Perché dovresti usare una licenza in stile BSD per il tuo progetto Open Source

Indice

1. Introduzione . . . 1
2. Storia molto breve dell'Open Source . . . 1
3. Unix dal punto di vista delle licenze BSD . . . 2
4. Lo stato attuale di FreeBSD e delle licenze BSD . . . 3
5. Le origini della GPL . . . 3
6. Le origini di Linux e della LGPL . . . 4
7. Licenze Open Source e il problema dell'Orphaning . . . 4
8. Ciò che una licenza non può fare . . . 5
9. Vantaggi e svantaggi della GPL . . . 5
10. Vantaggi di BSD . . . 6
11. Raccomandazione specifiche per l'uso di una licenza BSD . . . 7
12. Conclusione . . . 9
13. Riferimenti bibliografici . . . 9

1. Introduzione

Questo documento promuove l'uso di una licenza in stile BSD per programmi e dati; in particolare raccomanda l'uso di una licenza in stile BSD al posto della GPL. Esso può anche essere interpretato come un'introduzione e un riassunto comparativo delle licenze Open Source BSD e GPL.

2. Storia molto breve dell'Open Source

Molto prima che il termine "Open Source" fosse usato, esisteva del software che veniva sviluppato da associazioni informali di programmatori e che veniva liberamente scambiato. A cominciare dai primi anni Cinquanta, organizzazioni quali SHARE e DECUS sviluppavano molto del software che le aziende di hardware informatico vendevano incluso nelle loro offerte. In effetti, a quel tempo le aziende informatiche vendevano hardware; qualsiasi cosa che riducesse il costo del software e rendesse disponibile un maggior numero di programmi rendeva tali aziende più competitive.

Questo modello cambiò negli anni Sessanta. Nel 1965 ADR sviluppò il primo prodotto con licenza software indipendente dall'hardware dell'azienda. ADR competeva contro un pachetto gratuito

dell'IBM originalmente sviluppato dai clienti di IBM. ADR brevettò il proprio software nel 1968. Per impedire la condivisione del suo programma, lo distribuì a condizione di noleggio di materiale il cui pagamento si estendeva per tutta la durata di vita del prodotto. ADR ne mantenne così la proprietà e poteva controllarne la rivendita e il riuso.

Nel 1965 il Dipartimento di Giustizia degli Stati Uniti accusò IBM di concorrenza sleale offrendo software gratuito incluso nel prezzo di hardware IBM. In seguito al processo, IBM escluse il suo software; cioè, i programmi divennero prodotti separati dall'hardware.

Nel 1968 Informatics introdusse la prima applicazione di successo e rapidamente stabilì i concetti di prodotto software e azienda software, nonché alti tassi di ricavo. Informatics sviluppò la licenza perpetua che è ora uno standard nell'industria informatica e che prevede che la proprietà non venga mai trasferita al cliente.

3. Unix dal punto di vista delle licenze BSD

AT&T, prorietario dell'implementazione originale di Unix, era un monopolio pubblicamente regolato e vincolato da una causa antitrust; non poteva vendere legalmente prodotti nel mercato del software. Poteva, tuttavia, distribuire software ad istituzioni accademiche al prezzo del supporto.

Le università adottarono rapidamente Unix dopo che una conferenza sui sistemi operativi ne pubblicizzò la disponibilità. Fu estremamente utile il fatto che Unix girasse su PDP-11, un computer 16-bit molto economico, e fosse codificato in un linguaggio di alto livello che era dimostrabilmente buono per la programmazione di sistemi. Il DEC PDP-11 aveva, in effetti, un'interfaccia hardware aperta progettata per rendere facile per i propri clienti la scrittura dei propri sistemi operativi, un'operazione allora comune. Come il fondatore di DEC Ken Olsen proclamò con una celebre dichiarazione "il software viene dal cielo quando hai un buon hardware".

L'autore di Unix Ken Thompson tornò alla sua alma mater, l'Università di Berkeley in California (UCB), nel 1975 e tenne corsi sul kernel insegnandolo riga per riga. Questo risultò infine in un sistema in evoluzione noto come BSD (Berkeley Standard Distribution). L'UCB convertì Unix a 32-bit, aggiunse la memoria virtuale e implementò la versione dello stack TCP/IP sulla quale sostanzialmente fu costruito Internet. L'UCB rese BSD disponibile al costo del supporto, sotto quella che divenne nota come la "licenza BSD". Un cliente comprava Unix da AT&T e quindi ordinava un nastro BSD dall'UCB.

A metà degli anni Ottanta una causa antitrust del governo contro AT&T si concluse con la divisione dell'azienda in diverse società. AT&T possedeva ancora Unix ed era ora autorizzata a venderlo. AT&T intraprese uno sforzo di licenziamento aggressivo a la maggior parte degli Unix commerciali di allora divennero derivati di AT&T.

Nei primi anni Novanta, AT&T querelò l'UCB per violazione di licenza in relazione a BSD. L'UCB scoprì che AT&T aveva incorporato, senza riconoscimento né pagamento, molti miglioramenti dovuti a BSD nei prodotti di AT&T, e ne seguì una lunga causa, principalmente tra AT&T e l'UCB. In questo periodo alcuni programmatori dell'UCB intrapresero un progetto di riscrittura di ogni codice AT&T associato a BSD. Questo progetto risultò in un sistema chiamato BSD 4.4-lite (lite perché non era un sistema completo; mancavano 6 file AT&T chiave).

Una lunga serie di articoli pubblicati poco dopo nella rivista Dr. Dobbs descriveva una versione di Unix per PC 386 derivata da BSD, con file sotto licenza BSD per la sostituzione dei 6 file mancanti da 4.4-lite. Questo sistema, chiamato 386BSD, era opera del programmatore William Jolitz, in precedenza nell'UCB. Esso divenne la base originale di tutti i BSD per PC oggi in uso.

A metà degli anni Novanta, Novell comprò i diritti di Unix da AT&T e fu trovato un accordo (allora segreto) per porre fine alla causa. L'UCB interruppe il supporto per BSD poco tempo dopo.

4. Lo stato attuale di FreeBSD e delle licenze BSD

La cosiddetta nuova licenza BSD applicata a FreeBSD negli ultimi anni è di fatto una dichiarazione per la quale si può fare qualsiasi cosa col programma o il suo sorgente, ma non si riceve nessuna garanzia e nessuno degli autori ha nessuna responsabilità (sostanzialmente, non è possibile fare causa a nessuno). Questa nuova licenza BSD è intesa per incoraggiare la commercializzazione di prodotti. Qualsiasi codice BSD può essere venduto o incluso in prodotti proprietari senza restrizioni sulla disponibilità del codice o su comportamenti futuri.

Non si confondano la nuova licenza BSD con il "dominio pubblico". Sebbene anche un oggetto di dominio pubblico sia libero di essere usato da chiunque, esso non ha proprietario.

5. Le origini della GPL

Mentre il futuro di Unix era così farraginoso nei tardi anni Ottanta e nei primi anni Novanta, la GPL, un altro sviluppo con importanti considerazioni in materia di licenza, giunse a compimento.

Richard Stallman, il programmatore di Emacs, era un membro del personale del MIT quando il suo laboratorio sostituì i suoi sistemi prodotti internamente con prodotti proprietari. Stallman si innervosì quando scoprì di non poter apportare legalmente piccoli miglioramenti al sistema. (Molti dei collaboratori di Stallman se ne erano andati per formare due aziende basate su software sviluppato al MIT e licenziato dal MIT; pare che ci sia stato disaccordo sull'accesso al codice sorgente di questo software). Stallman ideò un'alternativa alla licenza software commerciale e la chiamò GPL, o "Gnu Public License". Creò anche una fondazione senza scopo di lucro, la Free Software Foundation (FSF), che intendeva sviluppare un intero sistema operativo, software associato incluso, che non sarebbe stato oggetto di licenza proprietaria. Il sistema fu chiamato GNU, stante per "GNU non è Unix".

La GPL fu progettata per essere l'antitesi della licenza proprietaria standard. A questo fine, si richiese che ogni modifica che fosse apportata ad un programma GPL fosse restituita alla comunità GPL (richiedendo che il sorgente del programma fosse reso disponibile all'utente) e che ogni programma che usasse o linkasse codice GPL venisse sottoposto alla GPL. La GPL intendeva impedire che il software diventasse proprietario. All'ultimo paragrafo la GPL dichiara:

"Questa Licenza Pubblica Generale non permette di incorporare il tuo programma in programmi proprietari."

La GPL è una licenza complessa quindi ecco alcune regole pratiche sul suo utilizzo:

- è possibile fissare qualsivoglia prezzo per la distribuzione, il supporto o la documentazione del software, ma non è possibile la vendita del software stesso.
- la regola generale stabilisce che se è necessario codice GPL per la compilazione di un programma, il programma deve essere sottoposto alla GPL. Il link statico ad una libreria GPL implica che il programma sia sottoposto alla GPL.
- la GPL richiede che qualsiasi brevetto associato a software GPL debba essere sottoposto ad una licenza che ne garantisca il libero uso da parte di chiunque.
- la semplice aggregazione di software, come il porre programmi diversi su uno stesso disco, non conta quale inclusione di programmi GPL in programmi non GPL.
- l'output di un programma non conta quale lavoro derivato. Questo consente al compilatore gcc di essere usato in ambienti commerciali senza problemi legali.
- siccome il kernel Linux è sottoposto alla GPL, ogni codice linkato staticamente col kernel Linux deve essere sottoposto alla GPL. Questo requisito può essere aggirato linkando dinamicamente moduli del kernel. Questo permette alle aziende di distribuire driver binari, ma spesso con lo svantaggio che essi funzioneranno solo per versioni specifiche del kernel Linux.

A causa della sua complessità, in molte parti del mondo oggigiorno gli aspetti legali della GPL vengono ignorati per quanto riguarda Linux e software relativo. Le conseguenze a lungo termine di questo approccio non sono chiare.

6. Le origini di Linux e della LGPL

Mentre imperversavano le guerre tra gli Unix commerciali, il kernel Linux veniva sviluppato come un clone di Unix per PC. Linus Torvalds riconosce l'importanza del compilatore GNU C e degli strumenti GNU associati per l'esistenza di Linux. Egli sottopone il kernel Linux alla GPL.

Si ricordi che la GPL impone che ogni cosa linkata staticamente ad un qualsiasi codice sottoposto alla GPL debba essere sottoposta anch'essa alla GPL. Il sorgente per questa cosa deve dunque essere reso disponibile all'utente del programma. Il link dinamico, tuttavia, non è considerato una violazione della GPL. La pressione per inserire applicazioni proprietarie in Linux divenne schiacciante. Tali applicazioni devono spesso effettuare link con librerie di sistema. Ciò portò ad una versione modificata della GPL chiamata LGPL ("Library", da allora rinominata "Lesser", GPL). La LGPL consente di linkare codice proprietario con la libreria C di GNU, glibc. Non è necessario rilasciare il codice sorgente che è stato linkato dinamicamente ad una libreria sottoposta alla LGPL.

Se si linka staticamente un'applicazione con glibc, come è spesso necessario in sistemi integrati, non è possibile mantenere la propria applicazione proprietaria, vale a dire, il sorgente deve essere pubblicato. Sia la GPL che la LGPL impongono che ogni modifica al codice direttamente sottoposto alla licenza venga pubblicata.

7. Licenze Open Source e il problema dell'Orphaning

Uno dei gravi problemi associati al software proprietario è noto come "orphaning" (letteralmente:

"rendere orfano"). Esso si presenta quando un'impresa fallisce o un cambiamento nella strategia dei prodotti causa il fallimento di una vasta piramide di sistemi dipendenti e di aziende per ragioni al di là del loro controllo. Decadi di esperienza hanno dimostrato che la dimensione o il successo momentanei di un distributore di software non garantiscono che il loro software rimanga disponibile, dato che le condizioni del mercato e le strategie possono cambiare rapidamente.

La GPL tenta di impedire l'orphaning spezzando il legame con la proprietà intellettuale privata.

Una licenza BSD consegna all'azienda il software in una specia di acconto di garanzia senza complicazioni o costi legali. Se un programma sottoposto a licenza BSD diventa orfano, un'azienda può semplicemente subentare, in maniera proprietaria, nel mantenimento del programma da cui dipende. Una situazione anche migliore si presenta quando codice BSD viene mantenuto da un piccolo consorzio informale, siccome il processo di sviluppo non dipende dalla sopravvivenza di una singola azienda o di una linea di prodotti. La sopravvivenza di una squadra di sviluppo mentalmente concentrata è molto più importante della semplice disponibilità fisica del codice sorgente.

8. Ciò che una licenza non può fare

Nessuna licenza può garantire la futura disponibilità del software. Benché il detentore di copyright possa tradizionalmente cambiare i termini del copyright stesso in qualsiasi momento, si presuppone nella comunità BSD che tale tentativo causi semplicemente il fork del sorgente.

La GPL vieta esplicitamente la revoca della licenza. È accaduto, tuttavia, che un'azienda (Mattel) avesse comprato un copyright GPL (cphack), revocato l'intero copyright e vinto in tribunale [2]. Cioè, essa ha legalmente revocato l'intera distribuzione e tutti i lavori derivati basati su quel copyright. Se ciò possa accadere con una distribuzione più ampia e distribuita è una questione aperta; c'è anche qualche confusione riguardo a se il software fosse veramente sottoposto alla GPL.

In un altro esempio, Red Hat comprò Cygnus, un'azienda di ingegneria che era subentrata nello sviluppo degli strumenti del compilatore della FSF. Cygnus era autorizzata a svolgere questa operazione perché aveva sviluppato un modello commerciale che prevedeva la vendita di supporto al software GNU. Questo le permise di assumere una cinquantina di programmatori e guidare la direzione dei programmi contribuendo alla maggior parte delle modifiche. Come dichiara Donald Rosenberg "progetti che usano licenze quali la GPL... vivono sotto la costante minaccia di vedersi soffiar via il progetto da qualcuno che produce una versione migliore del codice e lo fa più velocemente dei proprietari originali." [3]

9. Vantaggi e svantaggi della GPL

Una ragione comune per l'uso della GPL è la modifica o l'estensione del compilatore gcc. Ciò è particolarmente conveniente quando si lavora con processori molto particolari in ambienti in cui è probabile che tutti i costi del software siano considerati spese di gestione, con minime aspettative riguardo all'uso da parte di altri del compilatore risultante.

La GPL è vantaggiosa per piccole aziende che vendono CD in un ambiente dove "compra basso, vendi alto" può ancora consegnare all'utente finale un prodotto molto economico. Essa è

vantaggiosa anche per le aziende che contano di sopravvivere fornendo varie forme di supporto tecnico, documentazione inclusa, per il mondo della proprietà intellettuale GPL.

Un uso meno pubblicizzato e imprevisto della GPL consiste nell'essere molto favorevole a grandi aziende che vogliono tagliar fuori aziende del software. In altre parole, la GPL è adatta per essere utilizzata come un'arma di mercato, potenzialmente riducendo il beneficio economico globale e contribuendo ad un comportamento monopolistico.

La GPL può costituire un vero problema per coloro che vogliono commercializzare software e trarne profitto. Per esempio, la GPL accresce la difficoltà di uno studente laureato di fondare un'azienda per commercializzare i risultati delle sue ricerche e accrese altresì la difficoltà di uno studente di entrare in un'azienda a condizione che un suo progetto di ricerca promettente venga commercializzato.

Per coloro che devono lavorare con implementazioni linkate staticamente di diversi standard software, la GPL è spesso una licenza infelice, perché preclude l'uso di implementazioni proprietarie degli standard. Così la GPL riduce il numero di programmi che possono essere costruiti usando uno standard GPL. La GPL intenzionalmente non fornisce un meccanismo per sviluppare uno standard sul quale si possano sviluppare prodotti proprietari. (Ciò non si applica alle applicazioni Linux perché non si collegano staticamente al kernel, bensì usano un'API trap-based.)

La GPL cerca di portare i programmatori a contribuire ad una collezione di programmi in evoluzione, quindi di competere nella distribuzione e nel supporto di questa collezione. Questa situazione è irrealistica per molti standard di sistema di base richiesti, che potrebbero essere applicati in ambienti ampiamente variabili richiedenti personalizzazioni commerciali o integrazioni con standard legacy sottoposti a licenze già esistenti (non GPL). I sistemi in tempo reale sono spesso linkati staticamente, quindi la GPL e la LGPL sono decisamente considerate potenziali problemi da molte aziende di sistemi integrati.

La GPL è un tentativo di mantenere gli sforzi, senza riguardo per la domanda, ai livelli della ricerca e dello sviluppo. Questo massimizza i benefici per i ricercatori e gli sviluppatori, ad un costo ignoto per coloro che guadagnerebbero da una distribuzione più ampia.

La GPL è stata progettata per impedire che i risultati di ricerca diventassero prodotti proprietari. Si suppone spesso che questo passo sia l'ultimo nel processo tradizionale di trasferimento tecnologico ed è solitamente abbastanza difficile anche nelle migliori delle circostanze; la GPL lo rende intenzionalmente impossibile.

10. Vantaggi di BSD

Una licenza in stile BSD è una buona scelta per ricerche di lunga durata o altri progetti che necessitano di un ambiente di sviluppo tale che:

- abbia costi quasi nulli
- evolva in un lungo periodo di tempo
- permetta a chiunque di mantenere la possibilità di commercializzare i risultati finali con problemi legali minimi.

Quest'ultima considerazione può essere spesso quella dominante, come fu il caso quando il progetto Apache decise la sua licenza:

"Questo tipo di licenza è ideale per promuovere l'uso di un codice di riferimento che implementi un protocollo per un servizio comune. Questa è un'altra ragione per cui l'abbiamo scelto per il gruppo Apache - molti di noi volevano vedere HTTP sopravvivere e diventare un vero standard comune e non si sarebbero preoccupati minimamente se Microsoft o Netscape avessero scelto di incorporare il nostro motore HTTP o qualsiasi altra componente del nostro codice nei loro prodotti, se fosse stato ulteriormente utile allo scopo di mantenere HTTP comune... Tutto ciò significa che, strategicamente parlando, il progetto necessita di mantenere slancio sufficiente e che i partecipanti realizzano valore maggiore condividendo il loro codice col progetto, anche quel codice che avrebbe avuto valore se mantenuto proprietario."

I programmatori tendono a trovare la licenza BSD vantaggiosa per il fatto che tiene i problemi legali fuori dai piedi e lascia loro fare ciò che vogliono con il codice. Al contrario, coloro che contano principalmente di utilizzare un sistema piuttosto che di programmarlo, o contano che altri facciano evolvere il codice, o coloro che non contano di vivere del loro lavoro associato al sistema (quali gli impiegati di governi), trovano la GPL vantaggiosa, perché fa sì che il codice sviluppato da altri sia loro restituito e impedisce il loro datore di lavoro di mantenere il copyright e così potenzialmente di "seppellire" il software o di renderlo orfano. Se si desidera costringere i concorrenti ad aiutare, la GPL è vantaggiosa.

Una licenza BSD non è semplicemente un regalo. La domanda "perché dovremmo aiutare i nostri concorrenti o lasciarli rubare il nostro lavoro?" sorge spesso riguardo ad una licenza BSD. Secondo la licenza BSD, se un'azienda è venuta a dominare una nicchia di mercato che altri consideravano strategica, le altre aziende possono, con uno sforzo minimo, formare un mini-consorzio mirato a riequilibrare le parti contribuendo ad una variante BSD competitiva che aumenta la concorrenza di mercato e l'equità. Ciò permette ad ogni azienda di ritenere che sarà in grado di trarre profitto da un qualche vantaggio che è capace di fornire, nonché di contribuire alla flessibilità e all'efficienza economica. Più rapidamente e facilmente i membri cooperanti sono in grado di attuare ciò, più successo conseguiranno. Una licenza BSD è essenzialmente una licenza complicata il minimo necessario da consentire tale comportamento.

Un effetto chiave della GPL, la costruzione di un sistema Open Source completo e competitivo ampiamente disponibile al prezzo del supporto, è uno scopo ragionevole. Una licenza in stile BSD, in congiunzione con consorzi ad hoc ed individui, può raggiungere questo scopo senza distruggere i presupposti economici costruiti attorno al termine del processo di trasferimento tecnologico, vale a dire la distribuzione.

11. Raccomandazione specifiche per l'uso di una licenza BSD

- La licenza BSD è preferibile per trasferire risultati di ricerca in modo che vengano ampiamente distribuiti e che benefici l'economia. Pertanto, agenzie di finanziamento di ricerca, quali le americane NSF, ONR, DARPA o, in Italia, il CNR, dovrebbero incoraggiare, nelle fasi iniziali dei progetti di ricerca finanziati, l'adozione di licenze in stile BSD per software, dati, risultati e hardware aperto. Dovrebbero incoraggiare anche la formazione di standard basati su sistemi

implementati Open Source e progetti Open Source in sviluppo.

- Le politiche di governo dovrebbero minimizzare i costi e le difficoltà nel pasaggio dalla ricerca alla distribuzione. Quando possibile, le sovvenzioni dovrebbero richiedere che i risultati vengano resi disponibili secondo le condizioni di una licenza compatibile con la commercializzazione in stile BSD.
- In molti casi, i risultati a lungo termine di una licenza in stile BSD riflettono più precisamente gli obiettivi proclamati nella carta di ricerca delle università rispetto a quanto accade quando i risultati sono sottoposti a copyright o brevetti e soggetti a licenze proprietarie di università. Aneddoti dimostrano che le università vengono ricompensate meglio finanziariamente nel lungo termine rilasciando i propri risultati di ricerca e quindi esortando a versare donazioni gli ex allievi che vantano successi commerciali.
- Le aziende hanno riconosciuto da molto tempo che la creazione di standard de facto è una tecnica di commercializzazione chiave. La licenza BSD interpreta bene questo ruolo, se un'azienda ha veramente un vantaggio unico nell'evoluzione del sistema. La licenza è legalmente vantaggiosa per il pubblico più ampio mentre l'esperienza dell'azienda assicura il controllo dello standard. Esistono casi in cui la GPL può essere il veicolo appropriato per creare uno standard del genere, soprattutto quando si cerca di sabotare o cooptare altri. La GPL, tuttavia, penalizza l'evoluzione dello standard, perché promuove una collezione di software piuttosto che uno standard commercialmente applicabile. L'uso di una tale collezione solleva costantemente problemi legali e di commercializzazione. Può non essere possibile mescolare standard quando alcuni sono sottoposti alla GPL mentre altri non lo sono. Un vero standard tecnico non dovrebbe imporre esclusioni di altri standard per ragioni non tecniche.
- Aziende interessate a promuovere uno standard in evoluzione, che potrebbe diventare il cuore dei prodotti commerciali di altre aziende, dovrebbero diffidare della GPL. Indipendentemente dalla licenza usata, il software risultante sarà generalmente trasferito a chiunque compia effettivamente la maggioranza dei cambiamenti tecnici e capisce meglio lo stato del sistema. La GPL semplicemente aggiunge più attrito legale al risultato.
- Grandi aziende, in cui viene sviluppato codice Open Source, dovrebbero essere a conoscenza del fatto che i programmatori apprezzano l'Open Source perché lascia il software disponibile all'impiegato quando cambiano datore di lavoro. Alcune aziende incoraggiano questo comportamento come un vantaggio dell'impiego, soprattuto quando il software in questione non è direttamente strategico. Si tratta, in effetti, di un benificio anticipato dell'interruzione del rapporto di lavoro con costi di perdita potenziali ma senza costi diretti. Incoraggiare gli impiegati a lavorare per colleghi al di fuori dell'azienda è un beneficio trasferibile poco costoso che un'azienda può a volte fornire quasi senza svantaggi.
- Piccole aziende con progetti software vulnerabili all'orphaning dovrebbero cercare di usare la licenza BSD quando possibile. Aziende di tutte le dimensioni dovrebbero considerare di formare tali progetti Open Source quando il mantenimento al minimo dei problemi legali e dei costi generali associati ad un vero progetto Open Source in stile BSD costituisce un vantaggio per tutte le parti.
- Le organizzazioni senza fini di lucro dovrebbero partecipare a progetti Open Source quando possibile. Per evitare problemi di programmazione del software, come il mescolamento di codici sottoposti a licenze diverse, bisognerebbe incoraggiare l'uso di licenze in stile BSD. Essere cauti con la GPL è particolarmente indicato nel caso di organizzazioni senza scopo di lucro che interagiscono con il mondo della programmazione. In località in cui l'applicazione della legge

diventa un esercizio costoso, la semplicità della nuova licenza BSD, paragonata alla GPL, può costituire un vantaggio considerevole.

12. Conclusione

Al contrario della GPL, che è progettata per impedire la commercializzazione proprietaria di codice Open Source, la licenza BSD pone restrizioni minime su comportamenti futuri. Questo consente al codice BSD di rimanere Open Source o di venire integrato in soluzioni commerciali, dato che i bisogni di un progetto o di un'azienda cambiano. In altre parole, la licenza BSD non diventa una bomba ad orologeria legale in nessun punto del processo di sviluppo.

Inoltre, siccome la licenza BSD non comporta la complessità legale delle licenze GPL e LGPL, consente ai programmatori e alle aziende di impiegare il loro tempo nel creare e promuovere codice di qualità invece di preoccuparsi riguardo alla possibilità che tale codice violi qualche licenza.

13. Riferimenti bibliografici


```
[1] http://www.gnu.org/licenses/gpl.html

[2] http://archives.cnn.com/2000/TECH/computing/03/28/cyberpatrol.mirrors/

[3] Open Source: the Unauthorized White Papers, Donald K. Rosenberg, IDG Books,
    2000. Citazioni da pagina 114, ``Effects of the GNU GPL''.

[4] Nella sezione "What License to Use?" di
    http://www.oreilly.com/catalog/opensources/book/brian.html

Questo documento è un riassunto del lavoro originale disponibile a
http://alumni.cse.ucsc.edu/~brucem/open_source_license.htm
```