Le licenze software

Eusebi Giampaolo

State University of Milan

12 maggio 2008

Nascita del modello closed-source

- Nascondere il funzionamento del programma e proteggere dalla copia
- La fidelizzazione di utenti e sviluppatori
- La sicurezza intrinseca del modello closed-source

Le prime licenze

- Le licenze non libere
- Le licenze meno restrittive

Il bisogno di una licenza libera

- La nascita del free software
- Il progetto GNU e il concetto di copyleft

L'avvento della GPL

- GNU, Hurd e Linux
- La licenza GPL
- La Library GPL e la Lesser GPL
- La MIT-license e la BSD-license
- La proliferazione delle licenze

La nascita del concetto di closed-source

Il concetto di software closed-source è nato per diversi motivi, molti dei quali a carattere economico o strategico per le aziende che tendevano a massimizzare i propri bacini di utenza e ad aumentare il loro profitto. Alcuni dei principali sono:

La nascita del concetto di closed-source

Il concetto di software closed-source è nato per diversi motivi, molti dei quali a carattere economico o strategico per le aziende che tendevano a massimizzare i propri bacini di utenza e ad aumentare il loro profitto. Alcuni dei principali sono:

- Rendere difficile al resto del mondo sapere come lavora il proprio software

La nascita del concetto di closed-source

Il concetto di software closed-source è nato per diversi motivi, molti dei quali a carattere economico o strategico per le aziende che tendevano a massimizzare i propri bacini di utenza e ad aumentare il loro profitto. Alcuni dei principali sono:

- Rendere difficile al resto del mondo sapere come lavora il proprio software
- Proteggere dalla copia le proprietà intellettuali riguardanti il software (non è permesso brevettare algoritmi in tutto il mondo)

La nascita del concetto di closed-source

Il concetto di software closed-source è nato per diversi motivi, molti dei quali a carattere economico o strategico per le aziende che tendevano a massimizzare i propri bacini di utenza e ad aumentare il loro profitto. Alcuni dei principali sono:

- Rendere difficile al resto del mondo sapere come lavora il proprio software
- Proteggere dalla copia le proprietà intellettuali riguardanti il software (non è permesso brevettare algoritmi in tutto il mondo)
- Fidelizzazione degli utenti

La nascita del concetto di closed-source

Il concetto di software closed-source è nato per diversi motivi, molti dei quali a carattere economico o strategico per le aziende che tendevano a massimizzare i propri bacini di utenza e ad aumentare il loro profitto. Alcuni dei principali sono:

- Rendere difficile al resto del mondo sapere come lavora il proprio software
- Proteggere dalla copia le proprietà intellettuali riguardanti il software (non è permesso brevettare algoritmi in tutto il mondo)
- Fidelizzazione degli utenti
- Fidelizzazione degli sviluppatori

La nascita del concetto di closed-source

Il concetto di software closed-source è nato per diversi motivi, molti dei quali a carattere economico o strategico per le aziende che tendevano a massimizzare i propri bacini di utenza e ad aumentare il loro profitto. Alcuni dei principali sono:

- Rendere difficile al resto del mondo sapere come lavora il proprio software
- Proteggere dalla copia le proprietà intellettuali riguardanti il software (non è permesso brevettare algoritmi in tutto il mondo)
- Fidelizzazione degli utenti
- Fidelizzazione degli sviluppatori
- Sicurezza

Un modo per proteggere il proprio lavoro

Per le aziende dare modo al resto del mondo di studiare il codice che producevano era un grande problema.
Se un'azienda creava un nuovo prodotto cercava di nasconderne l'implementazioni il più possibile.
Il motivo era che se una azienda dedicava decine o centinaia di ore/uomo per creare un algoritmo efficiente ed innovativo non poteva permettersi che le aziende rivali ne apprendessero i segreti con troppa facilità.

Fidelizzare gli utenti

Creare un prodotto che non fosse in grado di lavorare con i prodotti di altre aziende e distribuirlo ad un numero sufficiente di utenti era un buon metodo per tagliare fuori la concorrenza.
Un utente che lavorava con un word processor e che lo voleva cambiare spesso si trovava davanti a problemi non indifferenti come:

- Riscrivere tutto il lavoro corrente o semplicemente perderlo
- Cambiare sistema operativo e macchine

Fidelizzare gli sviluppatori

Allo stesso modo era molto utile fidelizzare anche gli sviluppatori.
Se la maggior parte degli sviluppatori vengono portati ad utilizzare gli ambienti di una determinata azienda ed i suoi linguaggi, allora quell'azienda si troverà in una posizione di vantaggio.
Un gruppo di sviluppatori che crea una suite di programmi per un determinato sistema operativo lega chiunque voglia utilizzare i loro programmi a quel determinato sistema operativo.

Sicurezza

Dare accesso al proprio codice sorgente rende possibile a persone malevoli scoprirne i difetti con più facilità.
Il concetto del "Security through obscurity" è un esempio importante.
Se un'azienda decide di utilizzare un metodo di cifratura delle password molto debole perchè non è in grado di crearne uno migliore, troverà conveniente nascondere questa informazione al mondo.

Le licenze non libere

Per questi ed altri motivi le grandi aziende di circa 30 anni fà decisero di adottare la distribuzione dei soli binari dei propri software, nascondendone l'implementazione ad ogni costo ed imponendo restrizioni agli utilizzatori al fine di bloccare ogni tentativo di compromettere la loro posizione.
Molte di queste restrizioni però non corrispondevano alle esigenze di molte persone.

- Non si poteva studiare il software
- Non si poteva fare reverse engineering
- Non si poteva modificare il software neanche per migliorarlo
- Non si poteva redistribuirlo

Questo modo di agire però fece sentire alcune categorie di utenti insoddisfatte.
Ed oltretutto la licenza alla quale si aderiva al momento dell'acquisto di questi prodotti non somigliava ad un acquisto: "Il software che io ti rilascio non è tuo ma ti concedo di usarlo".

Le licenze meno restrittive

In ambito accademico i sistemi operativi preferiti erano i sistemi UNIX e UNIX like e benchè non fossero totalmente liberi venivano a volte rilasciati con il codice sorgente e con diritti di utilizzo più orientati all'utente finale. In alcuni casi si dava la possibilità di utilizzare il software dietro compenso solo quando lo si utilizzava a scopo di lucro.

Il caso di TECO

Alla fine degli anni 70 Richard Stallman era studente al MIT e creò un programma di eloborazione testi chiamato TECO.
Era un programma incredibilmente potente, ma era molto difficile da usare, tanto difficile da essere considerato uno dei meno usabili.
Visto che era più un linguaggio di programmazione che un semplice text editor, Stallman creò una serie di macro, trasformandolo così in Emacs, un editor potente come TECO ma molto più facile da usare.

Allo stesso tempo i sistemi Unix divennero molto popolari e gli utenti sentirono l'esigenza di avere TECO anche sotto sistemi Unix-like.
Fu così che James Gosling (creatore di Java) riscrisse Emacs per Unix, ma invece di rilasciarlo alla comunità di sviluppatori così come lo aveva preso, decise di commercializzarlo, rivenderlo e di applicare un Copyright al software.
Richard Stallman naturalmente non ne fu entusiasta.

Queste circostanze spinsero Stallman alla creazione di un sistema software completo che fosse libero.
Si iniziò a garantire quattro libertà fondamentali:

- 0 Libertà di eseguire il programma per qualsiasi scopo
- 1 Libertà di studiare il programma e modificarlo
- 2 Libertà di copiare il programma in modo da aiutare il prossimo
- 3 Libertà di migliorare il programma e di distribuirne pubblicamente i miglioramenti

Si rese dunque necessario coniare nuovi termini per descrivere la licenza che sarebbe stata adottata per questo genere di software.

Per poter proteggere adeguatamente i sorgenti e le idee alla base dei software liberi, Stallman diede vita ad un progetto chiamato GNU.
GNU è un acronimo ricorsivo che significa Gnu's Not Unix, ed infatti GNU sarebbe diventato un sistema software del tutto simile a Unix, anche se di fatto non lo era.
Iniziò quindi la stesura di tutti i componenti del nuovo sistema operativo GNU, rilasciati con una specie di licenza libera descritta nel "GNU manifesto", un documento lungo e complicato che dettava con quali termini il software doveva essere trattato.

In questo periodo venne coniata anche una nuova parola per descrivere l'atto di rilasciare un software con una licenza libera.
Il "Copyleft" che significa copia lasciata (tradotto anche come copia sinistra), che è un gioco di parole per contrapporlo al "Copyright" che significa diritto di copia (anche tradotto copia destra).

Un evento importante per la diffusione del sistema GNU è stato il momento in cui la comunità ha avuto a disposizione un sistema operativo usabile e totalmente libero.
Questo però non sembrò possibile, anche se il sistema GNU era maturo, una componente fondamentale non era ancora stata completata.
Hurd era il kernel in via di sviluppo dalla comunità di sviluppatori che partecipava al progetto.
Si rivelò difficilissimo completare Hurd, un micro kernel che doveva essere degno di essere messo in competizione con i sistemi operativi commerciali del momento.

Stallman alla fine degli anni 90, dovette iniziare a rassegnarsi al fatto che forse il sistema operativo libero non sarebbe mai stato partorito (Hurd è tutt'ora in fase di sviluppo).
Ogni volta che [un ingegnere del software] dice "nessuno si prenderà la briga di fare quella cosa", c'è un ragazzino in Finlandia che si prenderà la briga di farla - Alex Mayfield

In quel periodo, infatti, dall'altra parte dell'oceano, uno studente programmatore di nome Linus Torvalds ebbe l'idea di creare un kernel per un sistema Unix-like, rilasciandolo con una licenza libera, la GPL.
L'avvento di GNU/Linux ebbe conseguenze incredibili, quello che sarebbe nato negli anni seguenti è divenuto con molta probabilità il progetto comunitario più grande del mondo nella storia dell'umanità.

GPL è l'acronimo di General Public License.
La GPL è stata scritta da Stallman per definire e far rispettare i diritti del software libero e degli autori dei programmi che l'avrebbero usata.
Fu fondata la Free Software Foundation, che negli anni avvenire sarebbe stata la fondazione incaricata di pubblicare la GPL e di gestirne le eventuali evoluzioni e modifiche.

GPLv1

La GPL garantiva i diritti fondamentali delineati sulla filosofia delle quattro libertà annunciate da Stallman.
Permetteva dunque di copiare, redistribuire e modificare il codice in piena libertà, con la garanzia che ogni lavoro derivato, comprese le modifiche e le migliorie, venissero rilasciate con la stessa licenza.
La GPL alla prima versione richiedeva anche che il codice fosse sempre rilasciato o reso disponibile alla comunità (non si poteva rilasciare il solo binario).
Una nota importante era anche che la GPL non poteva essere accompagnata da una licenza meno libera (casi di dual-license).

GPLv2

Con il passare del tempo venne l'esigenza di fare alcune modifiche alla licenza.
Nella versione due della GPL fu infatti introdotta una clausola che, in poche parole, non permetteva ad alcune persone di rilasciare codice sotto la licenza GPL.
Queste persone erano tutte quelle che, per motivi anche non da loro causati, non potevano far godere la comunità dei diritti di libertà descritti nella licenza stessa (leggi dello stato, regole aziendali).

GPLv3

Questa serie di clausole però non si rivelarono suffiecienti a garantire la libertà che Stallman desiderava.
Nel 2005 si ritenne necessario aggiungere altre clausole:

- la clausola anti tivoizzazione
- la clausola per impedire gli accordi in stile Microsoft-Novell
- la clausola Affero (dalla AGPL) per introdurre la GPL al livello del SaaS (Software As A Service)

Con l'uscita della GPLv2, venne creata anche una licenza (più che altro strategica), per una migliore convivenza tra software libero e software proprietario.
Questa licenza venne presentata come "Library Generic Public License" e permetteva ad i software non liberi di sfruttare parti software dei software non proprietari (era indirizzata alle librerie e simili).
Il numero iniziale della licenza era 2 (per indicare che derivava in modo diretto ed era allo stesso passo della GPLv2), successivamente venne rinominata in "Lesser Generic Public Lincense", senza però avanzare il numero maggiore di licenza che diventò 2.1.

Esistono molte licenze libere, due degne di nota sono la licenza del MIT (Massachusetts Insitute of Technology) e la BSD (Berkeley Software Distribution).
Le clausole in comune sono poche:

- libertà di modifica, redistribuzione e vendita
- libertà di essere utilizzate all'interno di software proprietari
- obbligo di riapplicare la licenza in lavori successivi e di mostrare sempre la licenza all'utente finale

L'unica grande differenza è che nella BSD-license è proibito, per chiunque sfrutti il codice, farsi pubblicità con il nome degli autori del software.
La cosa strana è che per entrambi le licenze, esistono versioni modificate che rendono compatile l'una con l'altra.
Da notare è anche il fatto che molto codice sotto queste licenze, è stato e viene utilizzato da aziende di punta (i.e. Microsoft per lo stack TCP/IP ha utilizzato per anni la BSD-license).

Nel corso degli anni molte grandi aziende o gruppi hanno proposto la propria licenza tanto che ad oggi se ne contano decine.
Tra le tante compatibili con la GPLv2 troviamo:

- Artistic License 2.0
- BSD license (modified version)
- CeCILL
- GPL / GNU General Public License
- ISC licence
- License of Perl
- MIT license
- W3C Software Notice and License
- X11 license
- Zope Public License
- Berkeley Database License
- BDL / BSD Documentation License
- Cryptix General License
- Intel Open Source License
- LGPL / GNU Lesser GPL
- License of Python
- Public Domain
- WTFPL
- zlib/libpng license

E tra le non compatibili troviamo:

- Academic Free License (AFL)
- Affero General Public License
- Apache License
- Apple Public Source License (APSL)
- BSD license (original version)
- Common Public License
- CDDL
- Creative Commons licenses
- Eclipse Public License
- European Union Public Licence
- Hacktivismo Enhanced-Source License
- IBM Public License
- LaTeX Project Public License
- Microsoft Public License
- Microsoft Reciprocal License
- Mozilla Public License (MPL)
- Netscape Public License (NPL)
- Open Software License
- OpenSSL license
- PHP License
- Q Public License (QPL)
- Software Libre para Uso Civil (SLUC)
- Sun Standards License (SISSL)
- Sun Public License
- XFree86 1.1 License