Definition: http://www.opensource.org/docs/definition.html

em diversos outros idiomas, como o espanhol e o francês[14], de forma que usaremos esta expressão, e não "código aberto", para designar os programas que oferecem liberdade de uso, modificação e distribuição.

Software Proprietário

Softwares proprietários são aqueles que não oferecem qualquer uma das três liberdades fundamentais (uso, modificação ou distribuição).

Muitos softwares disponíveis comercialmente são proprietários, pois impõem restrições ao uso, distribuição e modificação; mas isso não significa que softwares livres não possam ser comercializados[15]. Portanto, é conveniente não dizer "software comercial" quando se quer dizer "proprietário".

Por outro lado, nem todo programa disponível gratuitamente é um software livre, pois nem sempre o código-fonte está disponível, e às vezes existem restrições para o uso (por exemplo, comercial ou governamental). Por essa razão, não se deve dizer "software gratuito" quando se quer dizer "software livre".

Finalmente, mesmo um programa com código-fonte aberto pode ser um software proprietário. Um bom exemplo disso é a iniciativa *shared source*, da Microsoft:

14 *Software Libre* e *Logiciel Libre*, respectivamente

15 As "distribuições Linux", que veremos a seguir, são exemplos de softwares livres que também são comercializados

> The Shared Source initiative is a balanced approach that allows Microsoft to share source code with various communities while maintaining the intellectual property rights needed to support a strong software business.[16]

É importante lembrar que as principais licenças de software livre têm como objetivo manter a propriedade intelectual dos autores originais, sem que, para isso, seja preciso restringir os direitos dos usuários. No caso da licença *shared source*, porém, "manter a propriedade intelectual" significa que as instituições selecionadas para receber esse código-fonte não têm liberdade para usá-lo, modificá-lo ou distribuí-lo, e, portanto, *shared source* não é software livre. Como já dissemos, esta é uma das razões pelas quais preferimos adotar a expressão software livre, ao invés de código aberto, ou *open source*.

Vantagens Estratégicas

Vantagem estratégica, segundo Porter, é toda aquela vantagem que possa reduzir o poder de um fornecedor, reduzir custos, diferenciar seus produtos ou serviços com relação à concorrência, ou oferecer maior segurança e confiabilidade na execução de processos.

16 The Microsoft Shared Source Philosophy
http://www.microsoft.com/licensing/sharedsource/philosophy.asp

Licenças

A maioria dos softwares são distribuídos com licenças de uso (*End User License Agreements*, ou EULAs), de forma que o usuário não "compra" um software, mas o "licencia".

Na maioria dos softwares proprietários o objetivo da EULA é restringir os direitos do usuário e proteger o fabricante do software.

Em geral, o programa pode ser usado em um número limitado de computadores, por um número limitado de usuários, não pode ser modificado ou redistribuído.

Algumas EULAs chegam a proibir a engenharia reversa, o que significa proibir a tentativa de aprender como o programa funciona.

GPL e outras licenças de softwares livres

Ao contrário das licenças de softwares proprietários, a licença criada e promovida pela *Free Software Foundation*, a *General Public License*[17] (GPL), não tem como objetivo restringir os direitos do usuário: na realidade, a GPL procura garantir os direitos de uso, cópia e modificação de um programa, para todos os usuários. O preâmbulo da GPL explica como isso é feito:

> ...Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free

17 General Public License: http://www.gnu.org/copyleft/gpl.html

> software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.
>
> To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.
>
> For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.[18]

Em outras palavras, a licença GPL procura oferecer ao usuário as quatro liberdades fundamentais do software livre, ao mesmo tempo em que impõe a obrigação de que essas liberdades sejam mantidas.

Existem outras licenças de software livre menos restritivas, o que significa dizer que elas não procuram garantir a liberdade de uso de programas derivados.

Um exemplo de licença menos restritiva é a licença do FreeBSD[19], que permite a redistribuição sem o código-fonte, efetivamente permitindo que um usuário (ou corporação) utilize partes do software (ou o software completo) em um produto proprietário.

18 Idem

19 O FreeBSD é um sistema operacional livre, derivado do BSD UNIX. http://www.freebsd.org/

Como de fato ocorreu: a Microsoft utilizou partes do código do FreeBSD em seus produtos, como mostrou o Wall Street Journal:

> ...[S]everal FreeBSD volunteers combing through Microsoft products, including the new Windows 2000 operating system, found numerous instances where Microsoft had made use of their software – something perfectly legal for it to do[20].

É interessante notar que a Microsoft não se opõe a licenças desse tipo, mas somente àquelas que procuram garantir que um software e seus derivados permaneçam livres. Possivelmente por esta razão, Bill Gates afirmou que a licença GPL teria uma "natureza Pac-man", enquanto Steve Ballmer classificou-a como um "câncer"[21].

GNU/Linux

GNU[22] é o nome de um sistema operacional iniciado por Richard Stallman nos anos 80, parte fundamental do que muitos conhecem simplesmente como "Linux".

Linux é o *kernel* (ou núcleo) atualmente utilizado pelo sistema operacional GNU, e foi iniciado por Linus Torvalds, então estudante da Universidade de Helsinki, na Finlândia, em 1991.

---

20 Wall Street Journal: Microsoft uses Open Source code http://public.wsj.com/news/hmc/sb992819157437237260.htm

21 ZDNet UK: Gates wades into open source debate http://news.zdnet.co.uk/story/0,,t269-s2089446,00.html

22 O nome GNU é uma sigla recursiva, que significa "GNU is Not Unix". Curiosamente, esta sigla não explica o que é GNU, mas somente o que não é.

A distinção entre kernel e sistema operacional é muito sutil, como mostra o seguinte texto da Free Software Foundation:

> An operating system, as we use the term, means a collection of programs that are sufficient to use the computer to do a wide variety of jobs. A general purpose operating system, to be complete, ought to handle all the jobs that many users may want to do.
>
> The kernel is one of the programs in an operating system--the program that allocates the machine' s resources to the other programs that are running. The kernel also takes care of starting and stopping other programs.
>
> To confuse matters, some people use the term "operating system" to mean "kernel". Both uses of the term seem to go back many years. The use of "operating system" to mean "kernel" is found in a number of textbooks on system design, going back to the 80s.[23]

Neste trabalho utilizaremos a expressão “GNU/Linux” quando estivermos falando sobre o conjunto formado pelo kernel e programas do sistema operacional GNU, e “Linux” quando estivermos falando apenas do kernel.

As “distribuições Linux”, em contrapartida, são conjuntos de softwares livres distribuídos comercialmente, ao redor do GNU/Linux. Cada distribuição engloba, além do sistema operacional, dezenas, centenas ou milhares de outros programas[24].

23 GNU/Linux FAQ: http://www.gnu.org/gnu/gnu-linux-faq.html

24 Existem milhares de softwares livres amplamente utilizados em todo o mundo: servidores web (como o Apache), bases de dados (MySQL, PostgreSQL), ambientes gráficos (KDE, Gnome), navegadores (Mozilla, Galeon, Konqueror), aplicativos para escritório (OpenOffice, KOffice, GnomeOffice), entre muitas outras categorias.

Utilizaremos a expressão “distribuições Linux”, e não “distribuições GNU/Linux”, por este ser o nome efetivamente adotado pela maioria das distribuições[25].

É importante lembrar que o projeto GNU compõe uma parte significativa de qualquer distribuição Linux.

25 Naturalmente existem exceções, como a distribuição Debian que se auto-denomina GNU/Linux http://www.debian.org/

3.Diferenciação

De acordo com Porter, “uma empresa diferencia-se da concorrência quando oferece alguma coisa singular valiosa para os compradores”[26].

A liberdade de uso, modificação e distribuição favorecem a inovação. O software livre permite que pessoas ou empresas retomem o trabalho a partir do ponto onde outra pessoa ou empresa parou, ou continue o trabalho de maneiras imprevistas:

> Open Source is the best way of leveraging outside talent. (...) The project could expand to become far bigger than it would have been at a single company. Outside resources make for a cheaper, more complete, and more balanced system, but there' sthis flip side: The expanded system no longer takes only the company needs into account. It actually might consider the needs of customers[27].

Nas próximas páginas nós veremos como a diferenciação é facilitada pelo software livre, e estudaremos alguns casos que demonstram como a liberdade de modificação se traduz em vantagens para todos os usuários, através do aumento da

26 Porter (2), pp. 111
27 Linus Torvalds: Just for Fun – The Story of an Accidental Revolutionary, pg. 232

segurança, do desempenho e da escalabilidade do software, reduzindo os custos e elevando o desempenho dos sistemas.

O software como forma de conhecimento

Uma das maneiras de se entender por que o software livre possibilita a diferenciação é estudar o processo de criação de conhecimento.

De acordo com Nonaka, o conhecimento pode ser dividido em duas categorias: o conhecimento tácito, que é particular, de difícil reprodução; e o conhecimento explícito, codificado, que pode ser facilmente reproduzido.

O ato de programar pode ser entendido como a conversão de um conhecimento tácito, em um conhecimento explícito: a partir do know-how e da experiência de um programador (particulares e dificilmente reproduzidos) é criado o código-fonte de um software (codificado e de fácil reprodução).

O código-fonte de um software é uma forma de conhecimento explícito, da mesma forma como um livro, artigo, ou uma receita de bolo.

Propriedade intelectual

De acordo com Lawrence Lessig[28], as leis de propriedade intelectual foram criadas para estimular a difusão do conhecimento, e não simplesmente para recompensar os autores.

Portanto, os direitos sobre as obras deveriam valer pelo menor período possível, o mínimo necessário para motivar a publicação do conhecimento.

As leis de copyright criadas em 1710 previam um período de 14 anos após a publicação da obra, durante os quais o autor teria direitos de reprodução exclusivos. Ao final deste período a obra cairia em domínio público, permitindo a completa disseminação do conhecimento.

No entanto, o período desta proteção tem sido constantemente expandido, especialmente durante o século XX, quando os interesses de grandes corporações detentoras de propriedade intelectual entraram em jogo.

Somente nos últimos 40 anos o período do monopólio foi modificado 11 vezes: hoje, a duração é de 70 anos após a morte do autor, ou 95 anos para corporações.

> Eleven times in the last 40 years it has been extended for existing works--not just for new works that are going to be created, but existing works. The most recent is the Sonny Bono copyright term extension act. Those of us who love it know it as the Mickey Mouse protection act, which of course [means] every time Mickey is about to

28 Lessig, Lawrence. *The Future of Ideas: The Fate of the Commons in a Connected World.*

> pass through the public domain, copyright terms are extended. The meaning of this pattern is absolutely clear to those who pay to produce it. The meaning is: No one can do to the Disney Corporation what Walt Disney did to the Brothers Grimm. That though we had a culture where people could take and build upon what went before, that' sover. There is no such thing as the public domain in the minds of those who have produced these 11 extensions these last 40 years because now culture is owned.[29]

A espiral do conhecimento

A espiral do conhecimento, de acordo com Nonaka, é o processo pelo qual o conhecimento é criado.

A transformação de conhecimento explícito em tácito é onde começa a espiral do conhecimento. Esta é uma atividade que exige empenho pessoal, e existem inúmeras razões que podem levar uma pessoa a realizar esta articulação: uma necessidade de expressão, a crença em um ideal, ou uma recompensa financeira.

Tanto o modelo proprietário quanto o software livre dependem fortemente da motivação financeira, mas o movimento do software livre também depende, em grande parte, da curiosidade e interesse pessoal dos desenvolvedores[30].

29 Lawrence Lessig: Free Culture
http://www.eff.org/IP/freeculture/

30 De acordo com Eric S. Raymond: "Every good work of software starts by scratching a developer's personal itch".
http://www.free-soft.org/literature/papers/esr/cathedral-bazaar/cathedral-bazaar-2.html

A duplicação do conhecimento explícito, em contrapartida, é um processo simples, e vem tornando-se cada vez mais simples, com a internet.

Como vimos, as leis de propriedade intelectual procuram restringir a duplicação com o objetivo de estimular a divulgação de um conhecimento.

A indústria de software proprietário, porém, utiliza estas leis em seu benefício sem ao menos divulgar o conhecimento: elas publicam o código binário e ocultam o código-fonte. O código binário não representa conhecimento tácito, pois é codificado, mas também não é conhecimento explícito, pois não se encontra na forma adequada para ser internalizado[31].

A proteção intelectual é usada, portanto, como um mero instrumento para permitir a recompensa financeira, o que é uma deturpação da lei; embora esta forma de agir não seja considerada ilegal, ela não beneficia a sociedade, este sim o objetivo original da lei.

Embora a ocultação do código-fonte tenha beneficiado as empresas de software proprietário durante anos, também trouxe prejuízos ao não permitir as duas outras etapas da espiral do conhecimento, a combinação e a internalização.

31 De acordo com a OSI, "the source code must be the preferred form in which a programmer would modify the program". (http://www.opensource.org/docs/definition_plain.php)

A combinação, que consiste no ato de associar múltiplas fontes de conhecimento, obtendo um resultado superior à soma de suas partes.

O software livre facilita a combinação, pois uma pessoa ou empresa pode utilizar o código-fonte de diversos projetos, e qualquer melhoria pode retornar para a comunidade[32].

O modelo proprietário procura facilitar algum tipo de combinação através da componentização, mas ainda assim o código-fonte dos componentes permanece oculto, impedindo a internalização do conhecimento[33].

A internalização é a última etapa da espiral do conhecimento, na qual o conhecimento explícito transforma-se novamente em conhecimento tácito, o que depende, mais uma vez do empenho pessoal[34].

O software livre permite que os programadores absorvam o conhecimento expresso no código-fonte de um programa, e podem em seguida contribuir para a melhoria do mesmo.

32 A licença GPL, em especial, tem como objetivo evitar que o conhecimento não seja devolvido à comunidade.

33 É interessante notar como empresas de software proprietário transformaram uma excelente idéia de organização de projetos de software, em mais uma forma de disponibilizar softwares sem transmitir o conhecimento. No capítulo sobre segurança, mostraremos através de um exemplo real que o uso de componentes sem acesso ao código-fonte pode representar um grande risco para a empresa.

34 Por este motivo, a maioria das pessoas não se preocupam em absorver o conhecimento expresso em um software, deixando esta tarefa para os programadores.

A internalização do conhecimento permite fechar a espiral do conhecimento, ou melhor, reiniciá-la em um nível ainda mais elevado.

Sob esta ótica percebemos que, ao garantir o direito de duplicar e modificar o código-fonte, o software livre facilita a diferenciação e o processo de criação de conhecimento, o que sempre foi dificultado pelo modelo proprietário.

Estudo de casos

SE Linux

O Security Enhanced ou SE Linux é um projeto da Agência de Segurança Nacional dos Estados Unidos, a NSA, visando especificamente o aspecto da segurança do sistema operacional.

Nós estudaremos o aspecto de segurança com mais detalhes no próximo capítulo; por enquanto, nosso foco estará no fato de que a liberdade de uso, modificação, e distribuição do software livre, possibilitou que este estudo fosse realizado, beneficiando todos os usuários para os quais a segurança é um aspecto fundamental.

> Linux was chosen as the platform for this work because its growing success and open development environment provided an opportunity to demonstrate that this functionality can be successful in a mainstream

> operating system and, at the same time, contribute to the security of a widely used system. Additionally, the integration of these security research results into Linux may encourage additional operating system security research that may lead to additional improvement in system security[35].

Como o SE Linux foi disponibilizado sob a licença GPL, o sistema criado por uma agência do governo dos Estados Unidos beneficia empresas e usuários de todo o mundo, o que gerou críticas de empresas norte-americanas[36].

Preemptible Linux

Um mercado em expansão, atualmente, é o de sistemas embutidos em outros equipamentos diferentes de computadores: desde uma televisão até um carro.

Em uma televisão, o tempo de responsividade de um sistema operacional comum (décimos ou centésimos de segundo) pode não ser um problema, mas imagine o sistema de freios de um carro ficar bloqueado por que existem diversas tarefas na fila esperando para serem executadas!

Por esse motivo, um sistema operacional para equipamentos embutidos exige alta responsividade, o que significa um baixo período de latência entre processos[37].

---

35 NSA: Security-Enhanced Linux
http://www.nsa.gov/selinux/

36 News.com: Linux makes a run for government (16/05/2002)
http://news.com.com/2100-1001-950083.html

37 Um processo é uma atividade executada pelo microprocessador. Em sistemas operacionais convencionais, o sistema fica bloqueado enquanto um processo não é concluído.

O Preemptible kernel é um *patch* que possibilita ao Linux realizar o processamento em tempo real, por meio da diminuição do período de latência. Curiosamente, esta mudança também oferece vantagens para usuários domésticos:

> [Kevin Morgan, MontaVista' s Vice President of Engineering] points out that the enhanced kernel responsiveness is not merely useful for esoteric real-time industrial process control systems, but rather is of value even to today' sdesktop PC users. "This technology has use in all segments of the market, including the desktop, for the simple reason that desktops these days require multi-programming and multi-processing, with live streaming media in action," says Morgan. "For example, as a desktop user I want to be able to watch a movie and hear the sound, while also running a browser and my mail program. And when I use the mail program and the browser, I don' twant any glitches in the movie or sound. That really requires improvements in Linux responsiveness, and that kind of behavior really is a real-time problem, because a human is really pretty good at detecting glitches in continuous real-time operation. So, this technology is applicable on the desktop. And, of course, it' s highly applicable in the more traditional embedded control environments."[38]

A disponibilidade do código-fonte do Linux, e a possibilidade de modificá-lo, permitiu que uma característica essencial para um campo de aplicação fosse implementada. Como essa característica também é desejável para o usuário comum, ela foi incorporada à última versão do kernel oficial.

38 MontaVista unveils fully preemptable Linux kernel prototype http://www.linuxdevices.com/news/NS7572420206.html

User-mode Linux

O projeto User-mode Linux[39] (UML) é um patch do sistema operacional GNU/Linux que permite a execução de máquinas virtuais Linux sobre o Linux.

Isto significa que é possível rodar múltiplas cópias do Linux em um único computador, e cada uma delas permanece isolada das demais: em caso de problemas, como um travamento, apenas uma máquina virtual é afetada, e todo o sistema continua estável.

A aplicação inicial do UML consistia numa ferramenta para o desenvolvimento do próprio kernel do Linux, mas como diversos projetos de softwares livres, logo surgiram novas aplicações[40]:

- Hosting virtual: serviços de hosting podem dividir um único computador para diversos clientes, com a segurança de que a utilização de cada um não prejudicará o serviço para os demais.
- Educação: cursos sobre sistemas operacionais, administração de redes, e administração de sistemas.
- Segurança: os processos executados no UML não têm acesso aos recursos externos à máquina virtual,

39 User-mode Linux: http://user-mode-linux.sourceforge.net
40 User-mode Linux: what are people using it for? http://user-mode-linux.sourceforge.net/uses.html

tornando-a um recipiente seguro[41] para programas potencialmente perigosos.

- Ambiente de testes: as máquinas virtuais UML são um ambiente adequado para realizar testes de reinicialização e recuperação de desastres.

O projeto User-mode Linux pode trazer um grande diferencial para empresas que atuam em mercados específicos, como os serviços de hosting, educação e segurança.

Clusters

Como nós vimos, o Preemptible kernel permite ao Linux ser o sistema operacional de equipamentos portáteis, eletrodomésticos e outros tipos de dispositivos. Veremos agora que o Linux não se adapta bem apenas a tarefas leves, mas também ao processamento intensivo, normalmente realizado por supercomputadores.

Beowulf

Em 1994, um grupo de pesquisadores ligados à Nasa estudava a aplicação de computação paralela em análises terrestres e espaciais. Em geral estes problemas, precisam lidar com quantidades gigantescas de dados, e o primeiro cluster, composto por 16 micros 486 DX4 comuns ligados em rede, foi construído para resolver esse problema.

41 Em inglês: *sandbox* ou *jail*.

Tal cluster foi chamado de Beowulf, e assim ficaram sendo conhecidos toda uma classe de clusters baseados em componentes comuns, de prateleira[42].

Por meio de um cluster de micros comuns é possível obter altíssimo poder de processamento a um custo bastante reduzido, pois, ao contrário dos componentes topo de linha, usados em supercomputadores, as peças de prateleira já tiveram seus custos bastante reduzidos devido à grande disponibilidade no mercado.

Como o Beowulf foi baseado em Linux e outros softwares livres, os avanços permitidos pela descoberta da equipe de pesquisadores da Nasa rapidamente se espalhou por toda a comunidade científica, e hoje está disponível para qualquer um que precise deste poder de processamento paralelo.

MOSIX

O MOSIX (Multicomputer Operating System for UnIX) é uma tecnologia que vem sendo desenvolvida há mais de 25 anos, e que a partir de 1998 vem sendo utilizada para construção de clusters baseados em Linux[43].

Entre as possíveis aplicações de um cluster MOSIX, estão[44] a distribuição automática de processos, o que possibilita

42 Em inglês, Components Of The Shelf, ou COTS.
43 History of MOSIX
http://www.mosix.org/faq/output/faq_q0004.html
44 What is MOSIX

que processos sendo executados em computadores mais lentos (ou utilizados mais intensamente) migrem para computadores mais rápidos (ou ociosos), e o gerenciamento de memória, permitindo que um computador cuja memória esteja no limite distribua sua carga, evitando o *swapping*[45].

Ao contrário do Beowulf, um cluster MOSIX não requer que as aplicações sejam modificadas para se beneficiar dos recursos computacionais disponíveis em uma rede.

http://www.mosix.org/txt_whatis.html

45 Swapping é o processo de copiar blocos de memória RAM para o HD, e vice-versa. Como o HD é uma peça física e seu tempo de acesso é ordens de grandeza maior do que o da memória RAM, o swapping frequentemente torna o processamento extremamente lento.

4.Segurança

A segurança de um sistema pode ser dividida em três aspectos distintos: confidencialidade, integridade e disponibilidade.

A confidencialidade refere-se à capacidade de manter informações em segredo; a integridade está relacionada à exatidão das informações; e a disponibilidade refere-se à capacidade de acesso, em um dado momento.

Em alguns sistemas, um aspecto pode ser mais importante do que os demais. Em geral a disponibilidade é considerada um aspecto fundamental, pois um sistema não disponível não pode sequer ser utilizado.

A segurança é um dos aspectos em que o software livre mais se destaca:

> The foundation of the business case for open-source is high reliability. Open-source software is peer-reviewed software; it is more reliable than closed, proprietary software. Mature open-source code is as bulletproof as software ever gets[46].

46 Open Source Case for Business

Um argumento comum é que, como o código-fonte está amplamente disponível, os bugs não permanecem escondidos por muito tempo.

O fato dos bugs tornarem-se visíveis pode parecer assustador à primeira vista, mas a realidade mostra que a visibilidade dos problemas é uma qualidade, e não um defeito[47].

De acordo com Eric S. Raymond:

> Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone[48].

A análise dos servidores web mais usados no mundo (Apache e IIS) confirma o fato de que o software livre apresenta muito menos problemas de segurança do que o software proprietário, mesmo contando com quase o dobro de participação no mercado.

O caso do Interbase mostra como, em um software proprietário, um problema de segurança pode permanecer escondido por muitos anos, sendo resolvido apenas após a liberação do código-fonte.

Finalmente, o AudioGalaxy mostra que, em uma indústria cada vez mais baseada em componentes proprietários,

---

http://www.opensource.org/advocacy/case_for_business.php

47 Podemos traçar um paralelo com algumas técnicas de administração da produção, como o Just in Time: ao reduzir os estoques, o administrador procura tornar os problemas visíveis, e assim corrigí-los.

48 Eric S. Raymond: The Cathedral and the Bazaar http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ar01s04.html

não há garantias de que todas as partes funcionem conforme o esperado.

O custo da insegurança

De acordo com o NIST (*National Institute of Standards and Technology*), bugs de software custam quase 60 bilhões de dólares somente aos Estados Unidos, anualmente:

> Software bugs, or errors, are so prevalent and so detrimental that they cost the U.S. economy an estimated $59.5 billion annually, or about 0.6 percent of the gross domestic product, according to a newly released study commissioned by the Department of Commerce' s National Institute of Standards and Technology (NIST).[49]

Jackob Nielsen aponta dados ainda mais alarmantes, para a economia mundial, causados pela instabilidade do Windows:

> The poor quality of Microsoft Windows costs the world economy **$170 billion per year in lost productivity** due to crashes. This is four times Bill Gates' net worth, so we are not talking pocket change, even for him, if he were forced to cover the cost of his deeds.[50]

49 Software Errors Cost U.S. Economy $59.5 Billion Annually http://www.nist.gov/public_affairs/releases/n02-10.htm

50 Jakob Nielsen's Alertbox: Poor Code Quality Contaminates Users' Conceptual Models http://www.useit.com/alertbox/20011028.html

Estudo de casos

Apache e IIS

O Apache é o servidor web mais usado no mundo, com 56% de participação no mercado, seguido pelo Microsoft IIS[51]. Mesmo tendo o dobro de mercado do IIS, o Apache teve muito menos problemas de segurança, do que o IIS em 2001. Um dos aspectos onde o Apache se destaca é na alta disponibilidade. Enquanto servidores NT/IIS precisam ser reinicializados a cada poucos dias, servidores rodando Apache ficam centenas de dias no ar, sem interrupção[52].

Em setembro de 2001, após o surgimento de um novo worm[53] que atacava o IIS, o Gartner Group publicou uma recomendação para que empresas procurassem alternativas ao servidor web da Microsoft. Na nota, o Gartner Group cita um worm que havia surgido no mês anterior, o CodeRed:

> Code Red also showed how easy it is to attack IIS Web servers (see Gartner FirstTake FT-14-2441 "Lack of Security Processes Keeps Sending Enterprises to ' Code Red' "). Thus, using Internet-exposed IIS Web servers securely has a high cost of ownership. Enterprises using Microsoft' sIIS Web server software have to update every IIS server with every Microsoft security patch that comes out -- almost weekly. However, Nimda (and to a lesser degree Code Blue) has again shown the high risk of using IIS and the effort involved in keeping up with Microsoft' s

51 Dados da Netcraft em outubro de 2001 http://www.netcraft.com

52 Idem

53 De acordo com o Jargon File 4.3.3, "a *program that propagates itself over a network, reproducing itself as it goes. Compare virus.*"

frequent security patches.[54]

Alguém poderia argumentar que os ataques são proporcionais à popularidade de uma plataforma, mas isto é desmentido pelo fato de que o Apache possui mais do que o dobro do mercado do IIS.

Por outro lado, não podemos afirmar que softwares livres, como o Apache, são invulneráveis. Em setembro de 2002, surgiu um worm que explorava uma falha do módulo OpenSSL do Apache, rodando em Linux:

> The worm typically affects Linux machines that are running Apache web server with OpenSSL enabled. Apache installations cover more than 60% of public web sites in the internet. It can be estimated that less than 10% of these installations have enabled SSL services. By some estimates, there are over one million active OpenSSL installations in the public web. A very big part of these machines have not yet been patched to close this hole, and are thus prone to infection by the Slapper worm.[55]

Este worm explorava uma falha que havia sido corrigida semanas antes, mas nem todos os servidores haviam aplicado a correção.

No entanto, como esta opção encontra-se desabilitada na maioria dos servidores, o worm não conseguiu se espalhar para além de 0,06% dos sistemas rodando Apache[56], o

54 Nimda Worm Shows You Can' t Always Patch Fast Enough
http://www3.gartner.com/DisplayDocument?doc_cd=101034

55 Global Slapper Worm Information Center
http://www.europe.f-secure.com/slapper/

56 Para este cálculo consideramos o pico de servidores afetados (13892) e o número de servidores rodando Apache – mais de 23 milhões, segundo a Netcraft.

que demonstra que a política de segurança do software livre é mais efetiva contra o surgimento de problemas de segurança.

Mas a frequência de ataques de worms e hackers não são os únicos indicadores da segurança de uma plataforma. Veremos a seguir uma análise da estabilidade dos servidores web, a partir de estudos de uptime[57].

Uptimes

Em outubro de 2001, a revista The Economist publicou uma reportagem em que apontava a República da Eslovênia como o país mais conectado ("internetted") da Europa.

De acordo a Netcraft[58], o site do governo da Eslovênia (e-gov.gov.si) roda o servidor Apache e sistema operacional Solaris, e apresenta uma média de uptime de 238 dias.

Ainda segundo as informações da Netcraft, verificamos que o site da Casa Branca (www.whitehouse.gov), que utiliza Apache e Linux, alcança uma média de 82 dias sem reboot.

Finalmente o site do governo brasileiro (www.brasil.gov.br), que roda Windows/IIS, tem um uptime médio muito inferior, ficando no ar apenas 9 dias sem reboot, o que se traduz em maiores custos de manutenção.

57 O "uptime" é uma medida da estabilidade de um sistema, representando o período em que um sistema fica no ar, sem a necessidade de reboot devido a falhas ou manutenção.

58 Informações coletadas em outubro de 2001

Se o governo brasileiro estivesse a procura de alta disponibilidade, poderia ter escolhido a combinação de Apache com uma versão proprietária de Unix, como fez a Eslovênia.

Se o governo brasileiro estivesse a procura de baixo custo e uma boa disponibilidade, deveria ter escolhido uma solução baseada inteiramente em software livre, como os EUA.

Ao escolher o meio-termo, o governo brasileiro conseguiu o pior de dois mundos: alto custo e baixa disponibilidade.

É curioso observar que, enquanto o país mais rico do mundo opta pela solução mais econômica e livremente disponível, um país que enfrenta inúmeras restrições como o Brasil opta por uma solução mais cara, sem que esta ofereça qualidade superior.

A lista dos maiores uptimes publicada pela Netcraft[59] confirma esta análise: 45 dos 50 maiores uptimes rodam o Apache, e todos eles utilizam software livre (Apache, Linux ou FreeBSD).

Interbase

O Interbase era um software proprietário, cujo código-fonte foi aberto no final do ano 2000.

Em janeiro de 2001, descobriu-se uma conta (nome de usuário e senha), escondida no código-fonte, que permitia

59 http://uptime.netcraft.com/up/today/top.avg.html (10/2001)

acesso completo ao sistema. O site da CERT[60] explica melhor a gravidade do problema:

> This back door allows any local user or remote user able to access port 3050/tcp [gds_db] to manipulate any database object on the system. This includes the ability to install trapdoors or other trojan horse software in the form of stored procedures. In addition, if the database software is running with root privileges, then any file on the server' s file system can be overwritten, possibly leading to execution of arbitrary commands as root.
>
> This vulnerability was not introduced by unauthorized modifications to the original vendor' s source. It was introduced by maintainers of the code within Borland.

O site de segurança SecurityFocus explica como o problema surgiu e permaneceu oculto por pelo menos 7 anos[61]:

> Rather than reflecting the work of a disgruntled insider or saboteur, the secret password appears to be a programmer' sill-advised solution to a software design problem, says Starkey, who has analyzed the back door code.
>
> Up until 1994, Interbase did not have its own access control mechanism -- the software was protected by the password scheme built into the underlying operating system. With version 4.0, engineers set out to change that.
>
> "What they decided to do was to set up a special database on every system that contained all the account names and the encrypted passwords," says Starkey. That model created something of a chicken-and-egg problem: To authenticate a user, the system had to have access to the password database; but to access any database -- including the password database -- the user first had to be authenticated.
>
> The unknown programmer' ssolution was to hardcode a

60 Interbase Server Contains Compiled-in Back Door Account http://www.cert.org/advisories/CA-2001-01.html

61 Interbase back door exposed http://online.securityfocus.com/news/136

> special password into the software itself--a secret shared by the client and server. The back door solved the problem, but was a devastatingly bad move from a security standpoint, says Bruce Schneier, CTO of Counterpane and author of Secrets & Lies: Digital Security in a Networked World. "As long as nobody knows about this back door, it works. It' sstill secure," says Schneier. "But as soon as somebody finds out about it, everybody is immediately and irrevocably insecure."

Em outras palavras, o erro não foi uma sabotagem, mas apenas uma solução rápida para um problema – e permaneceu oculto durante todos esses anos.

Algumas pessoas ainda acreditam que informações como essas deveriam simplesmente continuar escondidas. Para estas pessoas, tornar os problemas visíveis é um problema, e não a solução.

O caso do Interbase demonstra como a disponibilidade do código-fonte permite que um problema seja resolvido em alguns dias, ao invés de permanecer escondido por mais alguns anos.

Tornar um problema visível possibilita uma correção definitiva, e por esta razão o Gartner Group recomendou à Microsoft que a Microsoft abrisse o código fonte de seus produtos[62].

62 Gartner: Microsoft Sends Mixed Signals About Software (13/05/2002) Securityhttp://www3.gartner.com/DisplayDocument?doc_cd=106790

AudioGalaxy

Com a crescente tendência de componentização, torna-se cada vez mais difícil para o desenvolvedor de um software garantir a qualidade de todas as partes de um projeto, principalmente se ele não tiver acesso ao código-fonte dos mesmos.

Componentes de software funcionam como uma caixa-preta, isolando o programador da complexidade de um sub-sistema.

O exemplo do AudioGalaxy, uma ferramenta para troca de arquivos baseada na arquitetura peer-to-peer (P2P), mostra que, sem a transparência do código-fonte, a componentização pode trazer riscos importantes para usuários e desenvolvedores de softwares.

Descobriu-se que um componente denominado VX2 (vx2.dll), que era distribuído com o AudioGalaxy, era na verdade um *spyware*, um programa usado para espionar os hábitos e os dados digitados em um computador.

Segundo a Wired News, mostrou-se difícil descobrir quem eram os responsáveis pelo VX2:

> Trying to get to the bottom of who is behind VX2, what information it collects and what it does with it is a case study in just how insecure a place the Internet can be.
>
> The only contact information available on the company is a Hotmail address and a post office box in Las Vegas,

> Nevada. The address belongs to a company that specializes in setting up corporate shelters. E-mail to the Hotmail address went unanswered.[63]

Para piorar, nem mesmo os responsáveis pela AudioGalaxy sabiam quem – ou o que – era o VX2:

> "We know nothing about VX2," [Audio Galaxy spokesman Michael Merhej] said. The VX2 program file (called vx2.dll) was part of an advertising graphics enhancer made by the Onflow Corporation, he said. Audio Galaxy offered the Onflow program as part of its software package from Oct. 1 through Nov. 4, 2001, Merhej said. The partnership was cancelled due to unpaid bills.

Se elaborar sistemas seguros já é uma atividade difícil, é ainda mais difícil, garantir a segurança de um software, sem acesso ao código-fonte das partes que compõem um projeto.

Uma alternativa mais segura pode estar nos componentes baseados em software livre.

Nautilus

O Nautilus é um software livre, parte do ambiente gráfico Gnome.

Em 22 de maio de 2001, Miguel de Icaza enviou uma mensagem para a lista de desenvolvimento do Nautilus, onde sugeria uma modificação para torná-lo mais amigável:

> Currently when people download executables from the network, say for installing software, they can not execute them because the execute bit is not set. (...)

63 Spyware, In a Galaxy Near You
http://www.wired.com/news/technology/0,1282,49960,00.html

> So I would like to suggest that we set this bit manually if the user double clicks on a file that happens to have an a.out or ELF signature. Maybe we could popup a warning or something, but the result should be that files downloaded in this way would just work[64].

Em outras palavras, ao invés de exigir que um usuário marcasse um arquivo como executável (uma opção acessada através do botão direito do mouse) para então executá-lo, o Nautilus faria isso automaticamente, como acontece no Windows.

Se por um lado esta característica facilitaria a execução de programas, será que também não facilitaria a proliferação de vírus e trojans?

Vladimir Vukicevic apresentou o problema de uma forma bem clara:

> I still think that this is a bad idea. If a user does not have the ability to set the +x bit using Nautilus or gmc (following instructions on how to do so when downloading the installer), why is that user qualified to click "OK" on a dialog box that says "make sure this binary comes from a trusted source"? The windows ILOVEYOU scenario included many many reports of people receiving emails which said "do not open email called iloveyou", but they did anyway because it was "from someone they knew"[65].

É importante lembrar que a sugestão de Miguel de Icaza nunca foi permitir a execução de programas diretamente a partir do leitor de e-mails – o que tornaria o programa tão

64 http://mail.gnome.org/archives/mc/2001-May/msg00035.html
65 http://mail.gnome.org/archives/mc/2001-May/msg00063.html

vulnerável a vírus e worms quanto inúmeras versões do MS Outlook.

Após alguns dias de intensa discussão a idéia foi arquivada, e o Nautilus não permite a execução de arquivos sem que o usuário saiba exatamente o que está fazendo.

Windows 95 e 98

Em junho de 1999 a Microsoft anunciou a correção para uma falha no Windows 95 e 98, que fazia o computador travar a cada 49,7 dias[66], ou, mais precisamente, a cada intervalo de 4.294.967.296 milisegundos.

O erro acontecia devido à sobrecarga de um temporizador, que era armazenado em uma variável de 32 bits. O maior valor que pode ser armazenado em 32 bits corresponde à potência 2^32, ou seja, 4.294.967.296, o que acontecia a cada 49,7 dias, travando o computador.

O fato do problema só ter sido detectado em junho de 1999, mostra que até esta data ninguém conseguia manter um computador rodando Windows 95/98 por 50 dias ininterruptos. A maioria das máquinas provavelmente travava muito antes.

66 Microsoft Knowledge Base Article Q216641 - Computer Hangs After 49.7 Days http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q216641&

Windows XP

Um caso mais recente que demonstra como a facilidade de uso pode comprometer a segurança de um sistema pode ser verificado nas configurações padrão dos usuários do Windows XP (versões Home e Professional), como mostra o seguinte artigo do Microsoft Knowledge Base:

> After you install Windows XP, you have the option to create user accounts. If you create user accounts, by default, they will have an account type of Administrator with no password.[67]

Este artigo, publicado pela Microsoft, mostra duas graves falhas de segurança do sistema operacional: em primeiro lugar, todas as contas de usuários serão de tipo administrador; e em segundo lugar, estas contas de administrador são criadas sem senha, facilitando acessos não autorizados.

Para os fabricantes de softwares proprietários, facilidade de uso é uma característica desejável porque os permite vender um maior número de licenças. Para os desenvolvedores de software livre, que não têm o licenciamento como motivação principal, a facilidade de uso é uma característica desejável, desde que não comprometa a segurança de um sistema.

67 Article Q293834: User Accounts That You Create During Setup Are Administrator Account Types http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q293834&

Trustworthy Computing

Em 15 de janeiro de 2002, Bill Gates enviou um e-mail para todos os funcionários da Microsoft, afirmando que a segurança deveria ter prioridade máxima para a empresa:

> ...In the past, we' vemade our software and services more compelling for users by adding new features and functionality, and by making our platform richly extensible. We' vedone a terrific job at that, but all those great features won' tmatter unless customers trust our software.
>
> So now, when we face a choice between adding features and resolving security issues, we need to choose security[68].

Embora este e-mail tenha sido recebido com ceticismo por muitos, este é um esforço louvável, e é importante lembrarmos o bom histórico de mudanças da Microsoft.

Resta saber quanto tempo o mercado estará disposto a esperar, e quanto estará disposto a pagar por resultados que estão disponíveis hoje em diversos softwares livres.

68 Wired News: Bill Gates: Trustworthy Computing (17/01/2002) http://www.wired.com/news/business/0,1367,49826,00.html

5.Poder do fornecedor

O poder do fornecedor é uma das cinco forças competitivas que devem ser consideradas no planejamento estratégico de uma empresa[69]. Tornar-se extremamente dependente de um fornecedor pode ser um péssimo negócio, pois:

> Os fornecedores podem exercer poder de negociação sobre os participantes de uma indústria ameaçando elevar preços ou reduzir a qualidade dos bens ou serviços fornecidos. Fornecedores poderosos podem consequentemente sugar a rentabilidade de uma indústria incapaz de repassar os aumentos de custos em seus próprios preços[70].

Mas até que ponto o poder de dos fornecedores deveria ser considerado na área de tecnologia da informação? Um exemplo recente nos dá a resposta:

> Sometimes, however, a customer has little room to negotiate if a product is deemed an absolute necessity. As a case in point, many companies and agencies that use Microsoft software are bracing for a change that calls for them to commit to a certain level of annual spending. (...)
>
> Microsoft customers may face an increase in fees, ranging from 33 percent to 107 percent, if they do not

69 Porter, Michael, Estratégia Competitiva, pg. 24
70 Idem, pg. 43

> sign aboard by the July 31 deadline, according to Gartner. The research firm is advising Microsoft customers to either join the program or use a provision in the older licensing plan to avoid a substantial cost increase should they miss the deadline.[71]

Neste capítulos veremos diversos exemplos que demonstram como o software livre reduz o poder do fornecedor.

Estudo de casos

DR-DOS

Em 1988, quando a Digital Research lançou o DR-DOS, a Microsoft se viu diante do maior concorrente ao MS-DOS até então, sendo obrigada a reduzir o preço de seu sistema operacional.

Quando o DR-DOS 5 foi lançado, em 1990, ele foi aclamado pela imprensa especializada como uma alternativa tecnicamente superior e de custo mais baixo do que o MS-DOS.

Mas a Microsoft passou a adotar práticas como o anúncio de *vaporwares*, programas e características inexistentes que eram anunciados com o objetivo de diminuir a venda do software concorrente; o licenciamento de softwares por CPUs e a venda casada com o Windows 3.1 também tiveram como efeito

71 News.com: A clause for alarm (16/07/2002) http://news.com.com/2009-1017-943258.html

minimizar as vendas do DR-DOS pré-instalados em microcomputadores[72].

Além disso, a última versão beta do Windows 3.1 foi programada para simular mau funcionamento quando estivesse rodando sob o DR-DOS[73].

Nas palavras de Wendy Rohm:

> Em 1994, depois que o DR-DOS estava morto e enterrado, a Microsoft dobrou o preço do MS-DOS. Não havia outra alternativa no mercado. Como um monopolista clássico, eliminada a concorrência, elevam-se os preços.[74]

O fim do DR-DOS representou o início da ascensão de um monopólio capaz de ditar as regras de um mercado.

Netscape

O caso do Netscape apresenta algumas similaridades com o que aconteceu com o DR-DOS, mas com uma conclusão diferente.

Em 1995, com o sucesso cada vez maior da Internet (e após o fracasso de sua rede proprietária, a Microsoft Network), a Microsoft declarou guerra contra a Netscape, que na época dominava mais de 80% do mercado de navegadores.

72 Rohm, Wendy Goldman, O Caso Microsoft, pg. 62-64
73 Idem, pg. 115-117; 132-134
74 Idem, pg. 105

Através da distribuição gratuita de seu Internet Explorer e da integração com o sistema operacional, em 1998 a Microsoft conseguiu reverter a situação e dominar o mercado.

No entanto, ao invés de desaparecer do mercado, como o DR-DOS, a Netscape liberou o código-fonte de seu browser, iniciando o desenvolvimento de um novo browser a partir do zero, o Mozilla.

Esta iniciativa inédita permitiu que os browsers se transformassem em commodities, de certa forma impedindo que a Microsoft repetisse seu procedimento anterior de elevar os preços após eliminar a concorrência[75].

Hoje o Mozilla encontra-se na versão 1.1, e a versão 1.0 continua sendo mantida, no melhor estilo do software livre[76].

A abertura do código-fonte também permitiu a criação de novos produtos, como o Nautilus (que veremos a seguir) e o Galeon, um navegador extremamente leve para GNU/Linux.

Software Assurance

Um outro caso que ilustra a importância de reduzir o poder de fornecedores é a recente mudança de política de licenciamento de softwares da Microsoft, denominado *Software Assurance*, citado na introdução deste capítulo.

75 Não podemos afirmar que esta fosse a estratégia da Microsoft, mas o fato é que o Internet Explorer e suas atualizações continuam sendo disponibilizados gratuitamente.

76 Para compreender o benefício de múltiplas versões de um software, leia o texto sobre o kernel 2.0 do Linux, no capítulo "Redução de custos"

O objetivo desta nova política, segundo a Microsoft, é simplificar a aquisição e manutenção de grandes volumes de licenças de softwares[77].

Na prática, as empresas devem realizar assinaturas anuais para obter atualizações constantes, mesmo que não possam ou não queiram, perdendo o direito de realizar atualizações com descontos conforme a necessidade:

> Microsoft rolled over to the new licensing plan, called Licensing 6.0 or Software Assurance, effective July 31. Previously, enterprise customers paid a one-time licensing fee for perpetual software use. Part of that license entitled the users to pay a discounted price for upgrades when the user was ready to do so.
>
> Under Software Assurance, enterprises pay an annual subscription fee for software. That entitles them to free upgrades for the life of the subscription, but they pay the annual subscription whether they upgrade or not. The alternative: Pay full retail price for a perpetual license, and then pay full retail price again when they' reready to upgrade. Or they can simply get the latest software pre-installed on new systems, while continuing to use older versions on older systems.[78]

Segundo uma pesquisa realizada pela Sunbelt Software uma parcela significativa dos clientes da Microsoft não concordam com as mudanças nos termos de licenciamento:

> "Negative reaction to the upcoming Microsoft Licensing 6.0 Program is increasing and not lessening, as the August 1, 2002 launch date looms closer. A significant portion -over one-third - 36 percent -- of corporate enterprises report they don' thave the necessary funds to

77 Software Assurance and Volume Licensing 6.0 Programs http://www.microsoft.com/licensing/programs/sa/sadefined.asp

78 InternetWeek.com: Microsoft Licensing: Still A Sticking Point For Some Enterprises (09/11/2002) http://www.internetweek.com/story/INW20021009S0004

> upgrade to the Microsoft Licensing 6.0 Program, while another 38 percent say they are actively seeking alternatives to Microsoft products."[79]

Eazel

Quando um fornecedor de softwares proprietários fecha suas portas, ou, em casos menos críticos, simplesmente deixa de suportar uma antiga linha de produtos, seus clientes ficam de mãos atadas, com sistemas legados que precisam ser substituídos.

O caso da Eazel demonstra como o usuário de software livre se encontra em uma situação de vantagem, mesmo quando um de seus fornecedores encerra suas operações.

A Eazel foi criada com o objetivo de produzir uma interface amigável para o Linux, o Nautilus. Baseado no código-fonte do Mozilla, o projeto iniciou em 1999, gerando um software de altíssima qualidade, que infelizmente não se traduziu em lucros para a empresa, que fechou em 2001.

No entanto, assim que foi publicado o anúncio do fechamento da Eazel, imediatamente um grupo de usuários se juntou para dar prosseguimento ao projeto, que hoje é parte integrante da interface gráfica Gnome.

Este caso mostra que softwares livres continuam sendo mantidos enquanto houver interesse de pelo menos parte da

79 Microsoft Licensing 6.0 Survey
http://www.sunbelt-software.com/survey_02mar.cfm

comunidade. No caso de softwares proprietários, o resultado costuma ser um programa promissor fadado ao esquecimento.

A importância dos padrões abertos

Um dos aspectos fundamentais para reduzir o o poder de um fornecedor é a adoção de padrões abertos.

Nós percebemos claramente, que o modelo proprietário incentiva a ruptura de padrões, com o objetivo de beneficiar o desenvolvedor, muitas vezes em detrimento dos interesses de seus usuários.

Isso é observado claramente em táticas como a incompatibilidade planejada, que consiste em introduzir pequenas diferenças em um programa, de forma que ele não funcione de acordo com os padrões, ou da forma esperada.

A mensagem de erro do Windows 3.1 beta sob o DR-DOS[80] é um exemplo de incompatibilidade planejada, mas a incompatibilidade também pode existir entre diferentes versões de um mesmo software, obrigando seus usuários a realizarem um upgrade.

80 O caso do DR-DOS é discutido no capítulo que fala sobre o poder dos fornecedores.

Internet Explorer versus Apache

Considerando que a Microsoft domina 95% do mercado de navegadores, a possibilidade de que ela venha a utilizar esse poder para ditar padrões proprietários na internet é no mínimo preocupante.

Infelizmente, esta não é uma mera possibilidade:

> eWEEK Labs has discovered that Microsoft Corp.' s Internet Explorer Version 5.0 and higher—as well as the company' s IIS Web server—has a significant security incompatibility with other major Web browsers and with the Apache Software Foundation' s Apache HTTP Web server. (...)
>
> The upshot is that IE cannot be used as a Web client for any Apache-based Web application that uses digest authentication. In addition, every non-IE browser we tested couldn' t be used as a client for any Internet Information Services-based Web application that uses digest authentication.[81]

Da mesma forma como uma mensagem de erro no Windows 3.1 beta foi usada para inibir o uso do DR-DOS, uma incompatibilidade no protocolo de autenticação pode ser usada para inibir a adoção do Apache.

> Paul Leach, Microsoft' s representative to the W3C' s digest authentication standards committee and one of the specification' s authors, attributed the problem to how the definition of one part of the digest authentication header conflicted with other statements in the standard about how the header needed to be built. Microsoft went one way; everyone else went the other way. (...)
>
> "It definitely looks like a bug in MS IE," said Apache

81 IE, Apache Clash on Web Standard
http://www.eweek.com/article2/0,3959,26252,00.asp

> Software Foundation Chairman Roy Fielding, in Newport Beach, Calif. "We will not change our implementation in order to accommodate this bug, since it could be considered a weakening of that digest authentication feature."[82]

Estratégia para escolha de fornecedores

Os casos apresentados até aqui mostram que o poder do fornecedor é uma variável que precisa ser levada em consideração ao se tomar decisões de investimento em novas tecnologias, e que optar por soluções livres tem o efeito de minimizar a dependência de fornecedores.

Mais do que isso, optar por um único fornecedor e por padrões proprietários é um erro estratégico, pois fortalece as barreiras contra a mudança ao mesmo tempo em que reduz o poder de negociação. Na prática, a empresa se torna refém do fornecedor.

Mesmo que a decisão de optar por softwares livres levasse a um aumento de custos a curto prazo (o que pode ser minimizado graças à economia imediata em licenças, hardware e segurança, como veremos no próximo capítulo), o objetivo de garantir a independência de fornecedor deveria ser perseguido, de forma a garantir a redução de custos a longo prazo.

82 Idem

6.Custos

A vantagem de custo do software livre não está somente na economia com o pagamento de licenças: esta é uma vantagem clara e imediatamente percebida, mas o software livre também reduz os custos de mudança que estão frequentemente associados aos softwares proprietários.

Lembrando que a criação de custos de mudança é parte essencial da estratégia de muitos fabricantes de softwares, analisaremos a seguir os principais custos para a substituição de um software proprietário por um software livre equivalente.

É importante lembrar que muitos fabricantes de softwares proprietários deixam de suportar seus antigos sistemas, forçando um upgrade. No caso da Microsoft, por exemplo, qualquer empresa que ainda utilize Windows 95, NT, 98 e Me, é fortemente incentivada a adquirir versões mais novas do Windows:

> Microsoft said customers need to upgrade to the latest versions of Windows to get stronger security.
>
> Customers' continued reliance on earlier versions of Windows, rather than the current Windows 2000 and Windows XP, is slowing down efforts to secure the global

> computing infrastructure, Craig Mundie, Microsoft chief technical officer, said in an address at the company' s campus in Mountain View, Calif. He said it' simpossible to retrofit earlier versions of Windows to make them secure[83].

Novas versões de aplicativos também são usadas para incentivar esta mudança:

> According to published reports, Office 11 (due next year) won' t support Windows 95, 98, or ME. The earliest supported version, per the reports, will be Windows 2000. This means that a substantial number of organizations with pre-W2K systems will have to undergo a serious upgrade path or risk becoming incompatible with new versions of Office.
>
> Compare this with OpenOffice.org, which runs not only on Windows 95 and later, but on Linux, Solaris, and now Mac OS X as well. This gives you compatibility now and options in the future if migration to another OS supports your business needs. OpenOffice.org is continually working to maintain support for new Office file types[84].

A mudança para um software livre pode apresentar uma série de custos imediatos, mas oferece uma série de vantagens, incluindo a redução de custos a médio e longo prazo.

Custos de mudança

Entre as principais fontes de custos de mudanças de fornecedores, podemos destacar[85]:

83 InternetWeek: Microsoft To Customers: Want Better Security? Upgrade (15/11/2002) http://www.internetwk.com/breakingNews/INW20021115S0009

84 InfoWorld: Switching Offices (21/11/2002) http://www.infoworld.com/articles/op/xml/02/11/11/021111opsource.xml

85 Porter, Michael. *Estratégia Competitiva*. pp. 119

- Custo de modificação de produtos para adaptá-los ao produto de um novo fornecedor
- Custo de teste do novo produto
- Investimentos para reciclagem de empregados
- Investimentos em novos equipamentos para utilização do novo produto
- Custos psicológicos de desfazer um relacionamento.

Custo de modificação de produtos

Empresas que possuem sistemas legados desenvolvidos sobre uma infra-estrutura baseada em softwares proprietários, tendem a enfrentar custos de mudança na mudança para qualquer nova plataforma.

Nessa situação, existem pelo menos duas situações em que a substituição é vantajosa para a empresa:

- Se a substituição dos sistemas legados já estivesse prevista para a plataforma proprietária
- Se o custo da transferência dos sistemas legados para a plataforma livre for inferior ao custo da manutenção da infra-estrutura proprietária.

Custo de teste do novo produto

O custo de teste de um software está fortemente ligado à aquisição, instalação e manutenção do mesmo.

Considerando que o custo de aquisição de um software livre é tipicamente zero, e que os testes podem ser realizados em projetos piloto de pequena escala, o custo de teste de um software livre é relativamente baixo.

O mesmo não pode ser dito a respeito de novas versões de softwares proprietários, que a partir da aquisição apresentam custos mais altos.

Investimentos para reciclagem de empregados

Quando os funcionários de uma empresa não estão capacitados a utilizar um novo software, é necessário investir em treinamento.

Em alguns casos a adaptação é simples: um usuário do MS Office não precisa de muito treinamento para aprender a operar o OpenOffice. Em outros casos, pode ser necessário um investimento maior; as interfaces gráficas de GNU/Linux não são tão bem acabadas quanto as interfaces do Windows e do Mac OS X, embora estejam evoluindo rapidamente.

De qualquer modo, devemos lembrar que investimentos em treinamento são melhores do que investimentos em licenças de software, em termos práticos: um administrador

bem preparado pode dar suporte a centenas de servidores locais e remotos rodando software livre.

Investimentos em novos equipamentos

Como veremos ainda neste capítulo, enquanto as versões mais novas de softwares proprietários exigem equipamentos cada vez mais avançados, o software livre adapta-se às mais diversas configurações.

Assim, o investimento em novos equipamentos é um problema maior na migração para novas versões de softwares proprietários do que na migração para software livre.

Custos psicológicos de desfazer um relacionamento

Ainda que este custo não seja facilmente contabilizado, é interessante observar que o custo psicológico pode ser uma das maiores barreiras à mudança.

Por maiores que sejam os problemas causados por um sistema, estes podem ser minimizados inconscientemente pelo custo psicológico de uma eventual mudança.

Neste caso, até mesmo a vantagem de custo pode ser encarada com desconfiança:

> Os primeiros mercados a adquirir um novo produto, considerando as demais condições em situação de igualdade, são geralmente aqueles em que existe uma

> vantagem no desempenho. Esta situação ocorre porque a obtenção de uma vantagem de custo *na prática* é frequentemente vista com suspeita quando os compradores se confrontam com a novidade.[86]

O medo pode surgir como resposta, caso haja alguma percepção de que a novidade possa não ser tão boa quanto parece, o que é estimulado por campanhas de medo, incerteza e dúvida patrocinadas por empresas já estabelecidas:

> There are many reasons that individual managers opt for business survival over advancement; few of the reasons are rational, but they are based on solid fears, anxieties, and threats to self-image.[87]

Um planejamento rigoroso, a capacitação e os projetos piloto são importantes para minimizar o medo da mudança.

Licenças

A primeira e mais óbvia fonte de custos com softwares proprietários é o pagamento de licenças. As licenças podem representar, segundo algumas estimativas, até 20% do custo total de propriedade (TCO, ou *Total Cost of Ownership*) de um software[88].

Além dos custos diretos, com o pagamento de licenças, existem também os custos indiretos, como os problemas

86 Porter (1), pp. 216
87 Krogh,pp. 78
88 ComputerWorld: TCO: Linux Delivers On Big Iron (13/05/2002) http://www.computerworld.com/hardwaretopics/hardware/story/0,10801,70944,00.html

de segurança e a necessidade de gerenciar as licenças. Para minimizar este problema existem softwares especializados no gerenciamento de licenças de software; os quais, também possuem suas próprias licenças[89]. E, finalmente, existe o risco de que alguns programas não estejam devidamente licenciados.

O custo da pirataria

Recentes campanhas da Associação Brasileira de Software, procuram realizar apreensão de softwares piratas em empresas de todo o Brasil. Campanhas semelhantes são realizadas, por entidades similares, em todo o mundo.

A pena prevista na legislação, para o uso de softwares piratas, pode ser de ate 3 mil vezes o preço do software original, ou até quatro anos de prisão. Para pequenas empresas isso pode significar a falência; para empresas maiores, além do prejuízo financeiro, o uso de softwares piratas pode trazer danos à imagem.

Mesmo que a empresa tenha uma política de proibição de uso de softwares piratas, sempre existe o risco de que programas piratas sejam instalados por funcionários da empresa.

A determinação do custo relacionado ao risco da pirataria é uma atividade complexa, e que não entra nos estudos de TCO patrocinados por empresas de software proprietário.

89 Nós discutiremos os termos de licenciamento de softwares no capítulo sobre aspectos éticos e legais.

Requisitos de hardware

Softwares proprietários são conhecidos por incluir novas funcionalidades a cada nova versão, com o objetivo de estimular atualizações.

A inclusão de novas funcionalidades, porém, pode acabar indo contra os interesses do usuário final, pois estes programas acabam transformando-se em sistemas pesados, *bloatwares*[90] com milhares de funcionalidades que acabam não sendo usadas[91].

O software livre, em contrapartida, não coloca os interesses comerciais acima dos interesses do usuário final, de maneira que um usuário satisfeito com uma versão de um software não é "estimulado" a atualizar seu sistema para uma versão mais recente. Um exemplo disso é a versão 2.0 do *kernel* do Linux:

O kernel 2.0

Uma discussão interessante aconteceu na lista de desenvolvimento do kernel do Linux, a respeito do futuro do kernel 2.0, que havia sido lançado oficialmente há mais de 6 anos, em junho de 1996[92].

90 De acordo com o Jargon File: "Software that provides minimal functionality while requiring a disproportionate amount of diskspace and memory. Especially used for application and OS upgrades. This term is very common in the Windows/NT world. So is its cause."

91 A inclusão de novas características também pode trazer novos problemas de segurança.

92 A versão mais recente do kernel (usada na maioria das distribuições atuais), é o kernel 2.4, que foi lançado em janeiro de 2001, e a próxima versão estável (2.6), deve ser lançada nos próximos meses.

Curiosamente, para muitos, o kernel 2.0 continua sendo mantido. Em 13 de julho de 2002, um usuário lançou a seguinte pergunta na lista[93]:

> Hi everbody, Will kernel tree 2.0 stop developing and regard historical after the release of 2.6? I think we would put our focus on much more newer kernel. And I found this may confuse the newbies, because they don' t know much about versioning in Kernel. In nowsdays, there are less less compputers using 2.0. We should push them to upgrade, so I think stop developing 2.0 is better, in my opinion

As respostas vieram rapidamente. Alguns pareceram irritados com o comentário ("*I've never seen you develop 2.0. How can you stop if you didn't start?*"), mas a discussão que se seguiu foi bastante produtiva, demonstrando que havia, sim, interesse em manter o desenvolvimento de um kernel tão antigo:

> I think one always gets one' senvironment tuned to fit himself. (At least I do.) I still have some 2.0 machines running, and they' rerunning fine. They ran fine since Adam, and will still run fine as long as 2.0 is maintained. I can run them without too much administration effort (this is cool, since they' reabout 100 miles away...) When I have them administrated, I always get this comfortable feeling, because the most of it is done by large scripts which check input and compute the output by themselves. If I dropped e.g. the old firewalling style, I' d have to change ~60% of my scripts to the new firewalling style. I think this is a good reason not to upgrade to a so-called "recent" kernel on those boxes.

Alan Cox, um dos principais desenvolvedores do kernel também deu sua opinião sobre este assunto:

93 KernelTrap: The future of the 2.0 Kernel
http://www.kerneltrap.org/node.php?id=330

> Why should you care ? 2.0 can continue to slowly and cautiously get critical bug fixes between now and the end of time providing someone cares enough to do the work. There are plenty of 2.0 boxes employed as routers, print servers, intranet dialins etc which will probably only become 2.4 boxes when the hardware is taken out of service.

O comentário final foi dado pelo próprio David Weinehall, o mantenedor do kernel 2.0:

> The developer-force going into the 2.0-series is not very big. I consolidate the few fixes I get sent my way that are reasonable, and reject the rest (lately, most have been reasonable...), and try to backport some fixes from 2.2/2.4 that are applicable. No new drivers are added (or developed), and no new features are added. Besides me, there are a few (no more than five) persons that regularly report their success/failure/personal gripes with the latest 2.0-releases, and remind me to increase the release-number (I'm as bad as Alan in this regard...) The amount of work that I'd spend on a newer kernel would be about the same, and since I've grown fond of this work, I'll probably not drop 2.0 unless I get offered to take over 2.2 or 2.4 some point in the future. Mind you, there _are_ people that still use 2.0 and wouldn't consider an upgrade the next few years, simply because they know that their software/hardware works with 2.0 and have documented all quirks. Upgrading to a newer kernel-series means going through this work again. And most likely, the upgrade would be to 2.2 rather than 2.4, because 2.4 still gets new features and API-changes now and then, something generally frowned upon in a controlled environment. I am about to release 2.0.40 soon, and while 40 is a nice round number, 42 is an even better number to stop at, so that'll probably be the end of the road. That end lies quite some time in the future, though. Regards: David Weinehall

Tomsrtbt

Tomsrtbt (pronuncia-se "Tom' s Root Boot"[94]) é o nome de uma distribuição Linux contida em um único disquete de 3 ½ polegadas. O objetivo dessa pequena distribuição é permitir o resgate de sistemas em situações de emergência, e oferecer "o máximo de Linux em um único disquete".

Para que o sistema possa ser distribuído em um único disquete, não há interface gráfica, mas uma vez inicializado, o tomsrtbt oferece um shell e cerca de 200 módulos e comandos. Com isso é possível copiar arquivos, editar textos, montar e desmontar sistemas, e até mesmo conectar-se à internet.

Como a principal restrição do tomsrtbt é o espaço ocupado, essa distribuição utiliza o kernel 2.0, demonstrando mais uma vez, que os sistemas mais antigos ainda podem ser úteis.

O projeto RULE

O projeto RULE (Run Up2date Linux Everywhere)[95] tem como objetivos:

- Modificar o sistema de instalação do Red Hat Linux, para que ele possa rodar em sistemas com menos de 32Mb de RAM.

94 Segundo o autor, a sigla tomsrtbt significa "Tom' s floppy which has a root filesystem and is also bootable."

95 http://www.freesoftware.fsf.org/rule/

- Selecionar, testar e empacotar as aplicações que ofereçam a melhor funcionalidade com o menor consumo de recursos (CPU e RAM).
- Criar uma nova opção de instalação para o Red Hat Linux (e não uma nova distribuição), contendo todos e apenas os pacotes acima, otimizados para rodar como cliente ou servidor em equipamentos obsoletos, exigindo o mínimo possível de espaco em RAM e HD.
- Promover o uso desta opção em escolas, organizações privadas ou públicas, especialmente em países em desenvolvimento.

Essa distribuição Linux modificada facilitará o aproveitamento de micros antigos, oferecendo uma redução de custos imediata sem a necessidade de investir em hardware.

Devemos ressaltar o fato desse projeto só poder existir devido à liberdade de uso, cópia e modificação de softwares livres.

Terminais gráficos

Uma opção ainda mais atrativa para aproveitar micros antigos é a transformação dos mesmos em terminais gráficos com boot remoto.

Terminais gráficos não possuem disco rígido, o que elimina uma das principais fontes de problemas de manutenção. O

único hardware adicional necessário é uma placa de rede, que é usada para conectar-se ao servidor principal, onde residem todos os aplicativos.

Por exemplo, dez micros 486 podem se conectar a um servidor de 500MHz e 256Mb RAM, que cuida desde o gerenciamento dos arquivos dos usuários até as aplicações (aplicativos de escritório, navegadores, etc).

O Linux Terminal Server Project, um projeto voltado à divulgação desse tipo de configuração, explica melhor o que pode ser obtido:

> The really neat thing is that you can have lots of workstations served by a single GNU/Linux server. How many workstations? Well, that depends on the size of the server, and the applications that will be used.
>
> It' snot unusual to have 40 workstations, all running Netscape and StarOffice from a Dual PIII-650 with 1GB of ram.[96]

Comparando-se ao custo de renovar um parque de dezenas ou centenas de máquinas, esta configuração apresenta ganhos desde a aquisição até a manutenção.

No próximo capítulo veremos os aspectos legais e éticos do software livre, e veremos como iniciativas como esta são literalmente proibidas por licenças de softwares proprietários.

96 LTSP – Linux Terminal Server Project – v. 3.0
http://www.ltsp.org/documentation/ltsp-3.0.0/ltsp-3.0.htmlI

7.Aspectos éticos e legais

Como foi dito na introdução, o objetivo deste trabalho é apresentar as vantagens do software livre para o ambiente corporativo, independentemente de posições filosóficas ou princípios morais.

Tendo em vista este objetivo, procuramos mostrar que, além da economia imediata em licenças, o software livre oferece vantagens como a diminuição do poder de fornecedores, possibilidade de diferenciação, redução de custos diretos e indiretos, aumento da segurança e confiabilidade.

Veremos agora uma vantagem fundamental do software livre: os aspectos éticos e legais.

Licenças

As licenças de softwares proprietários, ou EULAs, frequentemente procuram estabelecer uma condição de domínio dos fornecedores sobre o usuário final[97].

97 Veja o item "Liberdade ou poder?", ao final deste capítulo.

A grande maioria dos usuários de software, porém, não lêem as licenças com a devida atenção:

> It never ceases to amaze me how people don' teven pay attention to these clauses as they blithely sign-off on one-sided agreements. It' sjust one little clause -- and yet it can cause so much damage. (...)
>
> Now, if you sign off on a clause like that because you figure that your lawyer will find some technicality to overcome it, I' d say don' t depend on it. As a generalization, it means what it says. Judges can read and will probably enforce it as written[98].

A maioria das empresas simplesmente aceita todas as cláusulas impostas, sem tentar qualquer tipo de negociação:

> Given the multimillion-dollar costs of today' ssoftware deals, customers often devote surprisingly little time and resources to scrutiny of contracts that are rife with obscure terminology, vague expansion charges, and mind-boggling license conversions. As a result, they end up paying in ways they had never imagined long after the deal is signed.[99]

Esta é uma situação bastante confortável para a indústria de softwares, pois uma leitura mais crítica poderia levar à renegociação do preço das licenças – ou, em alguns casos, o cancelamento do contrato.

É verdade que muitas das cláusulas dos softwares proprietários são consideradas padrões da indústria. Mas isto não significa dizer que estas cláusulas devam ser aceitas:

> "Standard" contracts are often one-sided agreements

98 Business 2.0: Don' t Accept Limits on Other Party' s Liability (25/06/2001) http://www.business2.com/articles/web/0,,16186,FF.html

99 News.com: A clause for alarm (16/07/2002) http://news.com.com/2009-1017-943258.html

> favoring the supplier. Customers should not sign these without thorough review by an attorney, even if the sales representative presents it as a way to get the ball rolling or is pressuring for immediate action[100].

Como também veremos neste capítulo, não bastando o fato dos usuários de softwares proprietários estarem sujeitos a cláusulas cada vez mais duras, estas são frequentemente modificadas após a venda, de acordo com os interesses do fabricante.

Como vimos anteriormente, o poder dos fornecedores é uma força competitiva que deve ser combatida através de políticas de compras que privilegiem a diversificação de fornecedores e produtos padronizados. As licenças de software são uma face visível do poder do fornecedor.

A licença GPL

A licença GPL pode ser resumida em quatro liberdades fundamentais e uma obrigação:

- A liberdade de usar um programa
- A liberdade para estudar o programa, e compreender seu funcionamento
- A liberdade para modificá-lo

100Idem

- A liberdade para redistribuí-lo, com ou sem modificações
- A obrigação de respeitar estas quatro liberdades.

No próximo ítem analisaremos a EULA do Windows 2000[101], comparando-a com a licença GPL em diversos aspectos.

EULA do Windows 2000

Como já discutimos anteriormente, os termos de licença da maioria dos softwares proprietários procuram proteger exclusivamente o fabricante, oferecendo direitos mínimos ao usuário final.

Instalação e uso

De acordo com a EULA do Windows 2000, o usuário "poderá instalar, usar, acessar, exibir e executar uma cópia do SOFTWARE em um único computador, por exemplo, uma estação de trabalho, terminal ou outro dispositivo". Isto significa que o usuário não tem liberdade para instalar o programa em quantos micros desejar, uma clara desvantagem em comparação à GPL, que permite que um programa seja instalado em quantos computadores forem necessários.

101Para toda a análise a seguir foi usada uma cópia da licença do Windows 2000 Professional, em português.

Além da restrição ao número de computadores onde é instalado, a EULA do Windows 2000 limita o número de processadores daquele computador onde o programa será usado: “o Produto não pode ser usado por mais de dois (2) processadores simultaneamente em qualquer Estação de Trabalho”. Novamente, a licença GPL não impõe nenhuma restrição ao tipo de computador, ou ao número de processadores utilizados simultaneamente.

A EULA do Windows 2000 também limita o número de computadores que poderão utilizar os serviços do computador, especificando claramente quais serviços poderão ser usados: “Você pode permitir que um máximo de dez (10) computadores ou outros dispositivos eletrônicos (cada um sendo um "Dispositivo") sejam conectados à Estação de Trabalho para utilizar os serviços do Produto, para serviços de arquivo e impressão, serviços de informação na Internet e acesso remoto...”. Mais uma vez, a licença GPL não impõe qualquer restrição ao número de dispositivos que possam acessar os serviços do sistema, e também não restringe quais serviços podem ser efetivamente utilizados.

A licença do Windows 2000 prossegue proibindo o uso dos programas de uma estação de trabalho a partir de outros computadores conectados, mesmo que isso seja tecnicamente possível: “Você não poderá usar o Software para permitir que um Dispositivo use, acesse, exiba ou execute outros softwares executáveis que residam na Estação de Trabalho, nem permitir

que qualquer Dispositivo exiba a interface de usuário do Software, a menos que o Dispositivo tenha uma licença separada para o Software". Mais uma vez, a licença GPL não coloca nenhuma restrição à execução de programas em terminais remotos, e de fato este é o princípio dos terminais gráficos vistos no capítulo anterior.

A licença do Windows 2000 prossegue informando as condições para que um programa possa ser compartilhado em uma rede: "...você deve adquirir e dedicar uma licença adicional para cada Estação de Trabalho separada na qual o Produto será instalado, usado, acessado, exibido ou executado". Vale lembrar que no GNU/Linux, isso pode ser feito gratuitamente, uma vez que não há quaisquer restrições.

Atualizações de licenças

A EULA do Windows 2000 permite que novos termos de licenciamento sejam incluídos na atualizações de um produto, quando diz que "Este EULA se aplica às atualizações e aos complementos do Produto original fornecido pela Microsoft, **a não ser que sejam fornecidos outros termos juntamente com a atualização ou o complemento**" (grifo nosso).

De fato, os termos de licenciamento dos produtos da Microsoft não apenas diferem entre si, o que já exige do usuário um certo empenho para manter-se atualizado com seus deveres e

obrigações, como também são modificados frequentemente no decorrer do tempo.

A licença GPL, em contrapartida, é uma licença geral, aplicada a inúmeros softwares livres, e permanece inalterada desde 1991.

Mais do que isso, a licença GPL permite, em algumas circunstâncias[102], que o usuário escolha qual versão da licença deseja obedecer:

> The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
>
> Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

Isto significa que, por omissão, o usuário é sempre beneficiado, o que não acontece no caso do Windows 2000: "A Microsoft reserva a si todos os direitos que não foram expressamente concedidos a você neste EULA".

102A maioria dos softwares livres são disponibilizados com o seguinte texto, de acordo com as recomendações da Free Software Foundation: "*This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.*"

Transferência

A licença do Windows 2000 permite que o software seja movido, e não copiado, para uma outra estação de trabalho interna, ou, no caso de transferências externas, "o usuário inicial do Produto pode fazer uma única transferência do Produto para outro usuário final". Neste caso, o usuário original deverá transferir todo o material que veio com o programa, e deverá interromper o uso do mesmo. Finalmente, "é vedado o aluguel, arrendamento ou empréstimo do Produto".

A licença GPL permite a distribuição de cópias de um programa, com ou sem modificações, desde que a licença original seja preservada. É permitido vender o programa, e não há restrições para o preço cobrado – é bom lembrar que a venda de softwares livres não apenas é o modelo de negócios de inúmeras distribuições Linux, como também é estimulada pela Free Software Foundation, que afirma claramente:

> Distributing free software is an opportunity to raise funds for development. Don' t waste it![103]

Engenharia reversa

A licença do Windows 2000 proíbe a engenharia reversa do programa, o que significa a proibição do aprendizado. A

103Selling Free Software
http://www.gnu.org/philosophy/selling.html

licença GPL, em contrapartida, foi criada justamente para impedir proibições como essa.

Rescisão

A licença do Windows 2000 dá direito à Microsoft de rescindir o contrato, caso você não cumpra com seus termos e condições. "Nesse caso, você deverá destruir todas as cópias do Produto e todos os seus componentes". A licença GPL não possui nenhuma cláusula semelhante.

Isenção de garantias

O único tópico no qual a licença GPL e a EULA do Windows 2000 possuem grandes semelhanças é na isenção de garantias: as duas licenças procuram limitar as responsabilidades do desenvolvedor e de seus contribuidores.

Ainda assim, a GPL apresenta uma vantagem ao permitir que um distribuidor do programa ofereça uma garantia, o que abre mais uma oportunidade de negócios com software livre:

> You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

A EULA do Windows 2000 conclui afirmando que "não obstante quaisquer danos a que você possa estar sujeito por qualquer motivo (...), a responsabilidade integral da Microsoft e de quaisquer de seus fornecedores (...) serão limitados ao valor

efetivamente pago por você pelo Produto ou U.S.$5.00, o que for maior".

Licenças complementares

Quando um usuário instala o Windows 2000, a primeira tarefa que deve ser realizada é a aplicação das correções de segurança, ou "*service packs*".

Pode-se afirmar que estas atualizações são indispensáveis, uma vez que inúmeros problemas de segurança foram corrigidos desde o lançamento do sistema operacional.

No entanto, a Microsoft utiliza as atualizações de segurança como meio de propagar novos termos de licença com uma EULA complementar.

Uma das novas cláusulas oferece à Microsoft liberdade para monitorar e modificar os programas instalados em um computador: "Você reconhece e concorda que a Microsoft pode verificar automaticamente a versão do Produto e/ou dos componentes sendo utilizados e pode fornecer atualizações ou correções para o Produto que serão automaticamente descarregados na sua Estação de trabalho"[104].

Com isso, o usuário tem apenas duas opções: aceitar que a Microsoft tenha controle sobre os programas instalados em

104CONTRATO DE LICENÇA DE USUÁRIO FINAL COMPLEMENTAR PARA SOFTWARE MICROSOFT

seu computador pessoal, ou utilizar o computador sem as atualizações de segurança (o que, para a maioria das empresas, não é uma opção viável).

Windows Media Player

Em abril de 2000, o Wall Street Journal publicou um artigo onde afirmava que a Microsoft tentaria enfraquecer o uso do formato MP3, através de seu Windows Media Player:

> MP3, a popular format for downloading music from the Web, is encountering competitive pressure as leading technology companies such as Microsoft Corp. work to subtly wean consumers away from the technology. (...)
>
> Microsoft, for example, plans to severely limit the quality of music that can be recorded as an MP3 file using software built into the next version of its personal-computer operating system, Windows XP. But music recorded in the Redmond, Wash., software company' s own format, called Windows Media Audio, will sound clearer and require far less storage space on a computer. (...)
>
> "The industry doesn' twant [MP3] pushed, and Microsoft and RealNetworks don' twant it pushed. The consumer is going to eat what he' sgiven," says David Farber, the former chief technologist at the Federal Communications Commission.

De fato, já existe uma grande pressão da indústria de conteúdo para que sistemas de proteção de direitos autorais sejam incluídos nos dispositivos digitais:

> The content industry -- especially Hollywood and the record labels -- wants the solution built into computers and other digital devices, such as Palm Pilots and MP3 players. The industry also wants it built into software,

> operating systems, Web browsers, and routers -- the devices that guide Internet traffic. It' sa solution designed around the assumption that nearly all computer and Internet users are potential large-scale infringers.
>
> In short: The content industry wants to place a copyright cop in your computer. It also wants to station one anyplace else on the Internet where an unauthorized copy might be made[105].

Em uma recente atualização de segurança do Windows Media Player, a Microsoft utilizou uma EULA complementar para conseguir o direito de instalar programas para o "gerenciamento de direitos digitais" sem a prévia autorização do usuário:

> "You agree that in order to protect the integrity of content and software protected by digital rights management ("Secure Content"), Microsoft may provide security related updates to the OS Components that will be automatically downloaded onto your computer. **These security related updates may disable your ability to copy and/or play Secure Content and use other software on your computer.** If we provide such a security update, we will use reasonable efforts to post notices on a web site explaining the update"[106]. (Grifo nosso)

Este exemplo mostra de maneira ainda mais clara que as mudanças nos termos de licenciamento podem ir diretamente contra os interesses do usuário final, e que, mais uma vez, um usuário é obrigado a escolher entre a segurança e a

105Law.com: A Cop in Every Computer (14/01/2002) http://www.law.com/jsp/statearchive.jsp?type=Article&oldid=ZZZGFPVOGWC

106Esta EULA é identificada como "SUPPLEMENTAL END USER LICENSE AGREEMENT FOR MICROSOFT SOFTWARE", e está disponível no diretório de instalação do Windows Media Player após a atualização para a versão 7.

liberdade para controlar as modificações que são feitas em seu computador.

Shared Source

Com o sucesso cada vez maior do software livre, a Microsoft também se viu obrigada a abrir partes do código-fonte de alguns programas.

Ao contrário das licenças de software livre, que dão total liberdade de uso, inclusive comercial, a licença Shared Source da Microsoft possui uma cláusula que inviabiliza o uso comercial do programa:

> You can use this Software for any noncommercial purpose, including distributing derivatives. Running your business operations would not be considered noncommercial[107].

O fato da licença shared source ser tão restritiva é expresso no seguinte lema, frequentemente lembrado pela comunidade de usuários de software livre: "*Open Source: share and enjoy. Shared Source: look but don't touch*".

107Shared Source License for Microsoft Windows CE .NET
http://www.microsoft.com/windows/Embedded/ce.NET/evaluation/sharedsource/eula.asp

Liberdade ou poder?

Em um artigo publicado no site da Free Software Foundation, Richard Stallman e Bradley Kuhn argumentam que softwares proprietários são, na verdade, um exercício de poder:

> Proprietary software is an exercise of power. Copyright law today grants software developers that power, so they and only they choose the rules to impose on everyone else--a relatively few people make the basic software decisions for everyone, typically by denying their freedom. (...)
>
> We believe you should decide what to do with the software you use; however, that is not what today' slaw says. Current copyright law places us in the position of power over users of our code, whether we like it or not. The ethical response to this situation is to proclaim freedom for each user, just as the Bill of Rights was supposed to exercise government power by guaranteeing each citizen' sfreedoms. That is what the GNU GPL is for: it puts you in control of your usage of the software, while protecting you from others who would like to take control of your decisions.[108]

Enquanto as licenças de softwares proprietários procuram estabelecer o controle de uma empresa sobre os usuários, a licença GPL procura garantir que nenhuma empresa exerça tal domínio.

Somada a todas as vantagens estratégicas, a licença GPL é um excelente motivo para optar por um software livre ao invés de uma alternativa proprietária.

108Freedom or Power? by Bradley M. Kuhn and Richard M. Stallman
http://www.gnu.org/philosophy/freedom-or-power.html

8.Objeções

À medida em que o software livre vem ganhando popularidade, uma série de críticas e objeções são levantadas com o objetivo de dissuadir empresas de migrar de plataformas proprietárias.

Neste capítulo vamos listar algumas das críticas mais comuns, que de maneira geral podem ser divididas em "ausência de inovação", "falta de padronização", "ausência de suporte", "dificuldade de uso", "problemas de segurança", "custos escondidos" e "problemas legais".

Ausência de inovação

Uma das críticas mais comuns ao software livre consiste em afirmar que o software livre apenas imita as inovações produzidas pelo modelo proprietário: o GNU/Linux seria uma imitação do Unix; as interfaces Gnome e KDE seriam imitações do Windows; e assim por diante. Uma variação desta crítica consiste

em dizer que, sem incentivos financeiros, não há motivação para inovar.

De fato o GNU/Linux é um tipo de Unix, e as interfaces gráficas Gnome e KDE procuram trazer as facilidades do Windows e do Macintosh[109] para o GNU/Linux.

No entanto algumas das tecnologias mais revolucionárias dos últimos anos foram criadas segundo a filosofia do software livre. Dois exemplos marcantes são a internet e a world wide web.

> The networked society we live in is in large part a gift from the University of California to the world. In the 1980s, computer scientists at Berkeley working under contract for the Defense Department created an improved version of the Unix operating system, complete with a networking protocol called the TCP/IP stack. Available for a nominal fee, the operating system and network protocol grew popular with universities and became the standard for the military' sArpanet computer network. In 1992, Berkeley released its version of Unix and TCP/IP to the public as open-source code, and the combination quickly became the backbone of a network so vast that people started to call it, simply, "the Internet."
>
> Many would regard giving the Internet to the world as a benevolent act fitting for one of the world' sgreat public universities. But Bill Hoskins, who is currently in charge of protecting the intellectual property produced at U.C. Berkeley, thinks it must have been a mistake. "Whoever released the code for the Internet probably didn' t understand what they were doing," he says.
>
> Had his predecessors understood how huge the Internet would turn out to be, Hoskins figures, they would surely have licensed the protocols, sold the rights to a

109É interessante lembrar que a interface do Windows foi inspirada na interface do Macintosh, que por sua vez foi inspirada nos protótipos desenvolvidos pela Xerox, a partir de 1973.

> corporation and collected a royalty for the U.C. Regents on Internet usage years into the future. It is the kind of deal his department, the Office of Technology Licensing, cuts all the time.[110]

A world wide web, ou simplesmente "a web", é a interface gráfica da internet, composta por textos, imagens e links. O primeiro servidor e navegador web foram criados por Tim Berners Lee, em 1990.

Em uma entrevista para a Internet World, em 1994, Tim Berners Lee, explicava a importância da web permanecer uma plataforma neutra, sem dos padrões proprietários:

> The web is like paper. It doesn' tconstrain what you use it for: you have to be able to use it for all of the information flow of normal life.
>
> My priority is to see it develop and evolve in a way which will hold us in good stead for a long future. If I, and CERN, hadn' thad that attitude, there probably wouldn' t be a web now.
>
> Now, if someone tries to monopolize the Web, for example pushes proprietary variations on network protocols, then that would make me unhappy[111].

Anos depois, em uma entrevista para a CNET News.com, em comemoração aos 10 anos da primeira página web, Tim Berners Lee explica como foi o surgimento da Web:

> What was the key event in widening the Web' s use beyond scientific research?
>
> People often ask for "key events" as though life was analyzable into such things. The Web actually spread,

110Salon.com: Public money, private code
http://www.salon.com/tech/feature/2002/01/04/university_open_source/print.html

111Press FAQ
http://www.w3.org/People/Berners-Lee/FAQ.html

> initially along three axes--in High Energy Physics, among NeXT users (who could try out the original application), and among people interested in hypertext applications. The explosion was amazingly steady--for an explosion. The load on the first Web server just grew a factor of 10 every year for three years.
>
> The reason for this was that it was a grassroots syndrome. It happened because of a mass of small decisions made across the world: here a student deciding what project to do, there a few partially informed middle managers wondering about which technology to use, somewhere a sysop deciding to install something fun after a long night' swork. **A very significant factor was that the software was all (what we now call) open source. It spread fast, and could be improved fast--and it could be installed within government and large industry without having to go through a procurement process**[112]. (Grifo nosso)

Falta de padronização

Críticas desse tipo procuram levantar dúvidas sobre a padronização de projetos que podem ser modificados livremente.

A primeira crítica vem do fato de existirem centenas de distribuições Linux espalhadas por todo o mundo, sendo que qualquer pessoa poderia criar a sua. Longe de ser uma desvantagem, isso permite que soluções baseadas em software livre atendam as mais específicas necessidades, como o Embedded Linux, o Tomsrtbt, e o SE Linux.

112News.com: Charting the Web' s next transformation (12/12/2001) http://news.com.com/2008-1082-276939.html

A segunda crítica comum é o fato de que a maioria das distribuições são compostas por inúmeros projetos separados, mantidos por empresas e desenvolvedores independentes. A afirmação é verdadeira, mas isso não é uma desvantagem. O fato é que, atualmente, qualquer projeto mais complexo do que um parafuso depende de uma cadeia de valor integrada, raramente podendo ser desenvolvido por uma única empresa.

> A idústria da aviação também é uma das indústrias que já oferece bons exemplos para apoiar essa previsão. (...) Dependendo do projeto, existem dezenas ou até centenas de equipes interorganizacionais trabalhando em conjunto através dos continentes, trocando informações eletronicamente e interagindo para desenvolver planos de projetos, designs, especificações e documentação[113].

Projetos monolíticos, controlados por uma única empresa do início ao fim, são cada vez mais raros por que a empresa em questão não conseguiria competir, em termos de custo e qualidade, em todas as áreas do projeto.

Uma terceira (e falsa) crítica, afirma simplesmente não existir padronização. Na realidade, a falta de padronização é muito mais comum entre os softwares proprietários, onde o lema "*embrace and extend*"[114] é frequentemente colocado em prática. Esta prática consiste em adotar um padrão e introduzir intencionalmente novas características que não existem em produtos concorrentes; dessa forma, o produto de uma empresa permanece compatível com os concorrentes, enquanto que os

113Portais Corporativos, pg. 32
114Em tradução livre, "adotar e expandir".

concorrentes, que não possuem essas "extensões", tornam-se aparentemente incompatíveis. No caso do software livre, essa opção não existe, pois qualquer nova característica torna-se imediatamente visível para todos. O código-fonte de um software é, efetivamente, a melhor especificação possível para um padrão.

Para concluir, devemos lembrar da existência um padrão formal e escrito, denominado *Linux Standard Base* (LSB)[115].

Ao contrário de muitos padrões proprietários, o LSB conta com a contribuição de dezenas de empresas, e permanece aberto à participação da comunidade.

O LSB também utiliza diversos padrões da indústria de softwares[116].

Ausência de suporte

Dúvidas sobre a adoção do Linux por grandes empresas parecem cada vez mais infundadas, uma vez que grandes empresas da indústria como a IBM, Dell, HP, Oracle, SAP e muitas outras já declararam seu apoio (através de investimentos financeiros ou em forma de desenvolvimento) ao Linux.

Ainda assim, algumas pessoas poderiam argumentar que, se ninguém é dono do Linux, ninguém é responsável por ele.

115Linux Standard Base
http://www.linuxbase.org/

116Linux Standard Base Specification 1.3.pr3
http://www.linuxbase.org/spec/gLSB/gLSB/rstandards.html

Essa é uma falsa relação de causalidade, que equivaleria a dizer que, se ninguém é dono da internet, ninguém é responsável por ela. O Linux faz parte de uma infraestrutura neutra, e é mantido por centenas de pessoas e organizações em todo o mundo. É possível adquirir suporte comercial, desde que se pague por isso; software livre não é sinônimo de gratuito, e todo serviço realizado contribui para a melhoria do todo.

Uma outra crítica falsa, porém bastante disseminada, afirma que ninguém garante a continuidade de um software livre. Como vimos no exemplo do Nautilus, um software livre continua sendo suportado enquanto houver interesse por ele – ao contrário de softwares proprietários, que desaparecem junto com suas empresas.

Dificuldade de uso

"O Linux é difícil de usar" é uma das críticas mais frequentes e que, cada vez mais, torna-se menos verdadeira.

Como vimos, o Linux conta com interfaces gráficas extremamente amigáveis, como o Gnome e o KDE, em alguns casos superiores a qualquer concorrente proprietário. Como qualquer software, porém, existe espaço para melhorias – que não devem jamais sacrificar a segurança do sistema.

Algumas pessoas podem dizer, porém, que o excesso de opções de interfaces pode se tornar um problema para o usuário comum. Isso parece não ser verdade, uma vez que todas essas interfaces podem conviver pacificamente no mesmo computador: é possível rodar os programas do Gnome no KDE, e vice-versa.

Quando a comparação é feita especificamente com o Windows, devemos argumentar que é o Windows é tão fácil de usar. Talvez ele pareça fácil, por ser tão comum; mas a Microsoft jamais conseguiu imitar a facilidade de uso, por exemplo, de um MacOS (que hoje funciona sobre um kernel livre, o FreeBSD). O Windows tornou-se tão popular por estar disponível em uma plataforma aberta e mais econômica (o PC) ao contrário do Macintosh, cujo hardware é proprietário. Na disputa de mercado entre Linux e Windows, portanto, é de se esperar que a plataforma aberta e mais econômica prevaleça mais uma vez.

Finalmente, devemos admitir que o Linux necessita de administradores altamente treinados. Ao contrário de outros sistemas operacionais, o software livre não sacrifica segurança para aumentar a facilidade de uso. Segurança e estabilidade vêm em primeiro lugar, e a facilidade de uso tem sido construída aos poucos, sobre uma base sólida.

Esta base mais sólida permite que um administrador de sistema Linux mantenha uma quantidade superior de

computadores, em comparação com um administrador Windows, o que reflete na redução de custos:

> It is interesting to note that despite the greater salary requirements for Linux and Solaris administrators, the greater numbers of servers they can manage yields a much lower cost per Processing Unit.[117]

Problemas de segurança

Um dos grandes mitos sobre o software livre é dizer que, se o código-fonte é aberto, ele não pode ser seguro.

Isso poderia ser verdade, se qualquer um tivesse permissão para modificar o código-fonte dos programas, mas esta não é a idéia por trás do software livre: embora qualquer um possa modificar *uma cópia* do código-fonte, existe um longo caminho a ser percorrido até que essa modificação seja aceita.

Portanto, é errado dizer que os softwares livres não passam por um controle de qualidade. Este controle de qualidade é realizado por uma hierarquia de programadores estabelecida por sua capacidade e mérito, e não por critérios arbitrários[118].

Uma outra alegação comum é que os vírus, ainda raros, se tornarão comuns quando a plataforma se popularizar. Esta afirmação perde credibilidade quando observamos exemplos de softwares livres já populares, como o Apache: mesmo sendo

117Total Cost of Ownership for Linux in the Enterprise, pp. 5
118O atual mantenedor do kernel 2.4, por exemplo, é um brasileiro de apenas 19 anos, Marcelo Tosatti.

líder de mercado, possui menos problemas de segurança do que o IIS da Microsoft. Além disso, no caso do Linux, observamos que a comunidade de desenvolvedores tomou decisões muito mais seguras desde o início, construindo fundações mais sólidas.

Custos escondidos

Uma objeção comum, que também precisa ser rebatida, é de que "não existe almoço grátis", o que em alguns casos equivale a perguntar "quem é que paga a conta?".

Em primeiro lugar, é preciso lembrar que software livre não é sinônimo de software grátis. Softwares livres podem ser comercializados, e softwares proprietários podem ser distribuídos gratuitamente.

Em segundo lugar, uma boa parte do trabalho por trás do software livre não é "gratuito", mas voluntário. Desenvolvedores ao redor do mundo gastam seu tempo e recursos para melhorar os softwares existentes, e de alguma forma, não necessariamente financeira, são recompensados por seu trabalho.

Finalmente, o software livre vem recebendo cada vez mais apoio de grandes empresas, que ajudam a financiar o desenvolvimento que não é atendido pelo trabalho voluntário.

Portanto, o software livre pode ser oferecido gratuitamente, pois a conta é dividida entre milhares de pessoas e empresas.

Existem, porém, algumas situações nas quais um comprador poderá estar disposto a gastar mais por um produto, seja em licenças, treinamento ou migração de sistemas legados:

- Quando a penalidade pela falha do software é alta em relação a seu custo
- Quando a eficácia do software pode trazer grande economia para o comprador
- Quando o comprador percebe que a qualidade do software contribui para a qualidade de seu produto ou serviço
- Quando o comprador quer um software diferenciado ou sob encomenda

Como já vimos, muitos softwares livres são mais confiáveis e seguros do que os softwares proprietários correspondentes: podemos citar como exemplos o Linux, em comparação ao Windows, e o Apache, em comparação ao IIS. O uso de uma plataforma livre, nesse caso, oferece grande economia nas operações de uma empresa.

Uma outra economia indireta vem da independência de fornecedores. Ficar preso a um fornecedor, como vimos, pode permitir a esse fornecedor retirar toda a lucratividade de uma empresa.

Em algumas categorias de software, como as bases de dados, o software livre (MySQL, PostgreSQL, etc) pode não ter atingido o grau de maturidade de alternativas proprietárias como o Oracle, DB2 ou SQL Server. Ainda assim, é preciso avaliar

corretamente seus requisitos: em muitos casos as alternativas livres podem oferecer todos os benefícios esperados, por um custo muito menor. Em outros casos é possível utilizar uma solução híbrida, como Oracle rodando em Linux. Deve-se evitar, porém, soluções que estejam atadas a uma única plataforma, como é o caso do SQL Server: ao optar por esta base de dados você é obrigado a aceitar o Windows e todos os outros custos – estes sim, escondidos, pois não interessam ser divulgados pelo fabricante – da plataforma Microsoft.

Uma última objeção que precisa ser derrubada é aquele que diz que as licenças de software representam uma pequena parcela do custo total de TI. Ao contrário da indústria de microprocessadores, cujo custo cai pela metade e o poder de processamento dobra a cada 18 meses, os gastos com licenças de software vêm crescendo constantemente ao longo dos últimos anos, sem oferecer ganhos correspondentes.

> [A]s history has proven, it's a bad idea to put yourself on the other side of Moore's Law, a place Microsoft finds itself now that PC prices have finally fallen below the cost of its software. Too many companies have fought the law, and the law has won.[119]

119Business 2.0: Microsoft's Newest Challenger: Moore's Law http://www.business2.com/articles/web/0,1653,43153,00.html

Problemas legais

Uma outra acusação comum é que o software livre poderia causar problemas legais para uma empresa, especialmente na área de propriedade intelectual. Como vimos, as licenças de software livre são muito mais simples do que as licenças de softwares proprietários – estas sim, que precisam ser avaliadas com cuidado, e são modificadas unilateralmente de tempos em tempos.

A licença GPL também já foi chamada de "anti-americana", "estilo pac-man" ou "um câncer" por altos executivos da Microsoft, mas este tipo de crítica não resiste a uma análise mais profunda.

Em primeiro lugar, a idéia de que a licença seria anti-americana (ou até mesmo comunista!) parte do falso conceito de que a licença GPL não permitiria a comercialização de softwares, o que é desmentido pela própria licença:

> You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Além disso, o simples uso de um software livre também não tem qualquer influência sobre qualquer propriedade intelectual da empresa. Na verdade, a própria licença GPL mostra que o uso de softwares livres são, de fato, livres:

> Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. **The act of running the Program is**

not restricted. (Grifo nosso)

A Free Software Foundation também deixa claro que o simples ato de modificar um software livre coberto pela GPL não obriga uma empresa a liberar o código-fonte do mesmo:

> Does the GPL require that source code of modified versions be posted to the public?
>
> The GPL does not require you to release your modified version. You are free to make modifications and use them privately, without ever releasing them. This applies to organizations (including companies), too; an organization can make a modified version and use it internally without ever releasing it outside the organization.
>
> But if you release the modified version to the public in some way, the GPL requires you to make the modified source code available to the users, under the GPL.
>
> Thus, the GPL gives permission to release the modified program in certain ways, and not in other ways; but the decision of whether to release it is up to you.[120]

O ato de incorporar um software livre a um conjunto de softwares proprietários (ou vice-versa) também não obriga uma empresa a adotar a licença GPL para seus próprios programas:

> In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

Uma variante destas críticas é dizer que não se pode fazer programas proprietários para Linux, pois um programa se tornaria obrigatoriamente livre. A principal prova de que essa afirmação é falsa, é que diversas empresas de softwares

120Frequently Asked Questions about the GNU GPL
http://www.gnu.org/licenses/gpl-faq.html

proprietários estão adaptando seus programas para rodar em Linux. Oracle e SAP são apenas alguns exemplos entre muitos outros. O próprio Microsoft Office pode rodar em Linux, com a ajuda de um emulador chamado Wine, o que demonstra que não é sequer necessário modificar o código-fonte de um software proprietário para que ele funcione em Linux.

9.O que ainda precisa ser feito

Ao analisar estatísticas de softwares como o Linux e o Apache, verificamos que o software livre já conquistou uma parcela significativa do servidor, e vem apresentando forte crescimento. O próximo passo é conquistar o desktop, com o Linux, KDE, Gnome e OpenOffice.

Para que isso possa acontecer, alguns problemas precisam ser superados.

Drivers de dispositivos

Apesar de todos os avanços já realizados, o sistema GNU/Linux ainda sofre com a ausência de drivers e a consequente dificuldade de instalação de dispositivos.

Para que isto seja possível, é necessário exigir que fornecedores utilizem padrões abertos, e que os padrões abertos tornem-se padrões de fato.

Para isso é importante divulgar amplamente quais fabricantes oferecem boa compatibilidade com GNU/Linux, o que já é feito através de documentos como o Linux Hardware Compatibility HOWTO[121] e o Printer Compatibility Database[122].

Para a maioria dos fabricantes de hardware, os drivers não são fundamentais para a diferenciação de seus produtos, e portanto a divulgação das informações – ou mesmo a abertura do código fonte – pode não representar perda de vantagem competitiva. Muito pelo contrário, a abertura de novos mercados e a inovação proporcionados podem transformar-se em vantagem imediata.

Softwares livres para Windows

Mudanças radicais costumam causar resistência, e portanto uma transição mais suave para o uso de software livre pode facilitar a adaptação.

É possível manter sistemas proprietários legados, e ao mesmo tempo substituir algumas camadas por softwares livres equivalentes. Por exemplo, em um local de trabalho operando inteiramente com Windows, o OpenOffice pode substituir o MS

121Linux Hardware Compatibility HOWTO 3.2.1 (12/11/2002)
http://www.tldp.org/HOWTO/Hardware-HOWTO/
122Printer Compatibility Database
http://www.linuxprinting.org/database.html

Office, e o Mozilla pode substituir tanto o Internet Explorer e o Outlook.

Com isso dois objetivos são conquistados: o caminho para a transição para GNU/Linux começa a ser aberto e a interoperabilidade com softwares livres é facilitada.

Em termos de custos, pelo preço da licença do Windows XP e do Office XP é possível imprimir milhares de CDs contendo OpenOffice, Mozilla, Gimp e dezenas de outros softwares livres para Windows, o que pode ser usado para facilitar a transição de uma empresa e toda sua cadeia produtiva (clientes, fornecedores e funcionários).

Sistemas pré-instalados

Fabricantes de microcomputadores também podem explorar o uso de GNU/Linux ou softwares livres para Windows em micros de baixo custo, o que pode representar uma redução de centenas de dólares por micro, além de um diferencial para muitos consumidores.

Documentação para usuários comuns

Apesar do software livre já receber suporte comercial de grandes empresas, a mentalidade hacker pode intimidar os recém-chegados.

Para um hacker é normal saber que um disquete está em "/dev/fd0" e que o CD-ROM está em "/dev/cdrom". Um usuário comum, em contrapartida, prefere clicar em um ícone.

A cultura hacker valoriza a capacidade de resolução de problemas, e espera que um usuário tenha ao menos pesquisado um assunto antes de enviar uma pergunta a uma lista de discussão, como mostra Eric S. Raymond:

> Before asking a technical question by email, or in a newsgroup, or on a website chat board, do the following:
>
> 1. Try to find an answer by searching the Web.
> 2. Try to find an answer by reading the manual.
> 3. Try to find an answer by reading a FAQ.
> 4. Try to find an answer by inspection or experimentation.
> 5. Try to find an answer by asking a skilled friend.
> 6. If you are a programmer, try to find an answer by reading the source code.
>
> When you ask your question, display the fact that you have done these things first; this will help establish that you're not being a lazy sponge and wasting people's time. Better yet, display what you have *learned* from doing these things. We like answering questions for people who have demonstrated that they can learn from the answers.[123]

Atualmente os usuários comuns são contemplados com sistemas que atendem às suas necessidades, mas uma boa parte da documentação escrita por hackers ainda destina-se aos hackers.

123How To Ask Questions The Smart Way v. 2.3: http://www.tuxedo.org/~esr/faqs/smart-questions.html#before

Assim, é importante que hackers e usuários comuns passem a dedicar-se mais a esse novo público-alvo, criando documentação acessível para a maioria.

Finalmente, lembramos que este trabalho encontra-se disponível sob a licença GNU FDL, podendo ser reproduzido em qualquer meio.

10.Conclusão

Grande parte da discussão a respeito do software livre no ambiente corporativo tem sido ao redor das vantagens (ou desvantagens) de custo, argumentação estimulada pelos fabricantes de softwares proprietários – a quem interessa que os custos sejam analisados isoladamente, fora de um contexto estratégico.

Este trabalho apresentou outros aspectos que devem ser considerados, e a conclusão à qual podemos chegar é que o software livre oferece vantagens estratégicas para o ambiente corporativo.

Em primeiro lugar, vimos que o software livre possibilita a inovação e a diferenciação, o que é dificultado pelo modelo proprietário, segundo o qual o conhecimento permanece restrito.

Vimos também que o software livre oferece maior segurança, pois permite que problemas sejam encontrados e resolvidos, ao invés de permanecerem ocultos durante anos.

Ainda sob o aspecto da segurança, observamos que os desenvolvedores de software livre não sofrem a pressão para incluir novas funcionalidades que tornem o produto mais atraente, pois a venda de licenças não é a preocupação fundamental (como ocorre no modelo proprietário). Isto torna o software mais seguro e diminui os requisitos de hardware.

De todas as vantagens estratégicas, porém, o fator mais importante que deveria estimular a adoção de soluções baseadas em padrões abertos e software livre é a redução do poder do fornecedor.

Neste tópico em particular, o erro estratégico mais grave que uma empresa pode cometer é adotar como padrão as soluções proprietárias de um único fornecedor. Na prática, isto significa tornar-se refém deste fornecedor.

É importante notar, porém, que toda mudança implica em custos imediatos, e portanto a empresa deve estar disposta a realizar investimentos a curto prazo.

Felizmente os custos de mudança são minimizados graças ao baixo custo do software livre, bem como as diversas formas de economia que ele proporciona, dos requisitos de hardware à manutenção.

Do ponto de vista legal, o software livre oferece grandes vantagens sobre o modelo proprietário, a começar pelo fato que suas licenças não procuram restringir os direitos do

usuário, e permitem que inúmeros fornecedores prestem serviços comerciais com base no software.

O trabalho termina apresentando uma série de alternativas livres para as empresas, refutando possíveis críticas e objeções, e discutindo o que ainda precisa ser feito para que o software livre possa ser, cada vez mais, adotado pelo ambiente corporativo.

A conclusão mais importante que este trabalho apresenta é que, mais do que simplesmente uma questão de custo, as empresas devem começar a analisar o impacto estratégico da escolha de um fornecedor ou de uma solução, e que muitas vezes, o software livre é a melhor opção.

11.Referências bibliográficas

- Castells, Manuel. *A Sociedade em Rede - A Era da Informação: Economia, Sociedade e Cultura*. São Paulo: Paz e Terra, 1999.
- Dibona Chris et al. *Open Sources: Voices from the Open Source Revolution*. O' Reilly & Associates, 1999.
- Himanen, Pekka et al. *Hacker Ethic, The*. New York: Random House, 2001.
- Krogh, Georg Von et al. *Enabling Knowledge Creation*. New York: Oxford University Press, 2000.
- Lessig, Lawrence. *The Future of Ideas: The Fate of the Commons in a Connected World*. New York: Random House, 2001.
- Levy, Steven. *Hackers: Heroes of the Computer Revolution*. Penguin USA, 2001.
- Moody, Glyn. *Rebel Code: Linux and the Open Source Revolution*. Perseus Publishing, 2001.

- Morgan, Gareth. *Imagens da Organização: Edição Executiva*. São Paulo: Atlas, 2000.
- Porter, Michael E. *Estratégia Competitiva*. Rio de Janeiro: Campus, 1986.
- Porter, Michael E. *Vantagem Competitiva*. Rio de Janeiro: Campus, 1989.
- Terra, José C. C., Gordon, Cindy. Portais Corporativos. São Paulo: Negócio Editora, 2002.
- Toffler, Alvin. *Terceira Onda, A*. Rio de Janeiro: Record, 2001.
- Torvalds, Linus et al. *Just for Fun: The Story of an Accidental Revolutionary*. HarperBusiness, 2001.

Referências na Internet

- http://archive.salon.com/21st/feature/1998/10/cov_13feature.html
- http://estrategias.quilombodigital.org/
- http://estrategias.quilombodigital.org/wiki.cgi?search=TO-DO
- http://firstmonday.org/issues/issue8_1/nichols/index.html
- http://listas.softwarelivre.org/cgi-bin/mailman/listinfo/estrategia
- http://mail.gnome.org/archives/mc/2001-May/msg00035.html
- http://mail.gnome.org/archives/mc/2001-May/msg00063.html
- http://news.com.com/2008-1082-276939.html
- http://news.com.com/2009-1017-943258.html
- http://news.com.com/2100-100
- http://news.com.com/2100-1001-950083.html
- http://news.com.com/2100-1001-982305.html
- http://news.zdnet.co.uk/story/0,,t269-s2089446,00.html
- http://news.zdnet.co.uk/story/0,,t269-s2130760,00.html
- http://online.securityfocus.com/news/136
- http://public.wsj.com/news/hmc/sb992819157437237260.htm
- http://support.microsoft.com/default.aspx?scid=KB;
- http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2864877,00.html
- http://uptime.netcraft.com/up/today/top.avg.html

- http://user-mode-linux.sourceforge.net
- http://user-mode-linux.sourceforge.net/uses.html
- http://www.blinkenlights.com/classiccmp/gateswhine.html
- http://www.business2.com/articles/web/0,,16186,FF.html
- http://www.business2.com/articles/web/0,1653,43153,00.html
- http://www.caida.org/analysis/security/code-red
- http://www.cert.org/advisories/CA-2001-01.html
- http://www.cert.org/advisories/CA-2001-26.html
- http://www.cert.org/advisories/CA-2003-04.html
- http://www.computerworld.com/hardwaretopics/hardware/story/0,10801,70944,00.html
- http://www.debian.org/
- http://www.eff.org/IP/freeculture/
- http://www.europe.f-secure.com/slapper/
- http://www.eweek.com/article2/0,3959,26252,00.asp
- http://www.free-soft.org/literature/papers/esr/cathedral-bazaar/cathedral-bazaar-2.html
- http://www.freebsd.org/
- http://www.freesoftware.fsf.org/rule/
- http://www.gnu.org/copyleft/gpl.html
- http://www.gnu.org/gnu/gnu-linux-faq.html

- http://www.gnu.org/gnu/manifesto.html
- http://www.gnu.org/licenses/fdl.html
- http://www.gnu.org/licenses/gpl-faq.html
- http://www.gnu.org/philosophy/free-sw.html
- http://www.gnu.org/philosophy/freedom-or-power.html
- http://www.gnu.org/philosophy/selling.html
- http://www.infoworld.com/articles/op/xml/02/11/11/021111opsource.xml
- http://www.internetweek.com/story/INW20021009S0004
- http://www.internetwk.com/breakingNews/INW20021115S0009
- http://www.kerneltrap.org/node.php?id=330
- http://www.law.com/jsp/statearchive.jsp?type=Article&oldid=ZZZGFPVOGWC
- http://www.linuxbase.org/
- http://www.linuxbase.org/spec/gLSB/gLSB/rstandards.html
- http://www.linuxdevices.com/news/NS7572420206.html
- http://www.ltsp.org/documentation/ltsp-3.0.0/ltsp-3.0.html
- http://www.microsoft.com/Colombia/socios/downs/Vendiendo_Microsoft_Office_2000_Final.ppt
- http://www.microsoft.com/licensing/programs/sa/sadefined.asp
- http://www.microsoft.com/licensing/sharedsource/philosophy.asp

- http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/virus/alerts/slammer.asp
- http://www.microsoft.com/windows/Embedded/ce.NET/evaluation/sharedsource/eula.asp
- http://www.mosix.org/faq/output/faq_q0004.html
- http://www.mosix.org/txt_whatis.html
- http://www.netcraft.com
- http://www.nist.gov/public_affairs/releases/n02-10.htm
- http://www.npaci.edu/online/v7.3/slammer.worm.html
- http://www.nsa.gov/selinux/
- http://www.openbeos.org
- http://www.opensource.org/advocacy/case_for_business.php
- http://www.opensource.org/docs/definition.html
- http://www.opensource.org/docs/definition_plain.php)
- http://www.opensource.org/docs/history.php
- http://www.quilombodigital.org
- http://www.salon.com/tech/feature/2002/01/04/university_open_source/print.html
- http://www.softwarelivre.org
- http://www.sunbelt-software.com/survey_02mar.cfm
- http://www.theregister.co.uk/content/4/24885.html
- http://www.tuxedo.org/~esr/faqs/hacker-howto.html

- http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ar01s04.html
- http://www.useit.com/alertbox/20011028.html
- http://www.w3.org/People/Berners-Lee/FAQ.html
- http://www.wall.org/~larry/natural.html
- http://www.wired.com/news/business/0,1367,49826,00.html
- http://www.wired.com/news/technology/0,1282,49960,00.html
- http://www3.gartner.com/DisplayDocument?doc_cd=101034
- http://www3.gartner.com/DisplayDocument?doc_cd=106790

Índice