act; ensaio; interfaces; anotações; laboratório; entrevista;

Revista on-line
>arte
>cultura
>tecnologia

#10

Fevereiro de 2004
Quadrimestral

A filosofia do código-fonte aberto: de Richard Stallman a LINUX

[António Machuco Rosa]

O conceito de código-fonte aberto está intimamente associado ao movimento *free software*, o qual se confunde com a trajectória pessoal do seu criador e inspirador, Richard Stallman. A carreira de Stallman enquanto programador teve início num ambiente em que, se bem que ainda não institucionalizada enquanto tal, a prática de *free software* era a norma espontânea e corrente, pois era usual a cópia, distribuição e livre acesso ao código-fonte dos programas. Stallman é enfático ser esse o contexto do seu trabalho como programador. Era o contexto de uma comunidade de livre troca e acesso à informação:

> When I started working at the MIT Artificial Intelligence Lab in 1971, I became part of a software-sharing community that had existed for many years. Sharing of software was not limited to our particular community; it is as old as computers, just as sharing of recipies is as old as cooking

Esta era uma comunidade de partilha. Mas as comunidades podem fragmentar-se, desaparecer. Elas desaparecem segundo os mesmos mecanismos que conduzem à sua emergência. O raciocínio de Stallman debruça-se em termos gerais sobre esses mecanismos de formação e dissolução, sendo no entanto particularmente válido no que respeita à comunidade, e ao seu contexto, em que ele estava inserido. A tragédia da fragmentação e divisão estava a começar a ocorrer nessa comunidade específica.

Essa tragédia pode ser descrita em puros termos comunitários: uma comunidade de partilha de informação em risco de se fragmentar. Era uma comunidade científica, com todas as características que essas comunidades usualmente possuem: partilha e disponibilidade publica dos conhecimentos obtidos por cada um. No entanto, a comunidade de que Stallman fazia parte constituía não apenas uma comunidade científica, mas uma comunidade de programadores de *software*, e essa comunidade, devido a motivos bem específicos, actualiza ainda mais fortemente os riscos de desagregação em que incorrem as comunidades em geral.

A tragédia era então a seguinte. Durante as primeiras décadas da moderna computação, as décadas de 50 a 70, o *software* vivia num regime de liberdade, liberdade no sentido de qualquer um o poder utilizar, copiar e livremente aceder ao respectivo código-fonte. Mas eis que, a partir da década de 70, e ainda mais na década de 80, o *software* vai sendo progressivamente fechado: ele torna-se proprietário e começa a desaparecer do espaço público constituído pelo código-fonte aberto. Decisivo na carreira de Stallman, foi o momento em que ele viu recusado o pedido do código-fonte de uma impressora que o seu laboratório utilizava. Era um pedido normal, e constituiu um choque trágico a resposta ter sido

que esse código passava a estar sujeito a especiais contratos de licenciamento. Foi então que Stallman decidiu institucionalizar o movimento *free software* ao criar a free software foundation.

Antes de voltarmos ao momento dessa institucionalização, devemos reflectir no que significa o episódio relatado por Stallman. Ele atesta que uma comunidade de partilha de um bem publicamente disponível começou a desaparecer e a ser substituída pelo espírito concorrencial da economia moderna. É a recusa desse espírito, enquanto ele incide sobre o *software*, que parece ser um dos grandes motivos inspiradores da obra de Stallman. Não se trata de recusar por princípio o capitalismo e o mercado, substituindo-o em todas as suas dimensões por uma visão comunista da produção e troca dos bens de consumo. É algo bem mais específico e subtil, que se prende com as dinâmicas próprias das tecnologias da informação. Essas dinâmicas transitam necessariamente entre duas grandes fases - as fases de ordem e de desordem -, as quais têm de ser claramente identificadas. Sem o fazermos, não estaremos em condições de plenamente compreender a singularidade da posição de Stallman no mundo da informática.

Suponhamos que temos inicialmente um situação de desordem. Em tecnologias de informação, em especial no domínio particular do *software*, essa fase corresponde à situação em que surgem múltiplos *standards* e programas incompatíveis entre si. É a a situação da infernal fragmentação, bem conhecida na indústria nos anos 70 e 80. É a fase da concorrência pura e violenta. Mas essa fase tem em si mesma as condições da sua superação e consequente passagem a uma fase de ordem, a qual é a fase em que um *standard* monopolista emerge e as incompatibilidades tendem a desaparecer. A passagem da desordem à ordem faz-se através da retroacção positiva baseada na imitação. Considere-se, para simplificar, a existência de dois *standards* incompatíveis que se encontram inicialmente numa situação de aproximado equilíbrio em termos de quota de mercado. Pode demonstrar-se que uma pequena tendência no sentido de um deles se vai amplificar, daí resultando a sua emergência como um *standard* monopolista que exclui o seu concorrente[1]. Isto é, quando maior o número de utilizadores de um certo *standard* ou plataforma, maior o incentivo para que posteriores utilizadores adiram igualmente a essa plataforma, e assim sucessivamente até à formação de um monopólio. Trata-se do conhecido princípio de externalidades em rede: o valor de uma rede-plataforma aumenta exponencialmente com o número dos seus utilizadores, o que obriga os seguintes a imitarem o comportamento dos anteriores aderentes. Em tecnologias da informação, as dinâmicas económicas implicam que ‘o vencedor ganhe tudo’[2].

Standards de facto como o *standard* da Internet TCP/IP ou *standards* proprietários como o Windows ilustram o que acabou de ser dito, como veremos mais abaixo. Deve no entanto compreender-se um ponto fundamental. Trata-se de um verdadeiro princípio que podemos enunciar da seguinte forma: o mecanismo que leva à ordem (emergência de um *standard* monopolista) é o mesmo que leva à desordem (fragmentação em *standards* independentes e incompatíveis entre si). A emergência de um *standard* monopolista ocorre devido à retroacção positiva sob a forma da imitação. Mas é esse *mesmo* processo que conduz à fragmentação: se, por um qualquer motivo, começa a gerar-se uma quebra de confiança no *standard*, essa quebra de confiança pode acumular-se e conduzir a que se comece a abandoná-lo. Esse abandono ou fuga é um sinal para posteriores

indivíduos, os quais são incentivados a fazer o mesmo que os anteriores, isto é, a imitar-se e abandonar por sua vez o *standard*; é um processo que se reforça a si mesmo e que converge necessariamente para um *ponto fixo*, o ponto fixo oposto ao ponto fixo da ordem: o estado de desordem que corresponde à emergência de múltiplos *standards* ou programas incompatíveis entre si. Com efeito, se os indivíduos abandonam massivamente o *standard* dominante, eles fugirão competitivamente em *direcções independentes*. A razão para isso é simples e ela decorre da natureza concorrencial da economia, visto ser a única forma de evitar o esmagamento das margens de lucro que resultaria da existência de diversos *standards compatíveis*; neste caso, deixaria de existir um regime de externalidades em rede e verificar-se-ia a situação que a teoria neoclássica da economia prevê para produtos em concorrência pura: o preço tende a ser igual ao custo marginal do produto. Ao invés, ao produzir *standards incompatíveis* entre si, as empresas procuram que o seu *standard* particular se torne o *standard* universal, visando assim beneficiar do princípio de externalidades em rede, o qual implica que 'o vencedor ganhe tudo'. Noutros termos, ao incompatibilizar o *software*, as empresas geram de novo uma situação de desordem, que provoca uma nova competição com o objectivo da adopção universal de um *standard* inicialmente particular.

Um exemplo do processo que acabamos de descrever em termos teóricos é bastante relevante para a história dos programas em código-fonte aberto. Trata-se da emergência e fragmentação do sistema operativo (S.O.) UNIX. Esse S.O. começou a ser desenvolvido no início da década de 70 e era redistribuído acompanhado do respectivo código-fonte. Por volta de 1977, o lendário programador Bill Joy desenvolveu uma versão de UNIX distribuída nos termos da *Berkeley Software Distribution* (BSD), que é uma das primeiras licenças elaborada em espírito *open source*: essa licença permite a qualquer utilizador do programa fazer praticamente tudo o que quiser com ele, apenas se exigindo uma referência à Universidade que o originou. Mas a história de UNIX envolve a tragédia em que Stallman insiste: a partir dos anos 80, UNIX começou a evoluir para um regime proprietário, pois começaram a surgir *múltiplas* versões de UNIX que fechavam o código. Inicialmente, as diferenças entre essas diversas versões nem eram demasiadas, mas é um facto que o acumular dessas pequenas diferenças acabou por gerar enormes incompatibilidades, naturalmente não superáveis devido ao código se encontrar fechado. Nem existiria verdadeiramente desejo em as superar, visto o fabricante de cada versão procurar impor-se e assim 'ganhar tudo', isto é, aceder do particular ao universal da estandardização. Os fabricantes fugiram em direcções independentes, e essa violência competitiva antagónica ocasionou a inevitável fragmentação. O resultado final foi a existência de actualmente mais de 30 versões de UNIX. Pelo caminho, a Microsoft logrou impor o seu *standard*. Iremos vê-lo, mas também veremos que UNIX originou um outro sistema operativo com características bastante peculiares.

A tragédia de Stallman, bem como o seu gesto de recusa, está identificada em toda a sua raiz. Não estamos a afirmar que a situação dos anos setenta fosse de total ordem e estandardização. No entanto, existia um certo regime de ordem decorrente de uma comunidade de partilha que encerra em si mesma a superação da trágica fragmentação. Ao invés, se o *software* não é livre, é o regime de ordem que tende a desaparecer – tende a desaparecer uma comunidade como aquela de que Stallman fazia parte. Esse abandono apenas pode conduzir à desordem e competição violenta que, mais ainda do que sucede nos sectores tradicionais de actividade

económica, caracteriza necessariamente as tecnologias da informação. Pode mesmo formar-se um outro tipo de 'comunidade', termo que Stallman jamais aceitaria que se aplicasse a essa nova situação: a comunidade que gravita em torno de um empresa monopolista que exerce o *poder*. Essa não é uma 'comunidade', não é uma comunidade consistindo num espaço público de partilha. Como diz Stallman, o regime proprietário é:

a system based on dividing the public and keeping users helpless.

A natureza específica da produção de *software* agrava as tendências para a desordem das comunidades. Ela divide os indivíduos em função da dinâmica concorrencial na qual eles ficam inevitavelmente enredados. Ela reproduz adicionalmente as relações de poder que em muitos outros sectores se verificam. Em suma, o espírito concorrencial moderno, bem como a tendência para fragmentar a igualdade para a qual, no entanto, e paradoxalmente, esse espírito aponta, começou a invadir as regiões da produção intelectual

General Public License (GPL)

A ideia original de Stallman foi, em meados dos anos 80, procurar reconstruir uma comunidade num ambiente em que a fragmentação e a corrida competitiva se estavam a tornar a norma, e onde certos poderes absolutos não poderiam deixar de emergir. Fê-lo introduzindo uma ideia extremamente engenhosa, que consiste em conceber um *mecanismo* com a força da lei que fosse ao âmago do problema: um mecanismo que encerrasse em si mesmo a superação da desordem pela ordem. Stallman não lhe chama um 'mecanismo', apesar de na realidade ser um desde que os indivíduos a ele livremente adiram. O mecanismo é conhecido pelo nome que Stallman lhe deu: General Public Licence (GPL). É uma nova forma de licenciamento de *software* que se baseia em certas 'liberdades' ('freedoms'):

Freedom One is the freedom to help yourself, making changes to suit your own needs.

Freedom Two is the freedom to help your neighbor by distributing copies.

Freedom Three is the freedom to help build your community by making changes and publishing them for other people to use. If you have all of these freedoms, the program is free software for you.

Estas liberdades estão em pleno acordo com a moderna ideia de autonomia individual e horizontalidade total das relações. Não deverão existir quaisquer exterioridades, como o são certos poderes baseados em regimes proprietários e monopolistas de *software* que constrangem em absoluto o comportamento dos utilizadores. As liberdades são o quadro geral que levou Stallman à criação do regime de licenciamento GPL; este é uma sua consequência directa.

As liberdades enunciam, repita-se: (1) a liberdade em correr qualquer programa, (2) em o modificar e (3) em redistribuir o programa modificado e acompanhado do respectivo código-fonte. Mas a condição fundamental de GPL vem de seguida. Podemos enuncia-la assim:

O programa modificado tem de se encontrar sujeito às exactas condições acabadas de enunciar nas três prévias liberdades.

Quer isto dizer que é necessária uma aparente restrição às liberdades para que um programa continue *livre*. A consequência imediata consiste em que um programa modificado não pode de seguida ser fechado, tornado proprietário. A consequência seguinte consiste em que se impede que o moderno espírito competitivo e concorrencial se alastre ao domínio do *software*.

É esta última cláusula que distingue GPL do tipo de licença *open source* em sentido mais geral. Nesse último sentido, *open source* rege-se apenas pelas três condições iniciais, o que permite que um programa inicialmente aberto possa ser fechado. Exemplos são licenças como a já referida BSD, a licença Mozilla, etc[3]. Em certa medida, poder-se-ia sustentar que o *software* licenciado segundo as três primeiras condições deveria ser designado por *open source*, enquanto programas sujeitos a GPL seriam designados por *free software*.

Tal como Stallman insiste, *free software* nada tem a ver com *software* 'grátis', mas sim com '*software livre*'. A natureza aparentemente paradoxal da cláusula distintiva de GPL reside em ela procurar garantir uma certa liberdade através de uma restrição da liberdade - dado que impedir que um programa possa ser fechado pode ser ser entendido como constituindo uma tal restrição – e, desse ponto de vista, as licenças *open source* em sentido geral parecem envolver uma maior liberdade. Mas uma perspectiva inversa pode ser assumida, pois apenas a cláusula distintiva de GPL assegura que as próprias liberdades (1), (2) e (3) não podem desaparecer, e portanto ela implica uma liberdade mais vasta. Essa liberdade mais vasta é uma liberdade pública: a garantia que os programas permaneçam públicos, *livres*, no sentido de estarem num espaço aberto de que ninguém é dono.

A formulação de tipo recursivo da cláusula GPL indica que ela visa desencadear um processo propagativo e viral. Fechar o código significar interromper definitivamente um processo, pelo que apenas restringindo uma certa liberdade se pode garantir que as mais vastas liberdades públicas não desapareçam. Em particular, apenas GPL garante que o desenvolvimento de *software* se processe segundo linhas similares às do desenvolvimento científico. Em traços genéricos, a ciência é um processo público de discussão e crítica de hipóteses e argumentos. Ela é uma actividade crítica e auto-correctora, uma situação que, como veremos já adiante, tem uma exacta correspondência nos programas desenvolvidos enquanto *free software*. Pelo menos até há pouco tempo, qualquer ideia científica estava sujeita, de forma informal e espontânea, a um processo dentro do espírito de GPL. No caso do *software*, foi necessário enunciar esse tipo de licença de *copyright*, pois a tragédia a que Stallman alude começou aí a ocorrer. É uma licença que impõe no domínio do *software* a prática científica usual e que tem como uma consequência fundamental criar as mais amplas condições para o continuar indefinido da inovação.

Na verdade, seguramente parecerá absurdo que uma qualquer teoria científica crucial possa estar sujeita a um regime de *copyright* privado. Com a privatização do *software* ocorreu algo que muitos considerariam uma situação bizarra em ciência: imagine-se que a célebre equação $E=mc^2$ de Einstein estava submetida a um regime de propriedade intelectual, estipulando que qualquer sua utilização (ou mesmo o seu ensino) estaria

sujeita a condições de natureza comercial que tornavam restritivo o acesso ao enorme potencial de informação que ela encerra! Assim sendo, os programas devem estar disponíveis em conjunto com o seu código-fonte, dado ser essa é a única forma que garante plenamente a inovação e o crescimento. É um ponto cuja importância nos obrigará a realçá-lo de novo mais abaixo.

Open source e eficiência

A idéia fundamental de Stallman acerca do *software* é que ele deve indefinidamente permanecer 'free', no sentido de 'livre'. Ele é portanto bastante reservado acerca de um outro aspecto que começou a ganhar bastante impacto nos últimos anos: a eficácia do *software* desenvolvido em *open source* (ou GPL). Antes de voltaremos a analisar a relevância do código aberto na emergência de *standards*, devemos passar rapidamente em revista a questão da eficácia. Como programa de referência podemos utilizar o sistema operativo LINUX, que é indiscutivelmente um dos mais importantes programas desenvolvidos em regime GPL.

LINUX é um sucedâneo das diversas versões existentes de UNIX. Mais exactamente, o criador de LINUX, Linus Torvalds, utilizou parte de uma sistema operativo desenvolvido pelo próprio Stallman nos anos oitenta, GNU (*GNU but not UNIX*), e acrescentou-lhe o chamado *kernel*, tendo assim um S.O. completo no início dos anos noventa. O objectivo inicial de Torvalds nada tinha a ver com o desenvolvimento em grande escala de um S.O., mas apenas criar um que corresse no seu recentemente adquirido processador 386. No entanto, e como frequentemente ocorre nas novas tecnologias, aquilo que foi um mero episódio tornou-se imprevisivelmente um acontecimento maior. Ao disponibilizar na Internet o código-fonte de LINUX, Torvalds logrou atrair um número cada vez maior de programadores que, em regime voluntário e cooperativo, tornaram LINUX um S.O. extremamente estável e robusto que tem vindo a atrair cada vez mais utilizadores. A historia de LINUX tenderá a mostrar que programas desenvolvidos em *open source* são bastante mais fiáveis que as arquitecturas fechadas. Que razões para esse facto?

Uma razão fundamental parece consistir na *variedade* introduzida pelas diversas versões de LINUX. Essa variedade introduzida no código-fonte base de LINUX pode desde logo ter como consequência que o programa pode ser adaptado às necessidade específicas de um cliente individual ou empresarial. Essa liberdade existe em menor grau em programas fechados, pelo que ela é julgada por muitos como uma vantagem competitiva dos programas abertos. Mas a variedade também introduz um processo do tipo da selecção natural, visto o código acrescentado poder ser incessantemente testado, levando a que prevaleça a melhor variedade. A vantagem adaptativa de uma variedade tem naturalmente como sua condição que um grande número de indivíduos escrutinem os variantes propostos, pelo que a razão última e da robustez de um programa como LINUX foi a identificada por Eric Raymond, citando o próprio Torvalds[4]:

'given enough eyeballs, all bugs are shallow'

Isto é, a correção de erros ou efeitos indesejáveis de programação ('bugs') encontra-se assegurada desde que exista uma massa crítica suficiente de programadores que frequentemente analisem e corrijam o código, o que

seguramente se verifica no caso de LINUX. Essa actividade corresponde à actividade auto-correctora em ciência anteriormente referida, e é um dos componentes da estratégia que Raymond designou *como num bazar*, por oposição à programação das grandes empresas, designada por *tipo catedral*. Esta é *top-dowm*, aquela *bottom-up*, e representa efectivamente uma nova forma de conceber, e implementar, o desenvolvimento de *software*. Mas deve-se sempre não perder de referência um ponto fundamental: apesar da variedade introduzida, a fragmentação tende a não ocorrer em programas abertos, pois qualquer variação está publicamente accessível e, se se revelar bem adaptada a um certo ambiente, ela também tenderá a adaptada por outros produtores de variedade, pelo que a compatibilização permanece entre as múltiplas versões.

A compatibilização e eficácia é igualemente favorecida pela estrutura fortemente modular dos múltiplos componentes de LINUX. O resultado final (o 'todo') é notável, visto esse S.O. poder ser pensado como um sistema complexo adaptativo que exibe mecanismos de emergência. Se bem que desenvolvido modularmente, LINUX é mais que a soma linear das partes ou módulos que o compõem. Ele não pode deduzido das motivações específicas de cada indivíduo que contribui para o projecto - sejam altruísticas, de prestígio, técnicas, ou quaisquer outras. Um indivíduo pode estar a desenvolver programas de correio electrónico, um outro uma certa rotina específica, um terceiro procura optimizar a velocidade de processamento, etc., e o resultado é algo mais que a mera soma das diversas partes, mesmo que para isso tenha de existir algum trabalho centralizado de coordenação global: um S.O. completo dotado de inúmeras funcionalidades[5].

Open source e standards.

Já se deixou indicado que a eficácia de projectos *open source* incide na questão da estandardização. Devemos abordar de novo esse ponto crucial. Vimos que, em tecnologias da informação, tendem a formar-se monopólios que se tornam plataformas *standards* universais para inúmeras aplicações que sobre elas correm. Sublinhe-se agora que esses *standards* podem ser privados ou públicos. O mais importante exemplo de um *standard de facto* privado é talvez o sistema operativo Windows, propriedade da Microsoft Inc.. Através do mecanismo de retroacção positiva já referido em anterior secção, o Ms-Dos, e depois o Windows, atraíram novos utilizadores, o que constituiu um incentivo para que os programadores desenvolvessem mais aplicações para esse sistema operativo, o que representou um incentivo para a convergência de ulteriores utilizadores para essa plataforma, e assim sucessivamente até à monopolização de sistemas operativos para PC. É a emergência da ordem. Dito de um outro ponto de vista, a adesão a uma plataforma é muito diferente da forma como a teoria neoclássica da economia descreve o comportamento de um consumidor: segundo essa teoria, a escolha entre produtos concorrentes far-se-á, independentemente das escolhas dos outros indivíduos, segundo uma qualidade intrínseca do produto em questão e sob a constrição do orçamento disponível de cada um. Pelo contrário, em tecnologias da informação, a adesão a um *standard* não é um acto isolado do moderno indivíduo autónomo e soberano (o 'homo economicus'), mas sim um processo de imitação.

As consequências daí decorrentes podem ser analisadas a partir de um aspecto do recente caso Microsoft/USA. Uma das propostas punitivas da Microsoft foi a cisão da empresa em diversas 'Baby Bills' concorrentes na

área dos S.O. Neste caso, a fragmentação não seria um processo espontâneo, mas seria provocada pelo Estado. Contudo, o resultado final tem de ser necessariamente um único: os fabricante dos S.O. desenvolvidos pelas diversas empresas resultantes da cisão tenderiam a criar versões incompatíveis dos respectivos S.O. E isso, repita-se, pela simples razão de ser essa a única forma de evitar o esmagamento das margens de lucro que resultaria da existência de diversos S.O. *compatíveis*; no caso de S.O. compatíveis entre si, deixaria de existir um regime de externalidades em rede e verificar-se-ia a situação que a teoria neoclássica da economia prevê para produtos em concorrência: o preço tende a ser igual ao custo marginal do produto.

Assim sendo, e independentemente dos aspectos propriamente jurídicos do caso, a Microsoft teria razão em insistir nos perigos da fragmentação e, portanto, em recusar, por razões de princípio, a sua cisão. A fragmentação é um perigo permanente que pode ter consequências socio-económicas particularmente nocivas. Mas é neste preciso momento que a importância de *free software* se torna definitivamente clara e que os argumentos da Microsoft podem ser virados contra si mesmos. De facto, deve observar-se que, apesar de ser um espaço comum, o S.O. Windows não é um espaço público no sentido pleno desta última expressão. Um espaço público pode ser, simultaneamente, uma plataforma monopolista e uma plataforma cujo código-fonte está aberto: mesmo que as tecnologias da informação induzam necessariamente monopólios, não se segue que esses monopólios tenham de ser *privados*. Não é forçoso ser um adepto da socialização integral dos mercados para se ter essa posição. Os *standards* podem ser, excepcionalmente, públicos.

Ora, o que caracteriza o código-fonte em regime *open source*/GPL é precisamente ser um código público. E existem mesmo excelentes razões para pensar que os *standards* devem ser abertos. Um motivo, crucial, a ser analisado mais abaixo, prende-se com a questão da privacidade. Mas existe um outro que respeita à fragmentação. Em termos gerais, é evidentemente falso o argumento da Microsoft segundo o qual o perigo da fragmentação apenas é afastado quando os *standards* e os programas em geral são privados.

A passagem da desordem à ordem pode ser perfeitamente assegurada por *standards de facto* públicos. Um excelente exemplo é o TCP/IP, o qual é o *standard* de base da Internet. Concebido no início da década de setenta, ele é um *standard* público que satisfaz um princípio de *neutralidade*. Isto é, o TCP/IP satisfaz um princípio arquitectónico *end-to-end*, no sentido em que ele é indiferente ao conteúdo das mensagens ou formato de informação que transporta; por exemplo, não distingue uma página *web* de um *e-mail* ou de qualquer outra das milhares de aplicações que correm na Internet. É nesse sentido que ele foi uma condição fundamental do crescimento imprevisível que as redes de computadores vieram a ter[6]. É mesmo possível argumentar-se que *standards* abertos, diferentemente dos *standards* fechados, tenderão a favorecer a inovação[7]. No fundo, vimos ser esse um dos pontos essenciais da filosofia *open source*: apenas a abertura do código, a sua crítica e modificação, permite a melhoria permanente do *software*.

Apesar do desenvolvimento de *software* em regime *open source*/GPL introduzir variação nos programas, ele está longe de conduzir necessariamente à fragmentação. Pelo contrário. Como vimos, a passagem

da desordem (fragmentação) à ordem é marcada por uma intensa concorrência que conduz os diferentes competidores a produzir versões incompatíveis dos seus produtos em vista a 'ganhar tudo'. Essa tendência para a desordem pode ser evitada se o código for aberto, pois se um programador procurar desviar-se desse código referência (fechando código), os outros tenderão a permanecer fiéis ao código que se encontra aberto, o que ainda é mais reforçado se a licença for GPL Mais em geral, se um programa licenciado nos termos de GPL introduzir uma inovação, outros fornecedores desses mesmo programa (e.g., Linux) tenderão a introduzir também essa inovação, pois todos têm acesso ao código. Ao mesmo tempo, algumas incompatibilidades de aplicações face a um *standard* tenderão a ser harmonizadas mais facilmente quando o código está acessível a todos[8]. A situação daí decorrente é extremamente interessante, pois, por um lado, pode existir competição entre diversos fornecedores de um mesmo programa (como sucede em torno de LINUX), só que, por outro, as dinâmicas económicas são, neste caso, radicalmente diferentes das que geram regimes proprietários monopolistas. De forma singular, o código-fonte aberto acaba por induzir uma dinâmica conforme aos modelos clássicos da economia: a competição far-se-á em termos de qualidade e preço do serviço prestado.

Portanto, e concluindo, os perigos de fragmentação são *menores* em código aberto que em código fechado. Sob esse aspecto, pode agora comparar-se a evolução de UNIX com a de LINUX. Como vimos, UNIX sofreu um progressivo processo de fragmentação a partir dos anos oitenta. Ao invés, LINUX é licenciado em regime GPL, donde a publicidade do seu do código surgir como uma espécie de referencial que tende a impedir a fragmentação. Naturalmente que isso não representa uma garantia absoluta, e o mecanismo exibe aqui as suas potenciais limitações, pois pode ser necessária, apesar de tudo, uma nova exterioridade representada pela figura de Linus Torvalds e colaboradores próximos, os quais 'oficializam' a versão *standard* de LINUX. É assim possível que a comunidade se mantenha e cresça, enquanto a competição ocorre nas suas margens, isto é, nas múltiplos serviços de apoio à instalação, manutenção e serviço ao cliente utilizador do sistema operativo.

Código aberto: valores e privacidade

Foram até agora analisadas certas consequências decorrentes do conceito de código-fonte aberto: estandardização, robustez, inovação e liberdade, no sentido que Stallman confere a este conceito. Existe ainda uma outra que não é menos decisiva, e que respeita a certos direitos julgados fundamentais, em particular os de privacidade. Desse ponto de vista, não é irrelevante que o código-fonte seja aberto - acessível a todos e passível de ser modificado - ou, pelo contrário, fechado.

Isso é tanto mais importante quando as questões respeitando a privacidade adquirem um conteúdo específico quando inseridas nos ambientes das tecnologias da informação. É conhecido que um dos traços da evolução da *World Wide Web* e de outras redes durante os últimos anos foi o aparecimento de um número crescente de mecanismos de *software* que tornam a rede extremamente vulnerável às estratégias de controlo, de invasão da privacidade, de limitação da liberdade de expressão , etc. Podem ser instaladas rotinas em programas comuns como os *browsers* que permitem filtrar, catalogar e seleccionar a informação de forma muitas vezes invisível aos olhos o utilizador. *Sítes* de natureza comercial

desenvolvem poderosas ferramentas de *software* que permitem coligir e relacionar uma massa enorme de dados acerca das preferências dos potenciais clientes. Independentemente dos exemplos concretos, o ponto decisivo é a sublinhar que a vigilância e o controlo são possibilidades intrínsecas das tecnologias da informação

A fim de melhor avaliar essa nova situação, deve, em primeiro lugar, ter-se presente que as tecnologias da informação se distinguem da comunicação oral e escrita por permitirem a memorização de quantidade gigantescas de informação (o que a escrita também faz) e, sobretudo, relacioná-las *automaticamente* (o que o texto impresso já não faz). Isso não é ocasional, pois esse tipo de tecnologias pode precisamente ser *definido* como a capacidade de relacionar automaticamente informação existente em memória – em sentido genérico, é precisamente nisso que consistem todas as operações de um computador. Assim sendo, as tecnologias da informação assentes permitem, de forma intrínseca, que os programas (cuja operação é precisamente relacionar informação) estejam escritos de tal modo que princípios de privacidade sejam automaticamente violados, com as eventuais utilizações daí decorrentes.

Em segundo lugar, existe uma outra característica das tecnologias da informação que as distingue claramente de outros tipos de tecnologias, tais como as tecnologias da matéria e da energia. É aproximadamente rigoroso, pelo menos num sentido relativamente restrito, afirmar-se que as tecnologias da informação são constituídas por *software*, o qual pode ser implementado em máquinas, e que as isso distingue de outras tecnologias. São, como se diz, tecnologias da ... informação. Essa distinção pode estar associada a uma outra

importante distinção. As tecnologias da informação envolvem valores socio-políticos como uma sua possibilidade intrínseca. Escrevemos ‘intrínseca’ para sublinhar o facto de esses valores poderem estar presentes na própria concepção da tecnologia. Eles estão literalmente nela escritos. Estão escritos no seu código-fonte. Para a ideia se tornar mais clara pense-se, por contraposição, nas tecnologias da matéria e da energia. Um frigorífico, um carburador ou uma máquina de café não envolvem quaisquer valores na sua arquitectura. Não estamos a afirmar que os frigoríficos não foram concebido com uma certa função (conservar os alimentos) e que as pessoas não possam julgar que os frigoríficos são indispensáveis à felicidade humana. Só que isso é um juízo que um observador exterior faz acerca da tecnologia em questão. Esse juízo não está nela escrito. De facto, pelo menos até agora, pouca coisa está escrita na arquitectura de um frigorífico. Mas já assim não sucede com as tecnologias que, em certo sentido, mais não são que informação sob a forma do código-fonte que as governa. Aí podem ser escritas coisas. Valores [9].

Quando afirmamos que essas tecnologias envolvem valores no seu *design* não estamos a afirmar *que tipo de valores* elas envolvem. Elas envolvem a possibilidade indeterminada de valores mas não envolvem necessariamente este ou aquele valor específico. São *instáveis* por relações a valores específicos, no sentido em que um ponto fixo instável é indiferente à sua actualização num dos pontos fixos estáveis a que ele dá origem. Ora, um *standard* permanece neutral do ponto de vista dos valores quando o seu código-fonte é público e aberto. Um *standard* é um monopólio, e um monopólio significa a inexistência de competição. É uma situação potencialmente problemática do ponto de vista dos valores, pois o

monopolista tem a capacidade em determinar as funcionalidades que o *standard* executa. A importância de o código-fonte dos programas estar aberto reside em que assim se tornam visíveis os mecanismos de relacionamento e de memorização da informação que neles eventualmente existam. Ou seja, em programas abertos qualquer mecanismo de controlo estará inteiramente exposto e, visto possuir-se o código-fonte, este poderá ser modificado de forma a eliminar o módulo de programação indesejado. Os programas públicos e abertos participam de um certo ideal de *transparência* que tem precisamente como efeito garantir a privacidade e a autonomia individual. Noutros termos, aquilo que é público e partilhado por *todos* não pode atentar a liberdade e individualidade de *cada um*. Assim sendo, o movimento *open source* tem como consequência última que a regulação da vigilância seja realizada num *espaço público*, o espaço público constituído pelo código aberto sem regime proprietário. É nessa prevalência de certos valores colectivos em torno dos quais as comunidades que se encontram unidas que a filosofia de Richard Stallman insiste, sem que ela seja por um instante julgada incompatível com o que ele estima serem as *liberdades* ('freedoms') fundamentais.

---

[1] Cf. W. B. Arthur, *Increasing returns and Path dependance in the Economy*, University of Michigan Press, Ann Arbour, 1994.

[2] Cf. N. Economides e F. Flyer, 'Compatibility and Market Structure for Network Goods', Discussion Paper EC-98-02, Stern School of Business, N.Y.U., 1998, in: http://www.stern.nyu.edu/networks/98-02.pdf

[3] Para uma análise dos diversos tipos de licenças *open source*, cf. Bruce Perens, 'The Open Source Definition' in Chris DiBona, Sam Ockman, and Mark Stone, (Eds.), *Open Sources.*

Voices from the Open Source Revolution, O'Reilly & Associates, 1999. pp., 171-188.

[4] Cf. Eric S. Raymond. *The cathedral and the bazaar*, 1998, in: http://tuxedo.org/~esr/writings/cathedral.

[5] Cf. J. Boyle*, The Second Enclosure Movement and the Construction of the Public Domain*, in: http://www.law.duke.edu/pd.

[6] Para uma análise do TCP/IP no contexto da história da Internet, cf. A. Machuco Rosa, *Internet - Uma História*, Edições Universitárias Lusófonas, Lisboa, 2003.

[7] Cf. L. Lessig, L, *The Future of Ideas - the fate of the commons in a connected world*, Random House, New York, 2001.

[8] Cf. Working group on Libre Software, 'Free Software / Open Source: Information Society Opportunities for Europe?', Versão 1.2, 2000, in: http://eu.conecta.it.

[9] Cf. L. Lessig, *Code and Other Laws of Cyberspace*, Basic Books, New York, 1999.