

Elaborato intermedio

Software Libero e Open Source

Marco Faré

prof. Fiorenzo Scaroni

Istituto di Tecnologie della Comunicazione
Facoltà di Scienze della Comunicazione

Università della Svizzera Italiana

SE 2002

Indice:

1. Introduzione	3
2. Gli inizi: Stallman e il Software Libero	4
2.1. La preistoria del Software Libero	4
2.2. I primi software: il progetto GNU	5
2.3. La definizione della licenza GNU GPL	5
2.4. La Free Software Foundation (FSF)	6
2.5. Il Manifesto GNU	6
2.6. Linux	8
2.7. La qualità tecnica	8
2.8. Il futuro del Software Libero	10
2.8.1. I pericoli	10
2.8.2. I rimedi: la diffusione della cultura del Software Libero	11
3. L'Open Source	12
La definizione di Open Source	12
3.2. Le differenze tra Software Libero e Open Source	14
4. Oggi e domani	17
4.1. L'Open Source nel business	17
4.2. L'Open Source nella scuola	18
4.3. L'Open Source nella Pubblica Amministrazione	19
4.4. In Svizzera: da ch/open/ a GuglielmoTux	20
4.5. L'attacco all'Open Source: la sicurezza	21
4.6. L'Open Source al di fuori del software	22
4.7. Le aziende	23
5. Conclusione	24
6. Glossario	25
6.1. Hacker	25
6.2. Il termine <i>free</i>	25
6.3. La licenza d'uso	25
6.4. Codice sorgente e codice eseguibile	25
6.5. L'etica hacker	26
7. Bibliografia	28
7.1. Libri	28
7.2. Articoli	28
7.3. Siti web	30

1. Introduzione

Questo elaborato vuole essere una panoramica descrittiva sul software Open Source e sul Software Libero. Ho cercato di capire come è nata e come si è sviluppata la corrente filosofica del Software Libero, poi sfociata in un modello di sviluppo e in un modo di fare business che sta influenzando non solo il panorama dell'IT, ma tutto il mondo economico.

Soprattutto, ho cercato di capire, studiandone la storia, cosa rende interessante il movimento dell'Open Source. L'attenzione all'Open Source è evidente: molte organizzazioni private, aziende e anche amministrazioni pubbliche se ne interessano. Le mie ricerche in Internet mi hanno fornito sempre numerosissimi articoli e siti dedicati.

Il mio interesse è in parte tecnico, ma è soprattutto un interesse dedicato al fenomeno nella sua interezza, e ai suoi risvolti più imprevedibili. Richard Stallman, fondatore della Free Software Foundation, probabilmente era conscio del fatto che la sua filosofia del Software Libero poteva applicarsi anche a molti altri ambiti. In particolare viene esplicitamente attaccato il diritto d'autore: Stallman non parla di diritto d'autore (copyright) ma di permesso d'autore (copyleft), cioè un metodo che invece di privatizzare il software, permette di mantenerlo libero. In altre parole, Stallman lascia il diritto di copiare nelle mani dell'utente. Il concetto di copyleft applicato al software permette, fra le altre cose, di cambiare tipo di computer (per esempio, passare da PC a Mac) senza dover pagare nuovamente per usare lo stesso programma. Questo, secondo Stallman, è un diritto e una libertà legittima, che viene violata dalle licenze proprietarie.

In un periodo di grandi cambiamenti, non solo economici, il software Open Source è una filosofia nuova in un ambito (quello dell'informatica) che influenzerà pesantemente il nostro comportamento di domani.

L'ideologia alla base dell'Open Source si applica al di fuori dell'informatica e spinge molte persone ad avere voglia di libertà. Dopo anni di predominio aziendale, la massa di consumatori di software si sente sempre più una comunità che ha dei diritti e li vuole far valere, soprattutto quando si parla di formazione scolastica e di cosa pubblica.

2. Gli inizi: Stallman e il Software Libero

2.1. La preistoria del Software Libero

Per comprendere il software Open Source dobbiamo risalire alla fine degli anni Settanta, quando la programmazione cessò di essere un'esclusiva del mondo accademico. Fino a quel momento, gruppi di accademici avevano realizzato programmi scambiandosi idee e trucchi secondo i principi che da anni regolano il mondo scientifico: cooperazione e libertà di scambio di idee e di informazioni. Questi sono gli stessi principi che stanno alla base della libertà di pensiero e di espressione.

Software e idee hanno molto in comune. Come accade con l'evoluzione delle idee, anche per il software il processo di crescita dipende in gran parte dalla sua libera diffusione. Inoltre anche il software, come le idee, è immateriale e può essere trasmesso o riprodotto facilmente.

All'inizio degli anni Ottanta alcune aziende produttrici di software iniziarono a imporre delle restrizioni sull'uso dei sistemi operativi. I gruppi di hacker¹ si sciolsero e gli accademici che vi erano coinvolti vennero assunti dalle imprese private e costretti a firmare accordi di non diffusione. Sono stati costretti a mantenere il segreto sul software che producevano.

Uno di questi hacker non ci stette. In quegli anni Richard Stallman, mosso dalla delusione di non poter più sviluppare software in collaborazione con una comunità, definì il concetto di Software Libero (in inglese Free Software²).

L'idea era quella di formalizzare i diritti dell'utente nei confronti del software. Stallman descrisse quattro libertà di cui l'utente ha diritto di usufruire:

- **Libertà 0:** la libertà di eseguire il programma per qualunque scopo.
- **Libertà 1:** la libertà di studiare il funzionamento del programma, e di adattarlo alle proprie esigenze. L'accesso al codice sorgente³ ne è un prerequisito.
- **Libertà 2:** la libertà di redistribuire copie in modo da aiutare il prossimo.
- **Libertà 3:** la libertà di migliorare il programma, e di distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio. L'accesso al codice sorgente ne è un prerequisito.

Per mettere in pratica la sua idea di garantire queste libertà e per poter continuare a sviluppare software come aveva sempre fatto⁴, Stallman si rese subito conto di dover partire da un sistema operativo libero. Ma un sistema operativo con questa caratteristica non esisteva. Quindi, nel 1984, lasciò il Massachusetts Institute

¹ si veda il glossario (cap. 6) per una definizione di hacker

² si veda il glossario (cap. 6) per una precisazione sul termine *free*

³ si veda il glossario (cap. 6) per una spiegazione su codice sorgente e codice eseguibile

⁴ si veda il glossario (cap. 6) per una breve spiegazione sull'etica hacker

of Technology (MIT), dove lavorava, e diede vita a un progetto che chiamò GNU (GNU is Not Unix, pronunciato “ghniu”) e che aveva come obiettivo la creazione di un sistema operativo libero e compatibile con Unix.

2.2. I primi software: il progetto GNU

Il primo software per GNU venne scritto partendo completamente da zero ed era un compilatore multilinguaggio e multiplatforma, oggi noto come Gcc (GNU C Compiler, oggi GNU Compiler Collection). In seguito Stallman realizzò GNU Emacs, un editore di testi che nel 1985 era quasi completamente utilizzabile. Alcune persone mostrarono interesse in GNU Emacs e chiesero a Stallman di utilizzarlo. Lui lo mise a disposizione su un server ftp del MIT (prep.ai.mit.edu), ma, poiché all’epoca molte persone non avevano accesso a Internet, decise di spedire a pagamento una versione su nastro del software a chi lo avesse richiesto. Sin dall’inizio era molto chiaro che libero (free), in questo caso, non significa gratis. È altrettanto chiaro che Stallman, nonostante le posizioni fortemente idealiste, non osteggia il commercio.

2.3. La definizione della licenza GNU GPL

Stallman incontrò qualche ambiguità nel termine libero (free). Oltre alla confusione sulla gratuità, si presentò un altro problema, evidenziato dal caso dell’X Window System. Questo software venne sviluppato dal MIT e distribuito come Software Libero con una licenza che permise ad alcune società informatiche di integrare l’X Window System nei loro software Unix proprietari e di distribuirlo in forma binaria. Queste copie di X non erano più Software Libero. La situazione che si creò era la seguente: X era un Software Libero, stando a quanto dicevano gli sviluppatori. Gli utenti però non usufruivano di questa libertà. Gli autori dell’X Window System non erano preoccupati da questa situazione perché il loro scopo era il successo, cioè avere tanti utenti. Stallman invece si preoccupava non tanto della quantità degli utenti, quanto del fatto che gli utenti fossero liberi.

Per perseguire lo scopo di GNU (offrire libertà agli utenti) venne definita una licenza che impedisse di trasformare GNU in software proprietario. Il principio su cui si basa questa licenza è noto come copyleft.

Il concetto base della licenza è il seguente: *«Il succo dell’idea di permesso d’autore consiste nel dare a chiunque il permesso di eseguire il programma, copiare il programma, modificare il programma, e distribuirne versioni modificate, ma senza dare il permesso di aggiungere restrizioni. In tal modo, le libertà essenziali che definiscono il “free software” sono garantite a chiunque ne abbia una copia, e diventano diritti inalienabili..»*⁵

⁵ **Richard M. Stallman** in *Open Sources, Voci dalla rivoluzione Open Source* a cura di **Chris DiBona, Sam Ockman e Mark Stone**, Milano, Apogeo, 1999; disponibile su web presso <http://www.apogeeonline.com/openpress/libri/545/> (consultato il 12 giugno 2002)

In questo senso, il programma che deriva da qualsiasi aggiunta o combinazione con un programma protetto da permesso d'autore, è anch'esso un programma libero protetto da permesso d'autore.

Il concetto di permesso d'autore è concretizzato nella GNU GPL (GNU General Public License, licenza pubblica generica GNU). Esistono altre licenze per il Software Libero, ma hanno tutte qualche diversità dalla GNU GPL. La GNU GPL è la licenza ideata da Richard Stallman e pertanto è l'unica che ne rispecchi interamente la filosofia. Un'altra licenza che vale la pena citare è la licenza BSD. Essa permette a chi modifica un programma di ridistribuirlo con qualsiasi licenza, vale a dire che può imporre le restrizioni che preferisce.

2.4. La Free Software Foundation (FSF)

Nel 1985 Stallman si rese conto che era il momento di cercare nuovi finanziamenti, dal momento che l'interesse per il progetto GNU aumentava. Venne fondata la Free Software Foundation (FSF)⁶, un'organizzazione senza fini di lucro che ha come scopo lo sviluppo del Software Libero. Tra le prime attività vi era la distribuzione dei nastri di Emacs e di altro Software Libero (GNU e non GNU) e la vendita di alcuni manuali liberi (manuali liberamente modificabili che accompagnano il Software Libero). Ora la FSF vende CD-ROM di Software Libero, manuali e distribuzioni personalizzate di Software Libero. La Free Software Foundation è attualmente presente in molti Paesi. In Europa esiste la Free Software Foundation Europe⁷, alla quale fanno capo molte organizzazioni nazionali.



Logo di GNU

Con la creazione della Free Software Foundation diventa sempre più chiaro che la filosofia del Software Libero non è contro il commercio, ma pretende il rispetto delle libertà dell'utente. È però abbastanza evidente il fatto che la spinta a questo tipo di filosofia sia provocata da pratiche commerciali spregiudicate.

2.5. Il Manifesto GNU

Il manifesto GNU⁸ venne scritto agli inizi del progetto GNU da Richard Stallman. Esso serviva a chiedere sostegno. Ora il suo valore è storico e non viene più aggiornato. Il principio base su cui poggia è questo: se mi piace un programma, lo condivido con altre persone a cui piace. Per i programmatori usare software proprietario (o accettare clausole di non divulgazione) è un disonore, perché copiare programmi o parte di essi è naturale. Quindi, secondo Stallman, copiare programmi non dovrebbe essere vietato. Se questa conclusione può sembrare discutibile, bisogna considerare che, di fatto, la tecnologia si presta a copiare

⁶ <http://www.fsf.org/>

⁷ <http://www.fsfeurope.org/>

⁸ **Richard M. Stallman**, *Manifesto GNU*, <http://www.pluto.linux.it/pluto/ManifestoGNU.html> (consultato in data 28 aprile 1999)

software. Questo è vero soprattutto negli anni Ottanta, quando non erano ancora diffusi tutti i trucchi utilizzati oggi per proteggere il software commerciale.

I valori che spingono i programmatori ad aiutarsi tra loro sono cameratismo e amicizia. Secondo Stallman le pratiche commerciali in vigore dall'inizio degli anni Ottanta in ambito software non permettevano né amicizia né cameratismo. Quelle pratiche spingevano alla concorrenza, inducendo a considerare gli altri programmatori come dei nemici. Il Software Libero permette ai programmatori di condividere e di essere in armonia con gli altri programmatori, portando alla felicità. Si forma così una comunità di programmatori fortemente basata sul volontariato e l'altruismo. Il Manifesto GNU è una vera implementazione dell'etica hacker e mostra i lati buoni della globalizzazione.

Le stesse cose valgono anche per gli utenti. Innanzitutto il Software Libero permette agli utenti di condividere i programmi con gli amici e i colleghi nel rispetto delle leggi. Esso inibisce la duplicazione di sforzi e quindi lo spreco di energie, portando a un avanzamento dello stato dell'arte globale. Inoltre gli utenti non sono più costretti ad affidarsi a una sola ditta.

La scuola che usa Software Libero dispone di un ambiente più educativo perché gli studenti possono studiare e migliorare il software senza restrizioni. Stallman cita l'università di Harvard, dove è vietato l'uso di software per il quale non è disponibile la sorgente.

Ritroviamo ancora oggi le obiezioni che Stallman anticipa nel 1984. La prima riguarda la mancanza di supporto, ma è facile replicare dicendo che, data la disponibilità del codice sorgente, chiunque può offrire supporto, non solo l'azienda che ha prodotto il software. Oggi questa possibilità origina un nuovo business.

L'obiezione principale, soprattutto da parte degli addetti ai lavori, riguarda il compenso dei programmatori. Oggi sappiamo che il business basato sul Software Libero è possibile, e quindi lo è anche un profitto economico. Le società di consulenza e manutenzione e di istruzione sono una realtà, e i produttori di hardware, per esempio, hanno bisogno di programmatori che scrivano i driver per i loro prodotti. Oppure utenti (aziende) che necessitano di un particolare programma (o di un adattamento di un programma libero) potrebbero pagare per averlo. Questo è possibile perché il concetto di Software Libero non vieta di farsi pagare per realizzare un programma. Però nel Manifesto GNU si sostiene che imporre restrizioni per farsi pagare è pericoloso, perché limita i modi in cui il programma può essere usato. Inoltre Stallman aggiunge che *«la creatività può essere un contributo a favore della società, ma solo fin quando la società è libera di usarne i risultati.»*⁹ I programmatori, sempre secondo Stallman, sono come i musicisti: lo farebbero anche gratis.

Un'altra obiezione frequente ha come tema il controllo del prodotto della propria creatività. Nel Manifesto GNU viene spiegato che il diritto alla proprietà intellettuale non è intrinseco, ma è stato introdotto dai legislatori per favorire la diffusione della conoscenza proteggendo gli autori delle idee. Questo aveva senso all'epoca in cui il diritto d'autore fu introdotto, quando la scienza progrediva lentamente e copiare era

⁹ **Richard M. Stallman**, *Manifesto GNU*, op. cit.

costoso. Il principio che stava alla base di brevetti e diritto d'autore era favorire la società. Dobbiamo chiederci se il sistema di brevetti e di diritto d'autore in uso attualmente stia favorendo la società oppure sia obsoleto.

2.6. Linux

Quel che mancava a GNU era il cuore del sistema: un kernel. Infatti lo sviluppo del kernel progettato (GNU Hurd) aveva incontrato ritardi dovuti a difficoltà tecniche. Nel 1991 lo studente finlandese Linus Torvalds sviluppò un kernel compatibile con Unix, lo chiamò Linux e lo distribuì con la licenza GNU GPL. Nel 1992, dopo un intenso lavoro di integrazione, iniziò la distribuzione di GNU/Linux: un sistema operativo libero e completo. Da allora la diffusione del Software Libero è cresciuta e al giorno d'oggi esistono molte aziende che vendono Linux sotto forma di confezioni contenenti manuali e cd-rom con software precompilato, dette distribuzioni. Ci sono anche aziende che forniscono esclusivamente supporto. Il mondo aziendale, con alcune eccezioni, non è però schizzinoso quanto Stallman e mescola Software Libero con software commerciale. L'avvento di Linux ha reso il Software Libero un fenomeno di massa.

2.7. La qualità tecnica

GNU non ha mai preteso di essere un sistema tecnicamente perfetto. I vantaggi che Stallman propone sono di tipo sociale ed etico. GNU rispetta le libertà degli utenti e permette loro di cooperare. Nonostante ciò, la qualità tecnica di GNU fu abbastanza elevata sin dall'inizio. Le ragioni sono molte: innanzitutto era un sistema moderno, costruito sulle macchine in uso all'epoca, che avevano caratteristiche diverse da quelle su cui erano stati sviluppati i primi Unix. Inoltre i programmatori che partecipavano al progetto applicarono e applicano tutt'ora le regole di buona programmazione.

Il terzo fattore è una peculiarità propria di GNU: quello che Eric Raymond¹⁰ ha chiamato stile bazaar. Raymond è un hacker che si è occupato di GNU dall'inizio. Convinto della qualità del metodo di programmazione applicato a GNU (nessun approccio centralizzato), era però convinto che tale metodo non avrebbe potuto funzionare per progetti di elevata complessità. Un sistema operativo, per esempio, andava costruito con un approccio centralizzato che lui chiama stile cattedrale ed è attualmente utilizzato nelle grandi multinazionali del software.

Nel 1991 arrivò Linux, un sistema operativo nato e cresciuto in spirito bazaar. Raymond analizzò attentamente il fenomeno Linux a partire dal 1993 e qualche anno dopo avviò un test con fetchmail, un Mail Transfer Agent (MTA), grazie al quale poté sperimentare di persona il successo del metodo bazaar.

Ciò che aveva colpito Raymond era lo stile di sviluppo di Linus Torvalds: diffondere le release presto e spesso e delegare il più possibile. Linux si rivelava un sistema sovversivo, lontano dalle cattedrali da costruire in reverente silenzio e vicino alla promiscuità e alla confusione di un grande bazaar. La cosa

¹⁰ Eric S. Raymond, *La cattedrale e il bazaar*, 22 novembre 1998,

<http://www.apogeeonline.com/openpress/doc/cathedral.html> (consultato l'8 maggio 2002)

sorprendente era che Linux sembrava paragonabile a un agente indipendente in grado di auto-correggersi e non cadeva in preda della confusione. Anzi, migliorava a una velocità impensabile.

Tra i punti tipici dello sviluppo a bazaar, Raymond evidenzia l'interesse dello sviluppatore (la necessità è la madre di tutte le invenzioni, è il proverbio citato ne *La cattedrale e il bazaar*) e l'abilità nel riutilizzare il lavoro già fatto. Linus Torvalds era partito da Minix, Raymond stesso aveva iniziato con un software che si chiamava Popclient. Questo è il metodo che si usa nella scienza e che determina l'importanza della diffusione del codice sorgente anche se l'utente finale raramente è in grado di utilizzarlo. La presenza di una comunità in grado di usare il codice sorgente, di controllarlo e studiarlo è fonte di garanzia anche per l'utente inesperto. Allo stesso modo la comunità scientifica garantisce la validità di teorie che non sono verificabili da chiunque.

Le qualità del leader del progetto Open Source sono essenziali alla riuscita del progetto stesso. La prima di queste qualità è il saper riconoscere il momento in cui si perde l'interesse per il programma. In questo caso, l'ultimo dovere è quello di passarlo a un successore competente.

Nello stile bazaar gli utenti sono essenziali: essi possono diventare co-sviluppatori. Per questo è importante ricompensarli con aggiornamenti costanti ottenuti grazie al loro lavoro. In questo modo si sentiranno stimolati dalla soddisfazione di aver preso parte all'azione.

Il problema dei bug è fondamentale nello sviluppo di software. Linux ha avuto successo anche grazie al fatto che Linus Torvalds ha applicato da subito quella che poi Raymond ha chiamato "Legge di Linus": dato un numero sufficiente di occhi, tutti i bug vengono a galla. La differenza fondamentale del modello bazaar con il modello cattedrale, è che la scoperta dei bug, nel primo, avviene molto più rapidamente. Applicare la legge di Linus significa rilasciare gli aggiornamenti presto e spesso, senza vergognarsi se ci sono dei bug. Il modello cattedrale, d'altro canto, tende a nascondere i bug con release molto distanti fra loro e un intenso lavoro di analisi e correzione tra una release e l'altra.

La legge di Linus si ritrova anche in un fenomeno sociologico noto come "effetto Delfi", secondo cui l'opinione media di un gruppo di osservatori equamente esperti (o equamente ignoranti) si rivela parametro più affidabile di quella di un solo osservatore scelto casualmente in quel gruppo.

Per applicare lo stile di Linus Trovalds e tentare di ripetere il successo, Raymond ha applicato i seguenti principi al suo progetto di fetchmail:

1. diffusione presto e spesso delle nuove release (intervallo di pochi giorni);
2. ampliamento della base di tester;
3. mantenimento della base di tester con messaggi simpatici;
4. ascolto dei tester, ringraziandoli per ogni suggerimento.

Questa strategia ha permesso a Raymond di avere molti report sui bug, spesso completi di soluzioni.

Anche la semplicità nel codice è un fattore di successo: se il codice è complesso, la gente non troverà interessante cercare gli errori.

Al termine del suo esperimento, Eric Raymond era in grado di isolare le precondizioni per lo stile bazaar. Innanzitutto è necessario avere qualcosa da cui partire: è molto difficile avviare un progetto Open Source basato sullo stile bazaar da zero. Il programma iniziale non deve per forza essere perfetto, ma deve contenere

la promessa di evolversi in qualcosa di utile e ben fatto. Il design di base dovrebbe essere forte e attraente e chi guida il progetto deve avere un elevato livello di intuizione e saper riconoscere le buone idee progettuali degli altri.

Lo stile bazaar è in contraddizione con la legge di Brooks che dice: aggiungendo altri sviluppatori ad un progetto in ritardo, lo si fa tardare ancora di più. Questa legge è ritenuta vera perché i costi della complessità e delle comunicazioni di un progetto aumentano esponenzialmente con il numero degli sviluppatori coinvolti, mentre il lavoro cresce linearmente. Apparentemente ciò dipende dalle gelosie territoriali degli sviluppatori nei confronti del proprio codice.

Raymond propone un'alternativa alla legge di Brooks: *«Stabilito che il coordinatore dello sviluppo abbia a disposizione un medium almeno altrettanto affidabile di internet, e che sappia come svolgere il ruolo di leader senza costrizione, molte teste funzionano meglio di una.»*¹¹ L'accento viene posto sia sulla possibilità di comunicare facilmente grazie a internet sia sulla capacità del leader di far accadere le cose senza imporle. A questo proposito Linus Torvalds ha detto: *«Cerco di gestire senza prendere decisioni e lasciando che le cose accadano in modo naturale. È così che si ottengono i risultati migliori.»*¹²

2.8. Il futuro del Software Libero

2.8.1. I pericoli

I fattori che possono mettere in pericolo la diffusione del Software Libero sono quattro, come Stallman stesso spiega:

1. **Hardware segreto:** se i costruttori di hardware mantengono segrete le specifiche dei loro prodotti, non sarà possibile scrivere driver liberi e quindi non sarà possibile usarli su GNU/Linux. Per combattere questa minaccia si può ricorrere al reverse engineering (ricostruire le specifiche partendo dal prodotto), ma è una via tecnicamente complessa e con notevoli ostacoli legali. Oppure si può scegliere di usare solo hardware supportato da programmi liberi.
2. **Librerie non libere:** i programmatori che scrivono Software Libero e utilizzano le funzionalità di librerie non libere cadono in una trappola perché il programma creato non potrà essere utilizzabile a meno di non utilizzare anche software proprietario (le librerie). Casi celebri sono la libreria Motif, che ora è stata sostituita dalla libreria libera LessTif, e la libreria Qt, su cui si basa l'ambiente grafico KDE. KDE ebbe un gran successo perché fu il primo ambiente grafico per Linux completamente gratuito, ma era basato su una libreria proprietaria e pertanto non poteva essere definito né Software Libero né poteva fregiarsi del marchio Open Source, nonostante i tentativi in questo senso del team di sviluppo KDE. L'accordo trovato con TrollTech, proprietaria di Qt, consentiva l'accesso gratuito alla libreria e TrollTech si era impegnata a rilasciare Qt come Software Libero nel caso in cui

¹¹ Eric S. Raymond, *La cattedrale e il bazaar*, op. cit.

¹² Linus Torvalds, David Diamond, *Rivoluzionario per caso. Come ho creato Linux (solo per divertirmi)*, Milano, Garzanti, 2001

l'azienda fallisse o intendesse disfarsi della libreria. Quest'accordo, chiamato KDE Free Qt Foundation, non soddisfò Stallman. Le pressioni della comunità spinsero infine TrollTech a rilasciare la libreria Qt con una licenza libera, ma Stallman continua a preferire l'alternativa libera originale: l'ambiente grafico GNOME. Questo episodio mostra le divergenze di opinione e i tentativi di approfittare della notorietà del Software Libero.

3. **Brevetti sul software:** sono sicuramente il maggior pericolo per il Software Libero. I brevetti possono rendere inaccessibili algoritmi e funzionalità, impedendo lo sviluppo di Software Libero e facendo correre rischi legali a chi sviluppa software. Una limitazione di questo tipo è difficile da aggirare: si può tentare di discutere la legittimità di un brevetto oppure cercare strade alternative per ottenere lo stesso risultato. Ma il modo migliore per contrastare questa minaccia è la diffusione della cultura della libertà. Esistono numerose organizzazioni che si battono contro i brevetti sul software.
4. **Documentazione libera:** insieme al software, la documentazione riveste un ruolo chiave nella diffusione di un sistema operativo e di programmi applicativi. Al momento il Software Libero è carente di documentazione. In futuro sarà importante realizzare molta documentazione di buona qualità, e distribuirla autorizzandone le modifiche. È essenziale, perché chi modifica un Software Libero (ne ha il diritto) dev'essere in grado di adattare la documentazione alle modifiche fatte.

2.8.2. I rimedi: la diffusione della cultura del Software Libero

La comunità del Software Libero deve impegnarsi a diffondere la cultura del Software Libero insieme al software stesso. La qualità tecnica del Software Libero e i vantaggi pratici che ne derivano hanno contribuito alla diffusione di Software Libero. L'interesse per tale software però è di tipo pratico, e cresce più in fretta della coscienza della filosofia su cui è basato. Questa situazione, secondo Stallman, è pericolosa perché gli utenti corrono il rischio di essere confusi e ingannati. I promotori del Software Libero non perdono occasione per dire che bisogna parlare di libertà e che non se ne parla abbastanza: gli utenti devono apprezzare il Software Libero perché è libero, e non solo perché ne hanno una qualche convenienza. Altrimenti, il giorno in cui qualche software proprietario offrirà loro una convenienza in più, abbandoneranno in un batter d'occhio la libertà che hanno conquistato.

3. L'Open Source

Nel 1998 Linux era un sistema operativo già ampiamente diffuso e il mondo aziendale si stava accorgendo dell'esistenza di alternative valide ed economiche ai software commerciali più diffusi. L'interesse per il Software Libero cresceva. Tuttavia il movimento del Software Libero non era affatto unito e alcuni suoi esponenti ritenevano che il mondo degli affari potesse non vedere di buon occhio l'idealismo che pervadeva la filosofia di Stallman. Per promuovere i vantaggi pratici del Software Libero (che vengono identificati in facilità di adattamento, affidabilità, sicurezza, conformità agli standard, indipendenza dai singoli fornitori) venne lanciata una campagna che sarebbe sfociata nella Open Source Initiative. Tra i promotori ricordo Eric Raymond e Bruce Perens.

3.1. La definizione di Open Source

Vediamo ora cosa hanno fatto i promotori dell'iniziativa Open Source. Essi non hanno scritto un manifesto idealista, né si sono rifatti a teorie filosofiche che fanno di utopia. Hanno affrontato la questione dal punto di vista dell'utente e hanno definito una serie di diritti che chi produce software dovrebbe garantire agli utenti. La carta dei diritti dell'utente di computer si chiama Open Source Definition e precisa i diritti che una licenza deve garantire per poter essere definita come Open Source. Pertanto l'Open Source è una famiglia di licenze che garantiscono certi diritti all'utente.



Logo della certificazione dell'Open Source Initiative

Tra le premesse del successo dell'Open Source Bruce Perens evidenzia la frustrazione del programmatore che vede il suo lavoro venduto dal proprietario dell'azienda per cui lavora senza che lui (il programmatore) ne riceva alcun compenso. Il programmatore, o l'azienda che opera sull'Open Source, può adattare qualsiasi programma Open Source a nuovi mercati o a nuovi clienti senza doverne pagare diritti o concessioni di licenza. Perens non nasconde una somiglianza con ideologie comuniste e si preoccupa di evidenziare la differenza: il comunismo è fallito perché basato su un'economia di prodotti tradizionale mentre l'economia dell'informazione si fonda su un tipo di prodotto, l'informazione appunto, che è diverso dai prodotti tradizionali, perché può venire duplicato con costi minimi.

In sintesi i diritti assicurati agli utenti dalle licenze Open Source sono:

- il diritto di fare copie del programma e di distribuirle
- il diritto d'accesso al codice sorgente del software, condizione necessaria per poterlo modificare
- il diritto di apportare migliorie al programma

Come si può vedere, questi sono simili alle quattro libertà garantite dal Software Libero. La libertà "mancante" nell'Open Source è quella del libero uso del programma, ma la ritroveremo nella definizione dettagliata di Open Source (al punto 6).

Perens sottolinea la parentela dell'Open Source con il Software Libero. Perens stesso ha affermato che «*la Open Source Definition include molte delle idee di Stallman, e può ben considerarsi un derivato della sua*

opera.»¹³ In effetti, nonostante le molte polemiche che ancora si fanno sentire, la Open Source Definition prese avvio quando Perens scrisse, per la distribuzione Linux Debian GNU/Linux, la Guida Debian. Essa serviva a capire cosa fosse gratis e a chiarire l'idea di Software Libero. A Perens si unì presto Eric Raymond, attivo nella divulgazione del Software Libero con articoli di taglio antropologico, quindi destinati non solo agli informatici. Raymond, tra l'altro, aveva scritto il saggio *La cattedrale e il bazaar* grazie al quale la Netscape Corporation l'aveva chiamato per una consulenza quando essi decisero di rendere libero il software di navigazione Netscape. La principale preoccupazione di Raymond era che il liberalismo di Stallman, apprezzato tra i programmatori, scoraggiasse la mentalità conservatrice negli ambienti degli affari. Tra il 1997 e il 1998 la Guida Debian si trasformò, diventando una definizione generale. Perens e Raymond diedero vita alla Open Source Initiative, che è «un'organizzazione destinata alla gestione della campagna Open Source e della sua certificazione di marchio.»¹⁴ Ma molte critiche si levarono, e si levano ancora oggi, dalla comunità del Software Libero e perfino dalla comunità Linux, che inizialmente aveva appoggiato il progetto. Stallman ritiene che il movimento Open Source manchi di enfasi sul concetto di libertà, mentre molti obiettano che il termine “open” sia abusato.

Scopriamo insieme a Bruce Perens i dettagli della Open Source Definition, di cui è il redattore.

Innanzitutto, è necessario precisare che i criteri che permettono di definire un software come Open Source sono molti. Non basta l'accesso al codice sorgente. La Open Source Definition è un'insieme di specifiche di quanto è ammesso in una licenza software affinché esso possa essere definito software Open Source. Tutte le specifiche vanno rispettate ed esse devono applicarsi a tutte le versioni derivate del programma originale.

- 1) **Ridistribuzione libera:** dev'essere possibile fare tutte le copie desiderate del software e venderle o cederle. Non deve essere obbligatorio alcun pagamento per tale diritto.
- 2) **Codice sorgente:** il codice sorgente dev'essere distribuito con l'opera iniziale e con tutte le opere derivate.
- 3) **Opere derivate:** dev'essere possibile effettuare qualsiasi tipo di modifica. L'opera derivata può essere distribuita con la stessa licenza dell'opera originale, ma solo se desiderato da chi l'ha modificata. Per le questioni non contemplate nella licenza, Perens insiste con riferimenti al diritto civile e penale.
- 4) **Integrità del codice sorgente dell'autore:** per proteggere l'autore dell'opera originale, una licenza Open Source può imporre che le modifiche vengano distribuite solo sotto forma di patch e che le opere derivate abbiano un nome diverso da quello dell'opera originale.
- 5) **Nessuna discriminazione contro persone o gruppi:** indipendentemente dalla nobiltà dell'intento (per esempio: la proibizione dell'uso del software in Stati non democratici), i fautori dell'Open Source non desiderano che le licenze siano “razziste”.
- 6) **Nessuna discriminazione di settori:** Perens dice, testualmente: «*La licenza non deve proibire ad alcuno l'uso del programma in uno specifico campo o per un determinato proposito.*» Una licenza Open Source,

¹³ **Bruce Perens** in *Open Sources, Voci dalla rivoluzione Open Source*, op. cit.

¹⁴ **Bruce Perens** in *Open Sources, Voci dalla rivoluzione Open Source*, op. cit.

secondo Perens, non deve immischiarsi in questioni politiche. Questo articolo tocca gli stessi temi della libertà 0 (quella sul libero uso) della definizione del Software Libero. La cosa è confermata dal Jargon File: [Open Source is] «*software distributed in source under licenses guaranteeing anybody rights to freely use, modify, and redistribute, the code.*»¹⁵

- 7) **Distribuzione della licenza:** la licenza dev'essere automatica, senza richiesta di alcuna firma.
- 8) **La licenza non dev'essere specifica a un prodotto:** un programma deve rimanere gratuito anche se slegato dalla distribuzione originale da cui proviene.
- 9) **La licenza non deve contaminare altro software:** non si può impedire che sullo stesso medium siano presenti software non Open Source.
- 10) **Licenze esemplari:** esempi di licenze che rientrano nella definizione Open Source sono GNU GPL, BSD, X Consortium e Artistica.

3.2. Le differenze tra Software Libero e Open Source

Le differenze tra Software Libero e Open Source sono minime, ma hanno dato il via a polemiche e discussioni che continuano ancora oggi.

Roberta Guerini, giornalista laureata in scienze politiche e specialista di commercio internazionale, spiega nell'articolo *Open Source o Software Libero?*¹⁶ come ciò che è successo nel mondo del Software Libero è piuttosto comune in politica: siamo di fronte a un unico grande nemico, rappresentato in questo caso dalle grandi multinazionali del software proprietario, se non addirittura dall'unica grande multinazionale del software. Con l'intento di contrastare questo nemico unico, gli avversari adottano metodologie diverse. Si formano delle correnti radicali (quella del Software Libero) e delle correnti moderate (l'Open Source). A questo proposito, troviamo nelle affermazioni di Stallman¹⁷ una precisazione: solitamente in politica i movimenti che si oppongono a un unico nemico sono d'accordo sui principi base e in disaccordo sugli aspetti pratici. I due movimenti che si battono contro il software proprietario sono d'accordo sulla maggior parte degli aspetti pratici, ma sono in disaccordo sui principi base. La cosa importante, come conferma Stallman, è che il confronto è civile e i due movimenti collaborano in molti progetti.

E in effetti entrambe le fazioni sono concordi nell'affermare che le differenze sono poche. Mentre il lato pratico resta lo stesso, l'ideologia che ne sta alla base è diversa.

¹⁵ **Eric S. Raymond**, *The New Hacker's Dictionary*, Boston, MIT Press, 1993. Consultato come *The Jargon File* presso <http://tuxedo.org/~esr/jargon/>

¹⁶ **Roberta Guerini**, *Open Source o Software Libero?*, <http://www.aziendenews.it/articoli/Internet.asp?Id=104>, 24 aprile 2002 (consultato il 12 giugno 2002)

¹⁷ **Richard Stallman**, *Perché il "Software Libero" è migliore di quello "Open Source"*, 1998, <http://www.it.gnu.org/philosophy/free-software-for-freedom.it.html> (consultato il 12 giugno 2002)

Perens¹⁸ sostiene che inizialmente la sua idea su ciò che doveva essere l'Open Source era una dolce introduzione al Software Libero, e non un movimento separato, e ribadisce l'importanza di restare uniti.

Potortì (ingegnere elettronico e ricercatore presso il CNR italiano) dice che *«La voluta neutralità del movimento open source verso gli aspetti etici e politici del Software Libero è la caratteristica sostanziale che lo distingue dalla filosofia del Software Libero, che al contrario pone l'accento sulle motivazioni ideali. Parlare di Software Libero piuttosto che di open source è una questione politica piuttosto che pratica; i due movimenti concordano infatti sulle licenze considerate accettabili, ed hanno obiettivi e mezzi comuni.»*¹⁹

Anche il Jargon File²⁰ si trova d'accordo, soprattutto sull'origine dell'Open Source: *«The intent was to be able to sell the hackers' ways of doing software to industry and the mainstream.»* Per il Software Libero l'accento viene posto sull'ideologia: *«moral imperative at the heart of free software.»*

Stallman riconosce che l'uso della definizione Open Source è positivo in quanto permette di evitare l'ambiguità libero/gratuito (anche per questo in Italia si tende a tradurre "Free Software" in "Software Libero"), ma sottolinea che lo spirito del principio alla base del Software Libero viene a trovarsi in secondo piano. Pur riconoscendo l'importanza dell'appoggio della *«comunità delle aziende»*, Stallman insiste che non bisogna smettere di parlare di libertà, comunità e principi.

È ancora Roberta Guerini²¹ a evidenziare la differenza più evidente: *«mentre l'open source basa l'attenzione sull'obbligo di chi fornisce un software di fornirne anche il codice sorgente, il free software pone il codice come pre-condizione e si sofferma maggiormente sulle quattro libertà.»*

L'atteggiamento dei due movimenti nei confronti di questa diatriba è un po' differente. Ci sono articoli di entrambe le "parti" sul tema, ma sui rispettivi siti ufficiali la cosa è trattata in modo decisamente diverso. Sul sito <http://www.opensource.org/>, si dice semplicemente che l'Open Source è il programma di marketing del Software Libero, senza aggiungere altro.

Il progetto GNU propone sul suo sito <http://www.gnu.org/> (del quale ho consultato il mirror italiano tradotto <http://www.it.gnu.org/>) un lungo documento intitolato *Perché il "Software Libero" è migliore di quello "Open Source"*. L'autore è sempre Richard Stallman, il quale si rende conto dell'ambiguità del termine inglese "free", ma ritiene che sia il termine migliore perché tutti gli altri termini esaminati, compreso Open Source, presentano ancora più problemi. Stallman evidenzia che le definizioni complete di Open Source e di Software Libero non sono ambigue, e non sono nemmeno molto diverse. Però il significato istintivo delle due espressioni è molto diverso. Mentre Software Libero presenta la nota ambiguità, Open Source sembra

¹⁸ **Robert McMillan**, *Our Man in Palo Alto* (intervista a Bruce Perens)– in *Linux Magazine* – settembre 2001 (volume 3, issue 9), pag. 35

¹⁹ **Francesco Potortì**, *Cos'è il Software Libero*, <http://www.softwarelibero.it/GNU/softwarelibero.shtml> (consultato il 12 giugno 2002)

²⁰ **Eric S. Raymond**, *The New Hacker's Dictionary*, op. cit.

²¹ **Roberta Guerini**, *Open Source o Software Libero?*, op. cit.

voler dire “Puoi guardare il codice sorgente”. Questa definizione include quella di Software Libero, ma ingloba anche software semi-liberi o addirittura proprietari. Il più grande vantaggio dell’espressione Open Source è che non mette a disagio: parlare di libertà, di problemi etici e di responsabilità può non essere gradito. Per questo Open Source ha più successo nel mondo aziendale.

La coerenza di Stallman è tale che non ha mai cercato di registrare il marchio Software Libero, al contrario di Raymond, che ha registrato Open Source. Nel 1999 però la registrazione non è stata accettata e quindi la valenza legale è ugualmente nulla per le due espressioni: non esiste alcuna restrizione legale per il loro utilizzo. L’unico trademark che la Open Source Initiative ha ottenuto è quello sul logo. Stallman fa notare che il marchio Open Source è soggetto ad abusi molto più del marchio Software Libero: le aziende sono restie a usare il termine Software Libero e quindi è molto più frequente trovare aziende che tentano di spacciare prodotti proprietari per software Open Source che per Software Libero.

4. Oggi e domani

4.1. L'Open Source nel business

Molte aziende si stanno lanciando nell'Open Source. IBM lo fa da qualche anno, mentre Macromedia ha annunciato recentemente di aver sottoposto alla Open Source Initiative la propria licenza Open, basata sulla IBM Open License. Queste aziende investono molti soldi nello sviluppo di software Open Source e molti si stanno chiedendo che vantaggi ne potranno trarre, visto che questo tipo di software tipicamente viene distribuito gratuitamente.

Il modello Open Source è un modello di business sostenibile. Questo è stato affermato dal guru Alan Cox²², secondo il quale il modello non sostenibile è quello del software proprietario. Cox, sviluppatore del kernel Linux secondo solo a Linus Torvalds, afferma che la maggior parte dei grandi salti dell'epoca informatica sono avvenuti nonostante e non grazie ai diritti di proprietà intellettuale. L'esempio più lampante è Internet, che non è il prodotto di una grande azienda ma un esempio di cooperazione possibile grazie all'innovazione libera e aperta e non all'esistenza della proprietà intellettuale.

Tuttavia è necessario restare con i piedi per terra: Adam Goodman, editor del Linux Magazine, dice che un progetto Open Source ha successo solo se ha alle spalle una comunità di sviluppatori. Un software sviluppato in un'azienda, per quanto Open Source, risponde ai bisogni dell'azienda stessa e dei suoi clienti, ma non a quelli di una comunità: «*Open Source is not just about licenses. It's about community.*»²³ Per questo le aziende che vogliono entrare nel business dell'Open Source devono preoccuparsi di sostenere la comunità Open Source.

Secondo l'attuale presidente della Apache Software Foundation Brian Behlendorf²⁴, trasformare un progetto chiuso in Open Source può inizialmente ridurre i proventi dalla vendita del software, ma, considerando che distribuire il software gratuitamente o a basso costo farà aumentare la base di utenti, aumenteranno le entrate dovute a consulenze e assistenza, a corsi e manuali.

La bontà dell'Open Source come modello di business è ribadita anche da Joel Spolsky²⁵ (informatico e fondatore di una piccola società di consulenza), il quale parte dal presupposto che la maggior parte delle aziende che stanno spendendo molti soldi per sviluppare software Open Source lo fanno perché è un buon modello di business e non perché hanno improvvisamente smesso di credere nel capitalismo.

²² **Alan Cox**, *Così funziona il software open source*, 4 maggio 2001, <http://www.milug.org/articoli/articolo.php=10> (consultato il 12 giugno 2002)

²³ **Adam M. Goodman**, *Corporate Open Source*, in *Linux Magazine*, giugno 2001 (volume 3, issue 6), pag. 6

²⁴ **Brian Behlendorf** in *Open Sources, Voci dalla rivoluzione Open Source*, op. cit.

²⁵ **Joel Spolsky**, *Strategy Letter V*, 12 giugno 2002, <http://www.joelonsoftware.com/article/StrategyLetterV.html> (consultato il 18 giugno 2002)

Le teoria economica alla base di queste affermazioni è quella dei prodotti complementari: prodotti che vengono abitualmente comprati insieme. Automobili e benzina sono prodotti complementari e l'hardware è un prodotto complementare dei sistemi operativi. La cosa importante è che, a parità di condizioni, la domanda per un prodotto cresce quando il prezzo dei suoi prodotti complementari diminuisce.

Vi sono altre premesse; bisogna considerare il Total Cost of Ownership (TCO): il costo reale di un software non è pari al prezzo di acquisto della licenza, ma include fattori intangibili (installazione, formazione,...). In secondo luogo, il software Open Source è soggetto alle leggi dell'economia tanto quanto il software proprietario. Il prezzo che l'utente deve pagare per avere driver liberi aggiornati, per esempio, è il minore sforzo di sviluppo in altri progetti.

Un'azienda può avere molto successo se riesce a tenere bassi i prezzi dei prodotti complementari dei suoi prodotti. Il prezzo più basso possibile è detto "commodity price", ed è il prezzo che si raggiunge quando l'azienda ha un mucchio di concorrenti che offrono beni indistinguibili. *«Le aziende intelligenti cercano di rendere commodities i complementi dei loro prodotti.»*

In particolare Spolsky tenta di smentire alcuni miti formati intorno ad aziende tradizionali che hanno iniziato a supportare realmente l'Open Source.

IBM sta diventando un'azienda di consulenza IT. La consulenza IT è un prodotto complementare del software d'impresa. Rendendo commodity il software d'impresa, IBM aumenterà il suo business.

Netscape ha reso Open Source il proprio browser perché ha voluto, sin dal primo giorno, rendere i browser una commodity in modo che loro potessero vendere il server. Questa strategia non è stata modificata in seguito all'acquisto di Netscape da parte di AOL/Time Warner, che è un'azienda di intrattenimento. Le aziende di intrattenimento producono complementi alle piattaforme di intrattenimento di qualunque tipo, browser inclusi.

Transmeta paga Linus Torvalds per sviluppare Linux, perché Linux è un sistema operativo e Transmeta un'azienda che produce CPU. I sistemi operativi sono complementari alle CPU e Transmeta vuole che i sistemi operativi siano commodities. Lo stesso stanno facendo Sun e HP, finanziando lo sviluppo di Ximian e Gnome.

4.2. L'Open Source nella scuola

Nell'ambito scolastico la questione può assumere caratteri molto idealistici. Per questo si sente più spesso parlare di Software Libero che di Open Source, in relazione alla scuola.

In molti ritengono che l'uso del Software Libero nelle scuole di tutti i livelli abbia solo vantaggi. Accanto ai sostenitori del Software Libero vi sono anche insegnanti e genitori. Questo entusiasmo deriva dal fatto che tutti si rendono conto che i lavoratori di domani vorranno usare quei software che hanno imparato a usare sui banchi di scuola. Le ragioni concrete per sostenere la diffusione del Software Libero nelle scuole sono da ricercare anche nel fatto che i genitori dovranno acquistare il software proprietario che i propri figli usano a scuola. Invece il Software Libero può essere portato a casa.

L'insegnante francese Jean Peyratout²⁶ ha scritto un articolo dove spiega le ragioni del suo sostegno al Software Libero facendo leva sul patriottismo francese:

- **Liberté**: la libertà di usare, copiare, modificare, distribuire, avere accesso al codice sorgente.
- **Égalité**: uguaglianza tra gli utenti, qualsiasi mezzo abbia la scuola o la famiglia, una piccola scuola senza fondi ha le stesse scelte di una più influente e prestigiosa.
- **Fraternité**: fratellanza, cooperazione e mutuo aiuto tra sviluppatori e utenti, tra utenti, tra scuole e famiglia.

Nello stesso articolo si insiste anche sulla necessità di dare agli studenti reali capacità di programmazione, e non solo «"istruzioni per l'uso" di software proprietario.» Ma non si tratta solo di capacità di programmazione: anche il semplice uso di software, e non solo il suo sviluppo, dovrebbe essere slegato da un'azienda.

La scuola dev'essere formativa e non uno strumento di addestramento o propaganda. Per aiutare a comprendere l'importanza della libertà nel software, viene spesso citato l'esempio del teorema di Pitagora: i docenti non devono chiedere una licenza per spiegare il teorema di Pitagora e gli studenti non devono pagare ogni volta che lo usano.

A livello europeo, l'uso del Software Libero nella scuola è promosso da OFSET (Organization for Free Software in Education and Teaching, <http://www.ofset.org/>), che si pone come obiettivo lo sviluppo del Software Libero per il sistema educativo e dell'insegnamento. OFSET è un'organizzazione associata alla Free Software Foundation Europe. In Svizzera l'associazione [ch/open/](http://www.ch/open/) si sta battendo per l'uso di Software Libero nelle scuole del comune di Zurigo.

4.3. L'Open Source nella Pubblica Amministrazione

Negli ultimi mesi alcuni governi hanno iniziato a interessarsi all'Open Source. Come per la scuola, il discorso sull'Open Source nella pubblica amministrazione diventa idealistico. Vediamo cosa sta succedendo nei singoli stati²⁷.

- **Finlandia**: è stato effettuato un test su Open Office. Viene raccomandato soprattutto a quegli impiegati che non necessitano di scambiare file, perché in tal caso potrebbero riscontrare problemi di compatibilità. Il 13 % dei server governativi girano su Linux, ma non ci sono policy precise.
- **Perù**: è stata proposta una serie di leggi per imporre l'uso di software Open Source, a eccezione dei casi in cui non esista una versione Open Source del programma richiesto. Le ragioni sono principalmente economiche, ma gli oppositori sostengono che scegliere il tipo di software da usare non rientra nei compiti del governo.

²⁶ Jean Peyratout, *Perché dare precedenza al software libero nelle scuole?*,

<http://ism.abul.org/program/topic14/topic14.php3?langnew=fr>

(consultato in italiano presso <http://www.fsfeurope.org/> il 24 giugno 2002)

²⁷ Joel Spolsky, *Strategy Letter V*, op. cit.

- **Corea:** il governo coreano installerà Linux su 120.000 desktop, grazie ad un accordo con un fornitore nazionale.
- **Tailandia:** un gruppo di sviluppatori finanziato dal governo ha realizzato una propria distribuzione Linux e l'ha messa a disposizione del governo e delle scuole. Si vuole contrastare l'uso di software pirata.
- **Filippine:** si imita la Tailandia, ma con un po' di ritardo. Non ci sono comunque imposizioni governative sul tipo di software da usare.
- **Francia:** alcuni server governativi sono stati spostati su Linux, con grande soddisfazione.
- **Germania:** il governo intende sostituire tutto il software proprietario con software Open Source. A tal scopo, ha preso accordi con IBM e SuSE.
- **Taiwan:** per evitare la dominanza del mercato di un solo produttore, Taiwan sta studiando la possibilità di affidarsi all'Open Source.
- **Cina:** per non dipendere da un solo produttore, per di più americano, la Cina sta investendo molte energie nello sviluppo e nell'adozione di software Open Source.
- **Italia:** recentemente è stato pubblicato dal Ministero all'Innovazione il documento *Linee guida del governo per lo sviluppo della società dell'informazione*, dove viene richiesto uno studio approfondito, al pari di quanto sta facendo l'Unione Europea, sulla strategia Open Source nella Pubblica Amministrazione. Anche il governo nazionale, dopo alcune iniziative locali, riconosce il valore del software Open Source e ne apprezza i vantaggi. Il governo riconosce che «*si diffonderanno gli standard aperti e i software open source, cioè i software liberi, la cui proprietà non sia di un singolo fornitore ma governati da una licenza d'uso che ne garantisce la possibilità di libero utilizzo, scambio, studio e modificabilità.*»²⁸ Il dibattito si sta animando proprio in questi giorni, con l'invio al Senato italiano di "memorie" e lettere chiarificatorie sull'uso dell'Open Source e sul suo significato.
- **Svizzera:** non sembra che ci siano iniziative in favore della diffusione dell'Open Source nella pubblica amministrazione svizzera. Se vi è qualcosa, non si trova presso il sito dell'amministrazione federale (<http://www.admin.ch/>). Per iniziative non governative si veda il capitolo seguente (4.4).

4.4. In Svizzera: da ch/open/ a Guglielmo Tux

In Svizzera il sostegno ai sistemi aperti inizia nel 1982, con la fondazione dell'associazione ch/open/ (Swiss Open Systems User Group). L'interesse inizialmente verteva attorno a Unix, ma al giorno d'oggi si è spostato su GNU/Linux, versione libera di Unix. Lo scopo dell'associazione è principalmente la formazione continua dei soci.

Con l'avvento di GNU/Linux si sono formati molti LUG (Linux User Group), il più grande dei quali è il LUGS (Linux User Group Switzerland). Dai LUG di Friburgo e Berna ha avuto origine recentemente

²⁸ **Punto Informatico**, *Revolution Open Source nella PA italiana*, 25 giugno 2002, <http://punto-informatico.it/p.asp?i=40678> (consultato il 25 giugno 2002)

l'iniziativa Wilhelm Tux. Il sito è stato tradotto in italiano grazie al contributo della mailing list ti-linux (<http://www.linux-ticino.ch/>) e dispone di un sito in italiano quasi completo: <http://www.guglielmotux.ch/>.

GuglielmoTux è un'iniziativa volta a promuovere l'uso del Software Libero nelle pubbliche istituzioni. Oltre alla richiesta di utilizzare solo Software Libero in tutte le pubbliche istituzioni svizzere, GuglielmoTux domanda che vengano usati, per la conservazione dei dati, esclusivamente formati non soggetti a licenze o patenti. La stessa cosa viene chiesta per i protocolli di comunicazione. Nel *Manifesto per l'utilizzo del Software Libero in Svizzera* troviamo le ragioni che l'organizzazione ritiene più importanti: stabilità e trasparenza, sicurezza, libera concorrenza e politica di regolamentazione, economicità. È prematuro fare un bilancio di questa iniziativa perché è nata solo da pochi mesi.

4.5. L'attacco all'Open Source: la sicurezza

L'Open Source e il Software Libero sono ormai trattati anche da giornali non specializzati. Purtroppo non sempre i giornalisti sono precisi e informati e spesso troviamo definizioni poco corrette e fuorvianti. A ciò si aggiungono aziende che non hanno scrupoli e sfruttano i media per far passare dei messaggi palesemente scorretti. Nella cronaca recente ricordo uno studio dell'Alexis de Tocqueville Institution (un istituto che effettua studi sulla diffusione della democrazia) che lancia un allarme perché, afferma, il software Open Source sarebbe pericoloso a causa della non segretezza del codice sorgente che permetterebbe ai terroristi di Al Qaida di sfruttarne le falle. In realtà si scopre che l'istituto in questione è ampiamente finanziato da aziende produttrici di software proprietario, come indicato in un articolo²⁹ di Michelle Delio, di Wired.com. In Italia, la replica a questo allarme arriva dal divulgatore informatico Paolo Attivissimo, su Apogeeonline³⁰. Il giornalista smonta l'accusa dimostrando come il software Open Source e il Software Libero sono di principio più sicuri del software proprietario. Infatti, l'idea alla base del software proprietario è che se ci sono errori, è meglio non dirlo in giro. Il concetto di sicurezza attraverso la segretezza, usato con successo in altri ambiti, nel campo del software funzionerebbe solo con programmi robustissimi. Ma è comunque un concetto pericoloso: se la segretezza viene a mancare anche la sicurezza ne risulta fortemente compromessa. E la segretezza può venire a mancare più facilmente di quanto si pensi, infatti quando si dice che il codice sorgente del software proprietario è segreto, bisogna comunque ricordare che sono molte le persone che vi hanno accesso e che qualcuno può sempre divulgarlo perché è stato corrotto o per vendetta. Inoltre, nel caso in cui vengano scoperti degli errori o delle falle di sicurezza, è solo il produttore che può legalmente modificare il software. Se il produttore non intende effettuare le modifiche, l'utente deve tenersi la falla oppure cambiare produttore. Nel caso in cui l'utente abbia a disposizione il codice sorgente, può riparare gli errori da solo, o pagare qualcuno perché lo faccia, se non ne è capace.

²⁹ **Michelle Delio**, *Did MS Pay for Open-Source Scare?*, 5 giugno 2002,

<http://www.wired.com/news/business/0,1367,52973,00.html> (consultato il 18 giugno 2002)

³⁰ **Paolo Attivissimo**, *L'open source aiuta il terrorismo, parola di Microsoft*, 18 giugno 2002,

<http://www.apogeeonline.com/webzine/2002/06/18/01/200206180101> (consultato il 18 giugno 2002)

Inoltre gli attacchi informatici hanno successo soprattutto con quei siti che usano software a sorgente segreta, che sono molti meno di quelli che usano software a sorgente aperta. Paolo Attivissimo fa giustamente notare che *«l'idea che la trasparenza del codice conduce a una maggiore sicurezza non è intuitiva, per cui gli oppositori dell'open source hanno gioco facile nel suscitare terrori ingiustificati.»*

Altri critici tentano di scoraggiare l'uso dell'Open Source con note tecniche di propaganda massmediale, tra cui la famosa tecnica FUD. FUD sta per paura (Fear), incertezza (Uncertainty) e dubbio (Doubt) ed è una vera e propria tecnica di marketing usata quando il proprio prodotto non è più competitivo. Non avendo la capacità per rispondere con i fatti, vengono diffuse voci allarmistiche attraverso canali informali. Questa forma di disinformazione venne formalizzata per la prima volta quando IBM la utilizzò, negli anni Settanta. I venditori di IBM diffondevano continuamente voci sull'inaffidabilità dei prodotti hardware dei concorrenti e sul rischio che i clienti di IBM avrebbero corso in caso di migrazione. La FUD fu talmente efficace che contribuì al successo di IBM. Il termine FUD è presente anche nel Jargon File.

4.6. L'Open Source al di fuori del software

Se il successo della diffusione della filosofia della libertà non è totale nel mondo del software, possiamo citare alcune iniziative che ne applicano i principi a campi diversi dal software. Troviamo interessante che spesso la componente ideologica avvicini queste iniziative alla Free Software Foundation, mentre esse si fregiano del nome Open.

Il campo più interessante a cui è stata applicata l'ideologia dell'Open è la diffusione dei contenuti, siano essi tesi, immagini o suoni. L'iniziativa Open Content (<http://www.opencontent.org/>) è una sorta di ritorno alle origini: la filosofia della libertà d'espressione, applicata al software per dar vita al Software Libero, torna arricchita alle idee e alla libertà d'espressione. Arricchita perché esplicita maggiormente i concetti che hanno spinto a sostenere la libertà d'espressione: il concetto che se le idee circolano migliorano e quello che dà a tutti la possibilità di collaborare.

Il progetto Open Content è guidato dal ricercatore dell'università dello Utah David Wiley che, con la consulenza di Richard Stallman, ha adattato la GNU GPL ai contenuti. La licenza Open Content definisce i termini e le condizioni di copia, distribuzione e modifica, ma lascia in sospeso uso e attendibilità. La vendita, come nel caso del Software Libero, non è vietata, ma si garantisce che i diritti per l'utente sull'opera non diminuiscano con il passaggio di mano in mano.

Il riconoscimento dell'autore è garantito dalla licenza Open Publication, che permette di diffondere l'opera con ogni medium e mantiene il copyright all'autore, senza però ostacolare la partecipazione e l'arricchimento dei contenuti da parte di altri. La licenza Open Publication tutela l'autore garantendogli il diritto di impedire una versione commerciale dell'opera.

Questo metodo di diffusione della conoscenza funziona soprattutto grazie a Internet e contribuisce a ridurre la perdita di attendibilità. L'iniziativa è troppo giovane per capire se creerà un mercato per scrittori, grafici o artisti indipendenti, come è successo con l'Open Source e i programmatori. D'altra parte, sulla Rete lo spazio non manca ed è facile prevedere che un futuro ci sarà.

Tra le applicazioni più importanti dell'Open Content, oggi, vi sono l'enciclopedia pubblica Nupedia (<http://www.nupedia.org/>, che può vantare tra gli ideatori Richard Stallman) e l'Open Directory Project dmoz (<http://dmoz.org/>).

4.7. Le aziende

Ho interpellato il personale tecnico di alcune aziende e ho posto alcune domande sull'Open Source. Non si tratta di un sondaggio con pretese di rappresentatività.

L'impressione è che le aziende tecnologiche che hanno sempre lavorato su Unix non incontrino grossi problemi ad utilizzare software Open Source. I motivi sono due: il grosso del software Open Source è fatto per Linux, quindi spostarsi da uno Unix commerciale a Linux non pone problemi. Inoltre il personale tecnico è esperto e non ha problemi a imparare a usare software nuovi. La migrazione quindi è solitamente poco costosa e poco problematica. Si riscontrano problemi solitamente quando si tratta di migrare banche dati, ma questo accade anche con software proprietario. Tra i software utilizzati troviamo Linux, gcc, emacs, comandi di shell bash, Apache e altri, soprattutto server side. Le distribuzioni Linux commerciali offrono una vasta scelta di software. I vantaggi più apprezzati sono la flessibilità, la sicurezza, la frequenza degli aggiornamenti, la possibilità di fare modifiche e di controllare il software, il prezzo, la semplicità di integrazione con sistemi Unix già esistenti, la possibilità di effettuare test senza la preoccupazione di violare qualche licenza e, forse la motivazione più importante, il fatto che molti programmi Open Source non solo sono standard, ma hanno anche contribuito a determinare il successo di uno standard, come Apache. Le aziende tecnologiche si ritengono soddisfatte dai software Open Source che usano.

Ho parlato anche con il responsabile informatico di un'azienda il cui business non è di tipo tecnologico. L'atteggiamento è diverso: il personale addetto all'informatica è ridotto e gli utenti, generalmente, non hanno l'interesse o le capacità per cambiare tipo di software. L'installazione di software Open Source viene fatta se richiesto, in quanto non comporta problemi amministrativi legati a costi e licenze, ma l'assistenza è limitata. A volte però non è semplice integrare i programmi Open Source con i sistemi operativi commerciali. I servizi principali (desktop utenti, applicazioni, fileserver, webserver, printserver e mailserver) restano su piattaforme proprietarie perché sono semplici da installare e configurare e si trova facilmente assistenza.

5. Conclusione

Penso di avere approfondito alcuni argomenti che vengono trattati superficialmente anche da parte degli addetti ai lavori, degli appassionati e perfino dei più fanatici. La differenza tra Software Libero e Open Source è ampiamente dibattuta, ma non è conosciuta al di fuori dell'ambiente che ruota intorno alle organizzazioni coinvolte. D'altra parte le implicazioni della filosofia dell'Open sono molte e non si sono ancora completamente sviluppate. Le idee di Stallman stanno trovando in questi tempi un terreno fertile grazie a una certa intolleranza che si sta diffondendo tra i consumatori nei confronti di pratiche aziendali spregiudicate. Il movimento No Global, secondo me, ne è un'espressione.

Ma tra i consumatori di software vi sono anche le aziende, che a loro volta desiderano poter operare in un mercato più sano e più concorrenziale. Tutte queste persone si stanno rendendo conto che l'unico modo per impedire a un'unica ditta di imporre i suoi standard chiusi e appropriarsi del mercato non è ricorrere a un prodotto concorrente, perché si farebbe la stessa fine, ma è ricorrere a una filosofia alternativa, che pregiudichi la possibilità per un singolo di governare il mercato. L'Open Source è una filosofia la cui implementazione software principale è l'interoperabilità, cioè la possibilità di interagire con sistemi di altri produttori. Questo è anche il grande concetto che deve garantire. Ricordo che il web è libero anche grazie al Software Libero Apache, i cui sviluppatori, forti della notevole diffusione del loro web server, non hanno ceduto alle pressioni di produttori di browser commerciali non rispettosi degli standard. L'applicazione dell'interoperabilità in ambito biogenetico e bioinformatico è fondamentale per la libertà nella ricerca e nella medicina.

Il software Open Source, con la sua buona dose di idealismo, è una reale alternativa di qualità.

6. Glossario

6.1. Hacker

Secondo Richard Stallman «*uno che ami programmare, e a cui piaccia essere bravo a farlo.*»³¹ Anche nel Jargon File³²: «*A person who enjoys exploring the details of programmable systems and how to stretch their capabilities.*» Ma anche «*A person capable of appreciating hack value.*» Il termine può comunque essere più generico: «*An expert or enthusiast of any kind*» oppure «*One who enjoys the intellectual challenge of creatively overcoming or circumventing limitations.*» Nel senso di pirata informatico, come spesso si trova sulla stampa generica (ma anche specialistica) si preferisce *cracker*.

6.2. Il termine *free*

In inglese si dice *free software*. Il termine *free* può voler dire sia gratuito che libero. È Stallman stesso, in uno dei molti interventi sul Software Libero³³ a chiarire l'equivoco dicendo: «*Il "Software Libero" è una questione di libertà, non di prezzo. Per capire il concetto, bisognerebbe pensare alla "libertà di parola" e non alla "birra gratis".*» Nonostante i numerosi articoli e interventi in questo senso, il termine *free* continua a suscitare confusione.

Ho usato il termine *Software Libero*, invece dell'originale inglese *Free Software*, perché ho riscontrato che in Italia la traduzione è entrata nell'uso comune. Per l'espressione *Open Source* ho mantenuto l'originale inglese in quanto la traduzione *Sorgente Aperta* è inutilizzata.

6.3. La licenza d'uso

La licenza d'uso è un documento legale che definisce l'insieme di norme a cui è soggetto l'uso di un programma software. Infatti i programmi software sono opere di ingegno e l'acquirente non paga la proprietà, ma la possibilità di usarli a determinate condizioni. Ogni programma, o quasi, viene distribuito con una licenza d'uso. Le licenze d'uso proprietarie si basano sul diritto d'autore (copyright) e impongono restrizioni. La licenza d'uso GNU GPL, invece, concede le quattro libertà fondamentali all'utente del programma e le protegge (copyleft: permesso d'autore).

6.4. Codice sorgente e codice eseguibile

Un programma viene scritto in codice sorgente, un linguaggio comprensibile all'uomo. Esempi di questi linguaggi sono: C, C++, java, Algol, Pascal, Oberon, Ada, Basic,... Poi viene tradotto grazie a un programma compilatore in codice eseguibile (detto anche codice binario), cioè nella forma comprensibile

³¹ **Richard M. Stallman** in *Open Sources, Voci dalla rivoluzione Open Source*, op. cit.

³² **Eric S. Raymond**, *The New Hacker's Dictionary*, op. cit.

³³ **Richard Stallman**, *Cos'è il Software Libero?*, 1996, <http://www.it.gnu.org/philosophy/free-sw.it.html> (consultato il 13 giugno 2002)

alla macchina. Per usare il programma è sufficiente la versione binaria. Per studiare o modificare un programma è necessaria la versione sorgente, dalla quale è comunque possibile ricavare l'eseguibile. Tradurre all'indietro dall'eseguibile alla sorgente è (quasi) impossibile.

Il Software Libero viene obbligatoriamente distribuito completo di codice sorgente. Invece solo la versione eseguibile viene distribuita quando si tratta di software proprietario. Esistono alcune eccezioni: Microsoft ha distribuito la versione sorgente di alcune parti di Windows ad alcuni suoi clienti per permettere di studiarlo e quindi produrre hardware o software che si integri meglio. Non si tratta di Software Libero: le restrizioni sull'uso e sulla divulgabilità di quei sorgenti sono molto severe.

6.5. L'etica hacker

Sopravvivere. Socializzare. Divertirsi. (Linus Torvalds)

Cosa ha permesso al software libero di nascere e di svilupparsi? Pekka Himanen, professore di filosofia all'università di Helsinki, ha messo in evidenza quello che è lo spirito dell'età dell'informazione: l'etica hacker. Il suo recente saggio si intitola, appunto, *L'etica hacker e lo spirito dell'età dell'informazione*³⁴, con un evidente riferimento a *L'etica protestante e lo spirito del capitalismo* di Max Weber. Senza voler affrontare in profondità discussioni filosofiche o sociologiche, vorrei comunque riassumere in breve qual è l'etica hacker. In tal proposito ci basiamo anche su ciò che Linus Torvalds ha detto in *Rivoluzionario per caso*³⁵: «*Ci sono tre cose importanti nella vita. [...] Il primo è la sopravvivenza, il secondo l'ordine sociale e il terzo il divertimento. Tutto, nella vita, procede in quest'ordine. E dopo il divertimento non c'è nient'altro. Quindi da un certo punto di vista questo vuol dire che il senso della vita è raggiungere la terza fase.*»

Quindi ciò che conta per l'hacker è il divertimento, inteso come soddisfazione e realizzazione personale. In questo senso lavorare diventa un piacere e un modo per socializzare. Il senso della comunità è molto forte.

«*Linux è il più grande progetto collaborativo del mondo - finalizzato a costruire la migliore e più bella tecnologia disponibile a chiunque la desideri. È molto semplice. E divertente.*»³⁶

Raymond afferma in *Colonizzare la noosfera*³⁷ che la comunità hacker è basata sulla cultura del dono. Questo tipo di cultura si sviluppa in società dove prevale l'abbondanza e non vi è penuria di merci necessarie alla sopravvivenza. Invece dove prevale la scarsità, la cultura dominante è la cultura dello scambio. Nelle culture del dono, lo status sociale non viene determinato da ciò che si controlla, ma da ciò che si regala. Ciò è tipico in alcune culture di tipo aborigeno dove il cibo è abbondante oppure nell'élite di ricchissimi del mondo dello spettacolo. Nel caso della cultura hacker, non esiste penuria di materiale di sopravvivenza (spazio su disco, ampiezza di banda di rete, macchine potenti) e il software è liberamente condiviso. L'unico

³⁴ **Pekka Himanen**, *L'etica hacker e lo spirito dell'età dell'informazione*, Milano, Feltrinelli, 2001

³⁵ **Linus Torvalds, David Diamond**, *Rivoluzionario per caso. Come ho creato Linux (solo per divertirmi)*, op. cit.

³⁶ **Linus Torvalds, David Diamond**, *Rivoluzionario per caso. Come ho creato Linux (solo per divertirmi)*, op. cit.

³⁷ **Eric S. Raymond**, *Colonizzare la noosfera*, aprile 1998,

<http://www.apogeeonline.com/openpress/doc/homesteading.html> (consultato il 12 giugno 2002)

ambito disponibile per la competizione sociale è la reputazione fra colleghi, quindi lo status sociale è determinato da quanto software di qualità si scrive.

Ma perché la gente usa il software Open Source? Risponde Paolo Attivissimo: «è molto più cool.»³⁸ Più estesamente: è un indice di indipendenza ed è importante sentirsi parte di un progetto privo di secondi fini. Ovviamente, percorrere da soli certe strade è più faticoso, ma si è ripagati dalla sensazione di libertà.

Il senso della comunità e della possibilità di partecipare a qualsiasi progetto si vede anche in un esempio italiano: basta inviare le parole mancanti allo sviluppatore del controllo ortografico italiano della suite OpenOffice.org e lui le inserirà nella prossima versione. È una cosa assolutamente impossibile con un prodotto commerciale.

³⁸ **Paolo Attivissimo**, *OpenOffice.org: come fare (quasi) a meno di Office*, 25 giugno 2002, <http://www.apogeeonline.com/webzine/2002/06/25/01/200206250101> (consultato il 26 giugno 2002)

7. Bibliografia

7.1. Libri

- **Linus Torvalds, David Diamond**, *Rivoluzionario per caso. Come ho creato Linux (solo per divertirmi)*, Milano, Garzanti, 2001
- **Pekka Himanen**, *L'etica hacker e lo spirito dell'età dell'informazione*, Milano, Feltrinelli, 2001
- **Richard M. Stallman, Bruce Perens, Eric S. Raymond, Brian Behlendorf e altri**, *Open Sources, Voci dalla rivoluzione Open Source* a cura di **Chris DiBona, Sam Ockman e Mark Stone**, Milano, Apogeo, 1999; disponibile su web presso <http://www.apogeeonline.com/openpress/libri/545/> (consultato il 12 giugno 2002)
- **Richard M. Stallman**, *Manifesto GNU*, <http://www.pluto.linux.it/pluto/ManifestoGNU.html> (consultato in data 28 aprile 1999)
- **Eric S. Raymond**, *The New Hacker's Dictionary*, Boston, MIT Press, 1993. Consultato come *The Jargon File* presso <http://tuxedo.org/~esr/jargon/>

7.2. Articoli

- **Paolo Attivissimo**, *L'open source aiuta il terrorismo, parola di Microsoft*, 18 giugno 2002, <http://www.apogeeonline.com/webzine/2002/06/18/01/200206180101> (consultato il 18 giugno 2002)
- **Paolo Attivissimo**, *OpenOffice.org: come fare (quasi) a meno di Office*, 25 giugno 2002, <http://www.apogeeonline.com/webzine/2002/06/25/01/200206250101> (consultato il 26 giugno 2002)
- **Matt Berger**, *Snapshots from the OS front*, 10 giugno 2002, <http://www.infoworld.com/articles/hn/xml/02/06/12/020612hnossnapshot.xml> (consultato il 12 giugno 2002)
- **Antonio Bernardi**, *Ruolo e prospettive del sistema operativo Linux nella didattica dell'informatica e nella scuola*, 15 giugno 1998, <http://animal.unipv.it/GNU/linux-didattica.html> (consultato il 17 marzo 2001)
- **Luisa Bortolotti e Paolo Rauzi** (a cura di), *Atti del seminario "Informatica e scuola: il fenomeno Linux"*, Trento, Iprase-PAT, 2000, disponibile su internet presso <http://users.iol.it/telaio/enigma/introduz.htm> (consultato il 24 giugno 2002)
- **Alan Cox**, *Così funziona il software open source*, 4 maggio 2001, <http://www.milug.org/articoli/articolo.php=10> (consultato il 12 giugno 2002)
- **Michelle Delio**, *Did MS Pay for Open-Source Scare?*, 5 giugno 2002, <http://www.wired.com/news/business/0,1367,52973,00.html> (consultato il 18 giugno 2002)
- **Adam M. Goodman**, *Corporate Open Source* – in *Linux Magazine* – giugno 2001 (volume 3, issue 6), pag. 6

- **Roberta Guerini**, *Open Source o Software Libero?*,
<http://www.aziendenews.it/articoli/Internet.asp?Id=104>, 24 aprile 2002 (consultato il 12 giugno 2002)
- **Roger Irwin**, *Cosa significa FUD?*, 1998, http://www.pluto.linuc.it/journal/pj0204/whatisfud_it.html
(consultato il 12 giugno 2002)
- **Ransom Love**, *Linux: The Dawn of a New Day* – in *Linux Magazine* – maggio 2001 (volume 3, issue 5),
pag. 26
- **Robert McMillan**, *Our Man in Palo Alto* (intervista a Bruce Perens)– in *Linux Magazine* – settembre
2001 (volume 3, issue 9), pag. 35
- **Gianluca Miscione**, *Da Open Source a Open Content* – in *Internet News* – giugno 2002
- **Sam Ockman**, *Taking the “Free” Out of Open Source* – in *Linux Magazine* – agosto 2001 (volume 3,
issue 8), pag. 20
- **Tim O’Reilly**, *What’s Next for Linux and Open Source?* – in *Linux Magazine* – ottobre 2001 (volume 3,
issue 10), pag. 16
- **Jean Peyratout**, *Perché dare precedenza al software libero nelle scuole?*,
<http://lsm.abul.org/program/topic14/topic14.php3?langnew=fr> (consultato in italiano presso
<http://www.fsfeurope.org/> il 24 giugno 2002)
- **Francesco Potortì**, *Cos’è il Software Libero*, <http://www.softwarelibero.it/GNU/softwarelibero.shtml>
(consultato il 12 giugno 2002)
- **Punto Informatico**, *Revolution Open Source nella PA italiana*, 25 giugno 2002, <http://punto-informatico.it/p.asp?i=40678> (consultato il 25 giugno 2002)
- **Eric S. Raymond**, *Colonizzare la noosfera*, aprile 1998,
<http://www.apogeeonline.com/openpress/doc/homesteading.html> (consultato il 12 giugno 2002)
- **Eric S. Raymond**, *La cattedrale e il bazaar*, 22 novembre 1998,
<http://www.apogeeonline.com/openpress/doc/cathedral.html> (consultato l’8 maggio 2002)
- **Alessandro Rubini**, *FUD: Fear, Uncertainty and Doubt*, 2001,
<http://www.softwarelibero.it/GNU/nemici/fud.shtml> (consultato il 12 giugno 2002)
- **Joel Spolsky**, *Strategy Letter V*, 12 giugno 2002,
<http://www.joelonsoftware.com/article/StrategyLetterV.html> (consultato il 18 giugno 2002)
- **Richard Stallman**, *Perché il “Software Libero” è migliore di quello “Open Source”*, 1998,
<http://www.it.gnu.org/philosophy/free-software-for-freedom.it.html> (consultato il 12 giugno 2002)
- **Richard Stallman**, *Cos’è il Software Libero?*, 1996, <http://www.it.gnu.org/philosophy/free-sw.it.html>
(consultato il 13 giugno 2002)

7.3. Siti web

- <http://www.opensource.org/>
- <http://www.gnu.org/>
- <http://www.linux.it/GNU/>
- <http://www.linux-ticino.ch/>
- <http://www.ch-open.ch/>
- <http://www.guglielmotux.ch/>
- <http://www.softwarelibero.it/>