Software Libero - Le Ragioni di Una Scelta

7 novembre 2006

Contents

1 Cos'è il software libero

Il software è libero se l'utente finale gode di 4 libertà:

libertà 0: di eseguirlo per qualsiasi scopo; Non sempre è concesso di usare un software senza limiti, ad esempio:

- certi software possono essere eguiti solo sull'hardware col quale sono venduti (OEM);
- certi software non possono essere utilizzati per determinati scopi;

libertà 1: di studiare come funziona ed adattarlo alle proprie necessità: L'accesso al codice sorgente ne è il pre-requisito quindi:

- il codice sorgente può non essere distribuito ma deve essere almeno accessibile si richiesta;
- è possibile verificare cosa fa e come funziona un'applicazione

libertà 2: di copiarlo e redistribuirlo; il software proprietario nega evidentemente questa libertà , nel software libero invece:

- la copia è incoraggiata, non è possibile impedirla;
- non è necessario chiedere permessi all'autore e al distributore;
- libero non è gratuito: la copia può essere a pagamento per il servizio reso.

Attenzione: pagando od ottenendo il sotware non si diventa "proprietari" del software, esattamente come nel software proprietario!

Nota: La legge 248/200 imponeva che chi distribuiva contenuti digitali su supporto fisico doveva dichiararlo alla siae, acquistando un bollino da applicare sul supporto; questo andava in contrasto con il software libero perché richiedeva, per la copia del prodotto, il pagamento di una tassa che l'autore stesso non reclamava. Successivamente, con il regolamento attuativo, si è definito che il software libero non ha bisogno del fantomatico bollino siae.

libertà 3: di migliorarlo e distribuirne pubblicamente i miglioramenti: Avere il codice sorgente e poterlo modificare non basta: deve essere possibile redistribuire anche le proprie modifiche (è successo con la libreria QT e quindi con KDE):

- non possono essere posti limiti su chi può eseguire le modifiche o la loro redistribuzione;
- d'altro canto non si può obbligare nessuno a redistribuire le modifiche effettuate. Se però si distribuisce il binario si deve poter richiedere il codice sorgente (a meno di licenze particolari)

1.1 Implicazioni delle libertà

E' necessario avere l'accesso al codice sorgente;

- Avere queste libertà non significa che non possono essere introdotte altre restrizioni purché non entrino in conflitto con le libertà fondamentali. Ad esempio col copyleft si impone che le libertà fondamentali non possano essere revocate.
- Queste libertà non si applicano solo al software ma alla produzione culturale in generale: ci sono licenze specifiche pensate per testi, documenti, audio, musica...

Lo sviluppo del software libero si basa su principi quali:

- il libero scambio di informazioni;
- la condivisione di idee e risultati;
- il libero utilizzo del patrimonio culturale comune;

Sono i principi che stanno alla base della comunità scientifica.

1.2 Le origini del Software Libero

30 anni fa il software era perlopiù distribuito liberamente, non esistevano le libertà fondamentali ma semplicemente non era presa in considerazione l'idea di chiudere il software. L'utenza era decisamente più limitata, l'ambito era prettamente accademico e gli autori spingevano verso la redistribuzione dei loro programmi per farsi conoscere e farlo diffondere. In pratica il software era più o meno libero ma non sapeva di esserlo.

Il Progetto GNU è nato nel 1984 da un'idea di Richard Stallman. GNU sta per "GNU is not Unix" ricordando così che GNU proviene ed è ispirtato dal sistema operativo proprietario Unix. Una grande impulso "popolare" è stato dato con la creazione del kernel di Linux nel 1991. Con l'avvento di Linux, ovvero con l'opportunità di poter utilizzare un sistema completamente libero, c'è stata una grande diffusione.

All'inizio appunto il software era abbastanza libero, libero non nei termini che lo pensiamo oggi, ma non esisteva nemmeno un software proprietario come oggigiorno. Per scopi principalmente economici il software si è mosso sulla direttiva del software proprietario.

Il progetto GNU nasce con lo scopo di creare un sistema (formato da diverse utility) completamente libero. Il problema principale di questo progetto era che questo software girava sopra un sistema operativo non libero: mancava la parte principale del sistema, quella che si interfacciava con l'hardware. Nel '91 Torvalds annunciò la disponibilità di un kernel per PC386: era nato Linux. Mettendo insieme l'infrastruttura di Stallman con il kernel di Torvalds si disponeva quindi di una infrastruttura totalmente libera.

Negli stessi anni internet diventa sempre più pervasiva permettendo a molti utenti di contribuire e scambiarsi software. Lo sviluppo ha subito un'impennata e quindi anche la base di utenza è cresciuta.

Nel '98 Raymond e Perens volevano avvicinare le aziende al software libero ma le aziende avevano paura di perdere il controllo della base. Raymond creò quindi la definizione di Open Source per invogliare e venire incontro ai timori che il software libero risvegliava. La definizione, invece delle 4 libertà fondamentali, utilizza le linee guida di Debian (di cui Sperence era progettista): nell'Open Source si richiede la redistribuzione libera, la disponibilità del codice sorgente, la possibilità di fare opere derivate, la non discriminazione di utenti o modalità di utilizzo, ecc... Open Source voleva risolvere l'ambiguità del termine inglese "free", spesso inteso come gratis.

Sebbene nella pratica le due definizioni siano identiche c'è una differenza filosofica:

- Open Source mette l'accento sulla convenienza pratica tralasciando gli aspetti etici e filosofici;
- Software Libero invece insiste sull'aspetto filosofico, sulla libertà che salvaguardiano gli utenti al di là degli immediati aspetti di convenienza.

Giacché l'Open Source guarda la cosa da un punto di vista prettamente pratico portanto in sé alcune ambiguita:

1. si tende ad identificare l'Open Source come modello di sviluppo di un software piuttosto che come una caratteristica (licenza) da applicare al prodotto finale. Tale modello, però nella pratica, non è invece garanzia di qualità come dimostrano vari progetti presenti sulla rete. Quello che rende superiore il software libero sono i diritti associati alle licenze, non l'uso di un particolare modello di sviluppo.
2. c'è chi usa la definizione di Open Source per dire che i sorgenti sono disponibili anche se non modificabili e redistribuibili.

Recentemente lo stesso Sperence ha ammesso che l'Open Source ha portato a queste ambiguità a causa del suo approccio pragmatico.

2 Il Software Libero: gli aspetti legali

2.1 Introduzione

Il software, come qualsiasi altra opera d'ingegno è protetto dal diritto di autore. A livello normativa si basa sulla convenzione di Berna dove si descrive che all'autore sono concessi i diritti per la diffusione, copia, riutilizzo e sfruttamento commerciale dell'opera. Senza il permesso dell'autore, teoricamente, non si può fare niente dell'opera.

La **licenza** è, di fatto, il *contratto* dove l'autore o il detentore dei diritti specificano chi può fare che cosa con l'opera in questione.

Si può parlare di detentore dei diritti poiché l'autore può cedere tutti i diritti della propria opera ad altri (ma nella legislazione europea non può cederne la paternità, nei paesi anglosassoni è cedibile anche il diritto alla paternità). E' il caso, ad esempio, delle società con dipendenti nei quali l'opera dell'ingegno non appartiene appunto agli sviluppatori ma all'azienda. In teoria sarebbe possibile anche far sì che i diritti di autore siano cointestati sia alla società che ai singoli sviluppatori.

Si parla di diritto di autore e non di di *proprietà intellettuale* poiché è un termine che non esiste a livello giuridico e che descrive una gamma piuttosto ampia di elementi:

- il diritto di autore (copyright)
- i brevetti (è legato ad un'implementazione o, solo nel caso degli States, ad un'idea astratta; può essere concesso sotto pagamento)
- i marchi registrati (è legato ad un'immagine, un nome, un logo)
- il segreto industriale (simile al brevetto ma non cedibile neanche a pagamento)

che sono cose completamente diverse tra di loro e che quindi vanno affrontate in modo diverso poiché hanno problematiche diverse. Nel caso del software sono nate diverse aberrazioni: sono state brevettate le idee delle finestre con risposta sì, no; del mouse che cambia forma a seguito del passaggio da una finestra all'altra...

Il diritto di autore non neace per proteggere l'autore stesso, la convenzione di Berna pensa piuttosto a come la comunità possa beneficiare delle opere di ingegno che sono prodotte. Si cerca quindi di incentivare gli autori a produrre per il pubblico e il modo più semplice è fornire all'autore un vantaggio economico.

Sarebbe preferibile che le opere fossero di *pubblico dominio* sin da subito (ad esempio nell'antichità le opere letterarie nascevano di pubblico dominio) ma chiaramente se non si incentiva l'autore non si spinge a condividere le proprie invenzioni: il diritto d'autore è quindi un compromesso per il quale l'autore, per un certo periodo di tempo, gode di un vantaggio (il brevetto) per il quale è remunerato per l'uso della sua invenzione.

Ad oggi i benefici per le società stanno diminuendo a favore dei privilegi garantiti agli autori: il tempo di copertura del copyright si è progessivamente allungato e sono state poste sempre più limitazioni all'uso dell'opera.

2.2 Il Software Libero e il copyright

Nel caso del software ci sono alcune particolarità, essendo il software non concreto ci sono delle diversità:

- il software può essere anche solo un binario, non leggibile né modificabile; questa è una prima semplice protezione che non concede di arrivare direttamente all'informazione.
- il software stesso può essere programmato per girare solo a determinate condizioni o su determinato hardware.
- il costo della copia del software è bassissimo: non siamo nel caso di un armadio o di un'auto;
- la copia non diminuisce il valore dell'*"originale"* né toglie qualcosa a chi ne permette la copia.

Come è quindi legato il software libero al copyright? Il software libero esiste grazie al diritto di autore; il software senza copyright è software di pubblico dominio (che è una cosa ben diversa!). Nel software gli autori sono ben noti ed esso viene rilasciato attraverso una licenza ben precisa. Il software libero quindi usa il copyright, ma attraverso le licenze con cui è rilasciato, invece di togliere le libertà agli utenti gliele garantisce.

Il *copyleft* significa "permesso d'autore", è stato ideato all'interno del progetto GNU con la licenza GPL; il copyleft è una specie di "estensione" del copyright: indica una restrizione che impone che le libertà concesse all'utente non possano essere revocate in futuro, ma che restino tali: il copyleft, applicato al caso particolare del software libero è una garanzia che tale software restarà libero; è a garanzia dell'autore ma sopratutto dell'utente. L'utente sa che nessuno potrà togliergli questi diritti. Quello che può succedere, ad esempio, è che il/i detentore/i dei diritti, in una nuova versione del suo prodotto rilasci l'opera con un'altra licenza con restrizioni diverse.

Attenzione: il copyleft non è un'estensione delle quattro libertà; ad esempio può implicare la distribuzione ma non la copia. E' il caso di un'idea filosofica: sono interessato a che si diffonda ma non voglio che venga modificata (specie se a mio nome!).

Il copyright è usato per limitare le libertà dell'utente; il copyleft invece garantisce i diritti dell'utente, riservandosi quei diritti necessari a mantenere questa garanzia. Il copyleft vuole evitare che il software libero si chiuda come successe negli anni '80, ma ben inteso: senza copyright non potrebbe esistere il copyleft perché non esisterebbero le garanzie necessarie per evitare comportamente predatori.

Note: Free Software Foundation ha i diritti d'autore completi delle loro opere: chi modifica parti del codice deve cedere tutti i suoi diritti (meno la paternità della modifica); per contro i detentori dei diritti del kernel linux sono **tutti** coloro che hanno partecipato al suo sviluppo. Sono due estremi opposti: nel primo caso fsf prenderà lei tutte le future decisioni che sono ad appannaggio dell'autore; nel secondo caso invece tutti gli sviluppatori dovranno trovarsi d'accordo per approfare una nuova decisione.

2.3 Una classificazione delle licenze

- **software libero con copyleft**: restringe la redistribuzione delle modifiche ad essere rilasciate ancora una volta come software libero; nella pratica così evitiamo che un software libero possa essere chiuso diventando proprietario.
- **software libero senza copyleft**: non impedisce che le modifiche possano essere rilasciate come software non libero.
- **software di pubblico dominio**: è un software senza copyright, disponibile per chiunque.
- **freeware**: vuol dire software gratuito, il ché non comporta niente sotto il profilo dei diritti che si hanno su di esso (ad esempio skype).
- **software semi-libero**: dove non tutte le libertà fondamentali sono concesse.
- **software commerciale**: commerciale non significa proprietario, significa solo fatto per trarne profitto.
- **software proprietario**: uso, modifica e distribuzione sono proibilte o fortmente limitate, ma può essere gratuito.
- **shareware**: ne è permessa la distribuzione ma sotto certi limiti, normalmente dopo un certo periodo bisogna pagarne per l'uso.

2.4 Le principali licenze del Software Libero

2.4.1 Software libero con copyleft

- **GPL**: General Public Licence
- **LGPL**: Lesser General Public Licence (talvolta chiamata non correttamente Library perché usata spesso nella pubblicazione delle librerie)

Sono licenze che garantiscono le 4 libertà fondamentali ed adoperano il copyleft. La licenza GPL è la più diffusa e la prima ad utilizzare il concetto di copyleft nel suo testo; richiede che i lavori che usano codice GPL vengano redistribuiti con la stessa licenza. Tutela sia gli autori, a cui resta il copyright, che gli utenti a cui vengono garantite le libertà fondamentali senza possibilità di revoca. La LGPL permette di utilizzare questo software collegato a software non libero, è stata studiata per avvicinare al mondo del free-software i produttori di software non libero; ogni modifica a del codice LGPL deve comunque essere rilasciata sotto la stessa licenza.

2.4.2 Software libero senza copyleft

- **BSD**: concede le 4 libertà ma consente di cambiare, nel tempo, i termini della licenza; chiede che la copia riporti sempre la clausola di copyright e il contenuto della licenza; limita inoltre l'uso del nome dell'autore nella promozione di prodotti derivati; infine non impone restrizioni sull'uso del codice in altri programmi che possono essere proprietari.
- **X11** o **Xfree**: sono analoghe a BSD, con alcune differenze riguardo l'uso del nome dell'autore.
- **Apache**: anche la Apache Software Foundation è partita da software libero e poi si è sviluppata la propria licenza. Ovviamente garantisce le 4 libertà fondamentali (altrimenti non sarebbe software libero!) e poi è simile alla X11 ma impone un ringraziamento agli autori di Apache.

Ma di licenze ne esistono un mare... sul sito del progetto GNU tutte le varie licenze sono classificate e commentate in base al grado di libertà che offrono, dichiarando se sono compatibili con la GPL. Ovviamente sul progetto GNU è espresso il loro pensiero, diverso ad esempio da quello della Open Source Initiave.

3 Perché il software libero?

3.1 Quali vantaggi?

Il software libero si configura come un bene pubblico poiché il benificio è ad appannaggio di chiunque voglia utilizzarlo. Gli attori economici inoltre operano in condizioni di sostanziale parità. I vantaggi sono di diverso tipo:

- **strategico**: nel caso del software libero chi usa questo software è formalmente indipendente dal fornitore poiché l'utilizzatore ha il pieno controllo delle tecnologie adoperate; l'utilizzatore quindi ha pieno accesso alla tecnologia utilizzata.
- **tecnico**: anche se il software libero non è garanzia di qualità si può disporre di alcuni vantaggi preclusi ai software proprietari: innanzitutto si ha la possibilità tecnica di verificare la correttezza dell'applicazione e cosa effettivamente essa faccia; in seconda battuta la disponibilità di software disponibile funge da base di partenza per lo sviluppo di nuove applicazioni: lo sviluppo è quindi meno costoso e più veloce perché non è necessario reimplementare tutto da capo.
- **sociale**: sono vantaggi che non vanno al singolo utente finale ma alla collettività: la diffusione dei risultati e del codice diventa un patrimonio culturale liberamente accessibile; essendo il software libero un business orientato al servizio si incentivano le professionalità e le competenze delle persone (magari anche sul territorio!).

3.2 Software libero ed economia

Software libero non vuol dire software gratis. Il software libero è vendibile, il suo modello economico è un orientato ai servizi: posso farmi commissionare lo sviluppo o la personalizzazione di un software; in alternativa si possono offrire dei servizi di assistenza, supporto, installazione, formazione ecc ecc.

Il software è, di per sé, un *servizio* e non un *prodotto:* si paga per il lavoro effettuato (consulenza, formazione, sviluppo, personalizzazione, assistenza) esattamente come nel caso del lavoro professionale. Non si tratta di un prodotto che si consuma col suo utilizzo.

Il software libero è intrinsecamente *democratico* poiché la base di partenza è identica per tutti gli agenti economici, non ci possono essere rendite di posizione, monopoli o posizioni dominanti. Il mercato del software libero quindi, per il suo carattere di apertura e democraticità, garantisce alti livelli di qualità intrinsechi nei meccanismi della libera concorrenza.

E' possibile inoltre incentivare le risorse del territorio affidando il lavoro a realtà locali.

In questo periodo c'è un grande fermento: al di là dell'interesse generale che si è sviluppato, sempre più imprese adottano linux e si moltiplicano le iniziative per favorire l'ingresso di software libero all'interno delle pubbliche amministrazioni.

Comunque sia è anche un mercato dove è difficile entrare:

- il mercato è ancora ristretto;
- tutto il settore dell'informatico vive una crisi di credibilità;
- sono ancora poche le imprese che operano interamente con il software libero.

3.3 Perché migrare?

3.3.1 Vantaggi

L'impresa che usa software libero gode di una serie di vantaggi pratici:

- non ci sono costi di licenze per l'adozione di software libero; inoltre non ci sono neppure costi di gestione (non ci sono contratti da rinnovare, scadenzari da tenere d'occhio e così via);
- si può personalizzare e riutilizzare il codice a piacere;
- l'impresa ha il pieno controllo della tecnologia impiegata si può quindi investire in sviluppo, formazione e creazione di professionalità;
- l'impresa ha il potere di cambiare il fornitore (inpendenza strategica dal *lock-in*) poiché non ci possono essere vincoli artificiali imposti dal fornitore.

Nello scenario di indipendenza del cliente dal suo fornitore si innesta il discorso dei formati chiusi e/o aperti. Un formato chiuso, ovvero di cui non si conosce il modello di archiviazione dei dati, vincola un cliente al fornitore che utilizza quel tale formato. Se i formati sono aperti chiunque può inserirsi sul mercato: è un vantaggio del cliente che non si trova sottoposto ai capricci ed ai prezzi di un solo fornitore. Un formato chiuso impone invece a chiunque voglia

utilizzare quei dati a rivolgersi a chi implementa i software per la lettura e scrittura di quel tale formato: chiunque entri in contatto con questa realtà viene dunque costretto a rivolgersi a quel fornitore. Inoltre un formato chiuso è una spada di Damocle: chi gode di un brevetto di un formato può imporre, ad esempio, che non si possano realizzare programmi di lettura/scrittura per il suo formato al di fuori di quelli distribuiti in modo ufficiale.

3.3.2 Svantaggi

Sono i classici svantaggi legati al cambiare tecnologie:

- le persone hanno una naturale resistenza ai cambiamenti;
- gli utenti hanno bisogno di essere formati ai nuovi sistemi, è quindi necessario investire in formazione;
- devono esistere delle professionalità che possano sostenere il processo di migrazione.

Le obiezioni comuni infatti sono:

- se qualcosa va storto a chi mi rivolgo?
- ma adesso devo imparare tutto da capo?
- è una cosa nuova: in ditta non c'è nessuno di pratico...

3.3.3 La migrazione

A questi problemi si può ovviare proponendo una *formazione* in aula o sul campo. Ci sono ulteriori difficoltà:

- l'offerta delle soluzioni è talmente ampia da confondere l'utente che non sa cosa scegliere;
- è vero che il software libero è modificabile ma l'utente finale non sa che farsene di questa caratteristica perché non ha queste competenze;
- il software libero è percepito come un software difficile: sono quindi esperti costosi.

Il valore da mettere in luce, in questo caso, è quello della libertà di poter scegliere o di poter delegare a persone di fiducia. Inoltre queste competenze possono essere acquisite liberamente in modo autonomo o pagando per ottenerle. Nel software proprietario invece non si ha questa possibilità: prendere o lasciare. C'è inoltre un problema culturale: spesso l'uitenza non percepisce la distinzione tra hardware e software e si lega ad un fornitore senza esserne consapevole (classico esempio: programmi di fatturazione che dipendono dalla piattaforma sulla quale sono sviluppati).

La formazione risulta quindi uno strumento fondamentale a supporto di una migrazione. Anche se si dispone di una soluzione tecnica perfetta ma l'utente non la sa usare non si ottiene nessun risultato. E' necessario accompagnare gli accorgimenti tecnici ad una sensibilizzazione ed una formazione tecnica che permettano di superare la sensazione di vuoto e di smarrimento che è propria di ogni cambiamento.

Software libero e formazione (parlando tout-court dalle applicazioni di ufficio a quelle di sviluppo) è un binomio che porta in sé alcuni vantaggi:

- non ci sono costi di distribuzione;
- l'accesso alla tecnologia è completo;
- posso rivolgermi al fornitore di competenze che preferisco;
- si può scegliere il grado di approfondimento che si desidera;
- le informazioni sono molte (talvolta troppe), distribuite e condivise.

Ogni caso però deve essere trattato singolarmente perché in base all'oggetto della formazione i vantaggi esposti hanno pesi diversi.

La formazione basata sul software libero porta in sé altri due vantaggi:

- si impara come funziona una tecnologia e non un particolare programma;
- a fronte di una curva di apprendimento più ripida si ottiene un aumento di produttività molto elevata.

Anche nella formazione si presentano svantaggi che bisogna saper affrontare:

- la curva di apprendimento più ripida può scoraggiare e si può creare una sfiducia che ha un effetto deleterio;
- è difficile individuare e confezionare proposte formative di qualità;

- è difficile fare una valutazione del valore della formazione offerta;
- la formazione viene spesso percepita solo come un costo (la persona formata poi magari se ne va, non si vede un beneficio immediato...) e non come un investimento.

Si possono identificare tre livelli di competenze che entrano in gioco in una migrazione al software libero:

1. **competenze degli utenti nell'uso dei nuovi sistemi**: non ha nulla di diverso rispetto alla migrazione da una qualunque soluzione ad una diversa (sia libera o meno), bisogna identificare le esigenze dell'utente, scegliere le opportune soluzioni ed individuare un percorso formativo.
2. **competenze amministrative per mantenere i nuovi sistemi**: il sistemista normalmente ha una curva di apprendimento più rapida però è necessario verificare le competenze acquisite; il sistemista infatti è un elemento chiave e se questa figura non riesce a far funzionare le cose l'utente finale sperimenta solo un disservizio (e non sa discriminare tra problema legato al software e problema legato alla mancata competenza del sistemista); normalmente è bene che questa formazione avvenga prima di una migrazione.
3. **competenze gestionali per governare il processo di migrazione**: è necessaria una figura che abbia una visione d'insieme delle tecnologie disponibili, che abbia comprensione dei problemi politico-giuridici e socio-economici.

Quando si propone a qualcuno una migrazione verso il software libero si ha bisogno di:

- identificare una persona guida che abbia una visione d'insieme delle problematiche del cliente;
- definire una serie di competenze da acquisire per rendere sostenibile il nuovo scenario;
- avere una panoramica completa sulle possibilità disponibili.

In particolare la *Pubbliche Amministrazioni* (come la scuola) può avere un ruolo trainante nelle migrazioni, favorendo la formazione sul software libero può:

- investire sul personale piuttosto che sulle licenze (che spesso vanno all'estero);
- creare professionalità sul territorio;
- sostenere le piccole-medie imprese innescando un meccanismo virtuoso.

Riassumendo quindi i vantaggi "politici" legati al software libero sono:

1. indipendenza tecnologica
2. diffusione della conoscenza
3. abbassamento delle barriere di accesso alla tecnologia
4. stimolo alla concorrenza sana
5. aiuto e supporto alle piccole medie imprese locali

4 La Documentazione

Finora abbiamo parlato di licenze relative al software, ma il software non è tutto. Un ruolo importante lo ricopre la documentazione: esiste una licenza apposita nel settore ed è la **GFDL** (GNU Free Documentation Licence). Wikipedia, ad esempio, è coperta da questa licenza. GFDL utilizza il copyleft, certe parti del documento possono essere modificate, esattamente come nel caso del software; altre parti invece possono essere dichiarate come *invarianti*: ovvero non modificabili. Questo perché mentre per quanto riguarda un manuale tecnico si ha tutto l'interesse a poter modificare i testi per poterli migliorare diverso è il discorso per quanto riguarda le idee "filosofiche": l'autore non vuole che gli si mettano in bocca parole che non gli appartengono. Altre parti possono essere prottette: ad esempio la copertina o il logo che sono proprie di un'azienda.

Tecnicamente la GFDL parla dei sorgenti del documento. La GFDL non è considerata dalla comunità globale come una licenza libera, la discussione è ancora aperta. Oltre la GFDL esiste una licenza chiamata **OpenContent**: molti manuali stampati sono diffusi sotto questa licenza.

Grazie all'iniziativa di un professore americano chiamato Lorence Lessings è nata l'iniziativa **Creative Commons** che è un sistema di licenze (che non necessariamente garantiscono le quattro libertà) modulare e componibile a seconda di ciò che l'autore preferisce che sia fatto con la propria opera. Creative Commons è semplice ed adattabile alle diverse legislazioni nazionali. La GPL, invece, è stata pensata sul piano giuridico americano anche se poi si è dimostrata valida anche in Europa.

Un documento (o video, o audio o altro) rilasciato attraverso la Creative Commons può essere protetto attraverso una serie di licenze:

- **CC** (CreativeCommons): indica un'opera protetta attraverso le CreativeCommons.
- **BY** (*Attribution*): impone che l'autore deve continuare ad essere citato.
- **SA** (*Share Alike*): dispone che si distribuisca l'opera esattamente come si è ricevuta (con la stessa licenza).
- **NC** (*Non Commercial*): non si può fare un uso commerciale di questo documento.
- **ND** (*Non Derivatives)*: non si possono creare opere derivate da questo documento

Creative Commons, per semplificare il processo di pubblicazione, mette a disposizione una guida pratica da seguire per coprire le proprie opere attraverso le licenze di Creative Commons. Per il software la GPL è equivalente a BY-SA. Creative Commons dispone inoltre di un motore di ricerca che permette di cercare tra le opere che sono state rilasciate attraverso le sue licenze. Quello che non può fare Creative Commons è selezionare parti del documento da distribuire con licenze differenti tra di loro.

BY-SA-ND è in pratica il copyleft semplice.

La SIAE (Società Italiana Autori ed Editori) in Italia si occupa di tutelare gli autori: pagando alla SIAE questa si prende l'onere di remunerare l'autore. Con la legge 248/2000 si imponeva che qualsiasi opera si volesse distribuire dovesse essere protetta dalla SIAE (che peraltro è pure privata!), per fortuna poi il regolamento attuativo ha fatto sì che, al limite, si debba notificare la SIAE del fatto che si sta distribuendo un'opera ma non si è obbligati a farsi tutelare dalla SIAE e ad apporre il famoso bollino argentato. In questo modo, anche in Italia, un autore può farsi tutelare attraverso la CreativeCommons piuttosto che dalla SIAE.

In caso di controversie legali ci si può rivolgere al Software Freedom Law Center (che però è americano quindi può aiutare più sul piano teorico che pratico) che offre assistenza legale per quanto riguarda i progetti Free Software e Open Source.

5 Le Distribuzioni

Una distribuzione libera è una collezione di software selezionata attraverso un criterio ben preciso: quello che è distribuito è rigorosamente software libero. Esistono moltissime distribuzioni con una penetrazione diversa sul mercato. Realizzare una distribuzione è un lavoro impegnativo: non si tratta solo di mettere insieme un certo numero di applicazioni ma anche di farli convivere, integrarli e farli comunicare correttamente.

Debian è una distribuzione che contiene appunto solo software libero, ciò che è semilibero viene gestito separatamente dalla distribuzione ufficiale, ciò che è proprietario sta fuori. Debian è una distribuzione molto corposa.

Esistono, come si diceva, distribuzioni con altri scopi e criteri: DebianMed serve per calcolatori in ambito medico, Fuss opera per le strutture scolastiche e così via... altre distribuzioni includono anche software proprietario perché lo ritengono fondamentale per i propri utenti.

Le software house che sviluppano le distribuzioni non sono necessariamente sviluppatrici degli applicativi della distribuzioni: si tratta di due cose diverse; sicuramente viene sviluppato il codice necessario alla gestione della distribuzione. L'obiettivo principale è la facilitazione

Esistono poi distribuzioni commerciali e non commerciali: RedHat ha, ad esempio, una distribuzione sulla quale vende servizi. Debian invece non offre servizi a pagamento: il suo scopo è quello di fornire un sistema operativo completamente gratuito; altri ovviamente ne hanno fatto un business: c'è chi offre assistenza tecnica su Debian, il che è perfettamente lecito in base a quello che si diceva ieri.

Debian è sviluppato in modo completamente volontario e con alti standard di qualità e questo ovviamente si riflette in release che escono in tempi molto lunghi. Debian trova i fondi attraverso donazioni e finanziamenti da società terze.

Lunga divagazione sulla sostenibilità del modello di business: i modelli di business che guidano le imprese che operano nel software libero sono molti: è ovvio che diversificare l'offerta (servizi, sviluppo, assistenza, formazione) mette al riparo dalle crisi del mercato.

Debian è stata scelta all'interno di questo master come distribuzione sulla quale lavorare ed imparare per il suo carattere di indipendenza e libertà che offre ai suoi utenti. Laddove Ubuntu, distribuzione altrettano buona e sicuramente più user-friendly, è invece soggetta al finanziamento di un miliardario che un domani potrebbe cambiare idea...

Debian si appoggia ad una struttura esterna SPI che è un'organizzazione non-profit fondata per aiutare le organizzazioni a sviluppare e distribuire software e hardware libero. SPI però non ha alcun controllo su Debian. Debian è nata nel 1993 da Ian Murdock. Murdock creò il cosidetto "*contratto sociale*" da cui sono derivate le "Linee Guide Debian". Il contratto sociale è stipulato con gli utenti di Debian. Dalle Linee Guida di Debian è stata tratta la Open Source Definition.

5.1 Contratto Sociale con la Comunità Free Software

1. Debian rimarrà libera al 100%

Forniamo le linee guida che usiamo per determinare se un'opera sia "libera" nel documento intitolato "The Debian Free Software Guidelines". Promettiamo che il sistema Debian e tutte le sue componenti rimarranno liberi in accordo con le citate linee guida. Supporteremo le persone che creino o usino in Debian opere sia libere che non libere. Non renderemo mai il sistema dipendente da un componente non libero.

2. Renderemo alla Comunità Free Software

Quando scriviamo nuovi componenti del sistema Debian, li rilasceremo con una licenza che rispetti le Debian Free Software Guidelines. Realizzeremo il sistema migliore che potremo, cosicché le opere libere siano usate e distribuite il più possibile. Comunicheremo cose come bug fix, migliorie e richieste degli utenti agli autori "upstream" delle opere incluse nel nostro sistema.

3. Non nasconderemo i problemi

Manterremo sempre il nostro intero bug report database aperto alla pubblica lettura. I rapporti che le persone invieranno online saranno prontamente resi visibili a tutti.

4. Le nostre priorità sono gli utenti ed il software libero

Ci faremo guidare dai bisogni dei nostri utenti e della comunità del software libero. Metteremo al primo posto i loro interessi. Supporteremo le necessità dei nostri utenti di operare in molti diversi tipi di ambienti di calcolo. Non ci opporremo alle opere non libere che siano state pensate per l'uso in sistemi Debian e non richiederemo compensi a chi crea o usa queste opere. Permetteremo ad altri di creare distribuzioni contenenti sia il sistema Debian che altre opere, senza richiedere compensi. Per raggiungere questi scopi, forniremo un sistema integrato di materiali di alta qualità senza alcuna restrizione legale che limiti qualsiasi uso del sistema.

5. Opere che non rispettano i nostri standard free software

Ci rendiamo conto che alcuni dei nostri utenti richiedono di usare opere non conformi alle Debian Free Software Guidelines. Abbiamo creato le aree "contrib" e "non-free" nel nostro archivi per queste opere. I pacchetti in queste aree non fanno parte del sistema Debian, sebbene siano stati configurati per l'uso con Debian. Invitiamo i realizzatori di CD a leggere le licenze dei pacchetti in queste aree per determinare se possono distribuire i pacchetti sui loro CD. Inoltre, anche se le opere non libere non fanno parte di Debian, supporteremo il loro uso e forniremo infrastrutture per i pacchetti non liberi (come il nostro bug tracking system e le mailing list).

5.2 Debian Free Software Guidelines - DFSG

1. Libera ridistribuzione

La licenza di un componente Debian non può porre restrizioni a nessuno per la vendita o la cessione del software come componente di una distribuzione software aggregata di programmi proveniente da fonti diverse. La licenza non può richiedere royalty o altri pagamenti per la vendita.

2. Codice sorgente

Il programma deve includere il codice sorgente e deve permettere la distribuzione sia come codice sorgente che in forma compilata.

3. Lavori derivati

La licenza deve permettere modifiche e lavori derivati e deve permettere la loro distribuzione con i medesimi termini della licenza del software originale.

4. Integrità del codice sorgente dell'autore

La licenza può porre restrizioni sulla distribuzione di codice sorgente modificato _**solo**_ se permette la distribuzione di "file patch" insieme al codice sorgente con lo scopo di modificare il programma durante la compilazione. La licenza deve esplicitamente permettere la distribuzione di software compilato con codice sorgente modificato. La licenza può richiedere che i lavori derivati abbiano un nome o un numero di versione diversi da quelli del software originali. (Questo è un compromesso. Il gruppo Debian invita tutti gli autori a non impedire che file, sorgenti o binari possano essere modificati.)

5. Nessuna discriminazione di persone o gruppi

La licenza non può discriminare nessuna persona o gruppo di persone.

6. Nessuna discriminazione nei campi di impiego

La licenza non può porre restrizioni all'utilizzo del programma in uno specifico campo di impiego. Per esempio, non può porre restrizioni all'uso commerciale o nella ricerca genetica.

7. Distribuzione della licenza

I diritti applicati al programma devono essere applicabili a chiunque riceva il programma senza il bisogno di utilizzare licenze addizionali di terze parti.

8. La licenza non può essere specifica per Debian

I diritti applicati al programma non possono dipendere dal fatto che esso sia parte di un sistema Debian. Se il programma è estratto da Debian e usato o distribuito senza Debian ma ottemperando ai termini della licenza, tutte le parti alle quali il programma è ridistribuito dovrebbero avere gli stessi diritti di coloro che lo ricevono con il sistema Debian.

9. La licenza non deve contaminare altro software

La licenza non puòporre restrizioni ad altro software che sia distribuito insieme al software concesso in licenza. Per esempio, la licenza non può richiedere che tutti gli altri programmi distribuiti con lo stesso supporto debbano essere software libero.

10. Esempi di licenze

Le licenze "*GPL*", "*BSD*" e "*Artistic*" sono esempi di licenze che consideriamo "libere".

I concetti enunciati nel nostro "contratto sociale con la comunità del software libero" furono proposti da Ean Schuessler. Questo documento fu abbozzato da Bruce Perens, rifinito da altri sviluppatori Debian durante una conferenza via posta elettronica durata un mese (Giugno 1997) ed infine approvata come pubblica linea di condotta del Progetto Debian.

Bruce Perens in seguito ha rimosso i riferimenti specifici a Debian dalle Linee Guida Debian per il Free Software per creare "L'Open Source Definition".

6 Riferimenti

- Associazione Software Libero `http://www.softwarelibero.it`
- **Free Software Foundation** `http://www.fsf.org`
- **Progetto GNU** `http://www.gnu.org`
- **Creative Commons** `http://www.creativecommons.org`
- **Software Freedom Law Center** `http://www.softwarefreedom.org`
- **DistroWatch** `http://www.distrowatch.com`
- **La Cattedrale e il Bazar** `http://it.wikisource.org/wiki/La_cattedrale_e_il_bazaar`