Il pinguino che voleva volare

Sapere, Software e Sostenibilità per una Società Libera

Giuseppe Augiero
Nicolino Curalli
Jacopo Nespolo
GULP - Pisa

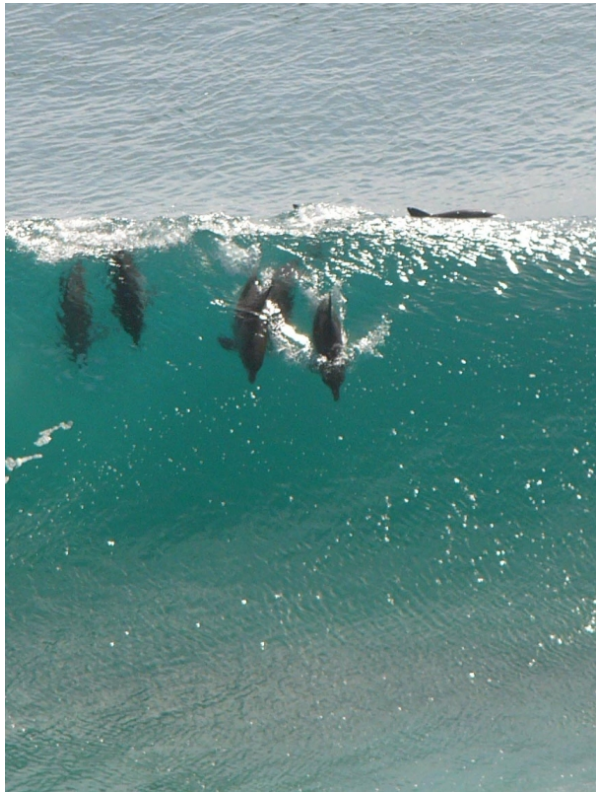
La Limonaia,
16 febbraio 2011

Indice

Sapere

Una terza dimensione...

Il potere di una scelta.



Lego, ergo sum

> *"Io sono quando scelgo e, se non sono, non scelgo".*
> *Karl Jaspers [?]*

La *Libertà* è poter scegliere.

Se venissimo limitati nella nostra capacità di scegliere, automaticamente non saremmo più liberi.

Imparando dai contrari

Talvolta si conosce il vero significato delle parole soltanto quando si esaminano i contrari [**?**].

Non libertà significa essere costretti a fare qualcosa controvoglia, ma anche non poter fare qualcosa che si desiderava.

Libertà nei saperi

- Libertà nei saperi è *accesso alle informazioni*.
- L'accesso include il potere di *modificare e condividere* le informazioni.
- Tutti hanno diritto ad arricchirsi culturalmente.
- Ostacolare l'accesso alle informazioni è di fatto violare il diritto al progresso culturale dell'individuo e della società.

Libertà nei saperi: perché?

> *"Nanos gigantium humeris insidentes".*
> *("Nani che si ergono sulle spalle di giganti").*
> *Bernard de Chartres [?]*

- La libertà delle informazioni è essenziale in una società moderna.
- Lo sviluppo sociale, scientifico, tecnologico ed economico si basa sul progressivo miglioramento di conoscenze precedenti.

Esempio: Pubblicazioni scientifiche

- La scienza si basa sullo scambio delle informazioni sotto forma di articoli.
- Ciascun articolo si basa sulle ricerche precedenti, che vengono citate per riconoscere i meriti dei rispettivi autori.

> *"Se copi da un autore, è plagio.*
> *Se copi da due, è ricerca."*
> *Wilson Mizner[**?**]*

- La ricerca si basa sullo scambio degli articoli tra università, ricercatori e studenti.
- Talvolta il costo per accedere agli articoli è eccessivo per i ricercatori. Libertà?

Esempio: Libri

- Quando acquistiamo un libro, diventiamo proprietari del supporto su cui è stampato il testo, ma non del suo contenuto.
- Siamo liberi di prestare/regalare il libro a un amico (anche in parte :-).
- Il contenuto appartiene al detentore dei *diritti d'autore* dell'opera.
- Questi può essere l'autore, ma spesso è la casa editrice.

Achtung!!

- Le case editrici delle riviste scientifiche acquisiscono i diritti d'autore. Talvolta gli autori devono pagare delle commissioni per poter pubblicare.
- Gli autori si vedono spesso usurpati della paternità delle proprie opere.

Libertà delle informazioni nell'era digitale

- Il concetto stesso di libertà si deve adattare ai mezzi di comunicazione digitale.
- Il "consumo" di opere dell'intelletto e delle informazioni avvengono a ritmi estremamente elevati.
- L'opera digitale deve essere tecnicamente **copiata** per ogni uso.
- Le leggi sul diritto d'autore pre-esistenti non sono quasi mai adatte a trattare informazioni digitali.

Proprietà dei saperi

- In tutti i Paesi, azioni legislative hanno introdotto strumenti quali il *brevetto* e il *diritto d'autore* (o *copyright*).
- Il loro scopo originario era di fornire temporaneamente alcuni diritti esclusivi di sfruttamento economico di un'opera, incentivandone così la produzione e lo sviluppo[**?**].
- La durata dei termini si è andata allungando progressivamente. Per alcune opere, la protezione dura per 80 anni dopo la morte dell'autore!

Quest'idea è mia

- È giusto che l'autore di un'opera sia riconosciuto e ricompensato.
- È necessario che le idee possano circolare liberamente, per favorire lo sviluppo sociale ed economico.

Bisogna trovare il giusto equilibrio tra protezione dell'individuo e protezione degli interessi della società.

I metodi attuali di protezione della *proprietà intellettuale* non sono al passo coi tempi, e stanno diventando un ostacolo allo sviluppo sociale e economico, volto a salvaguardare i privilegi di pochi.

Open Access. Un'alternativa?

- Già da alcuni anni, alcune riviste scientifiche offrono la possibilità di pubblicare un articolo in *Open Access*.
- Un articolo in Open Access è fruibile a tutti gratuitamente. Talvolta (Gold OA) vengono anche ceduti i diritti di modifica dell'articolo.

Open Access. Pro e contro

CONTRO

- Spesso gli editori esigono un pagamento da parte degli autori per la pubblicazione in OA.
- Sottrazione di proventi agli editori. Possibile insostenibilità delle riviste in formato cartaceo.

PRO

- Le pubblicazioni OA hanno un vantaggio in termini di impatto.
- Ricercatori e studenti (quindi le biblioteche) non devono pagare le laute spese di sottoscrizione alle riviste.
- Anche i paesi in via di sviluppo possono accedere agli articoli OA.

Come proteggerci

Cosa possiamo fare per proteggerci?

Quando compriamo: esigiamo le nostre libertà

- Ogni qual volta sia possibile, favoriamo pubblicazioni Open Access e archivi di pre- o post-print.
- Favoriamo informazioni digitali prive di protezioni, e che garantiscono le nostre libertà di usufruirne come meglio crediamo. La maggrior parte dei prodotti (musica, ebook, ecc.) è anche disponibile senza DRM presso venditori indipendenti.

arXiv.org

Quando produciamo: proteggiamo i nostri lavori

- Assicuriamoci che i nostri lavori siano fruibili a tutti, nel rispetto della nostra paternità sull'opera. Utilizziamo in maniera furba il diritto d'autore.
- Nel caso di pubblicazioni scientifiche, pubblicare l'articolo in un archivio di pre-print. Prediligere editori legati alle società scientifiche, il cui scopo non è solitamente legato a interessi economici.

creative commons

Software

In principio

- Nei primi anni '70 **Richard Stallman** entra a far parte del laboratorio di Intelligenza Artificiale del MIT.

- In quel periodo il software proprietario non esisteva e gli **hacker** si facevano da sè il software che utilizzavano per il loro lavoro.

Il Software

- La condivisione del software è una cosa vecchia quanto i computer, proprio come condividere le ricette è antico come il cucinare.

- *Non chiamavamo il nostro* **Software Libero**, *poichè questa espressione ancora non esisteva, ma si trattava proprio di questo.* ***[?]***

Il mercato della ferraglia

- Il computer che utilizzavano si chiamava **PDP-10** ed era prodotto da una azienda, la Digital.

- Il sistema operativo che faceva funzionare il computer, chiamato **ITS**, non era prodotto da una azienda ma veniva sviluppato dagli stessi informatici del laboratorio.

L'importanza del codice

- *Quando persone di altre università o di qualche società volevano convertire il nostro programma per il proprio sistema e utilizzarlo, erano le benvenute. [**?**]*

- *Se notavamo qualcuno usare un programma sconosciuto e interessante, si poteva sempre* **chiedere di vederne il codice sorgente** *in modo da poterlo leggere e modificare. [**?**]*

La fine di un'era

- La situazione cambiò drasticamente all'inizio degli anni '80 quando la Digital smise di produrre la serie PDP-10.
- Questo significò che quasi tutti i programmi che formavano ITS divennero obsoleti.
- Quando il laboratorio, nel 1982, acquistò un nuovo PDP-10, i sistemisti decisero di utilizzare il **sistema non libero** della Digital anzichè ITS.

I Pirati

- I moderni elaboratori di quell'epoca avevano il proprio sistema operativo, ma nessuno di questi era libero.
- I sistemi operativi prodotti dalle aziende erano venduti senza il codice sorgente quindi non era più possibile per gli hacker leggerli, modificarli e migliorarli.
- Una comunità cooperante era vietata.
- La regola era: **"Se condividi il software col tuo vicino sei un pirata, se vuoi modifiche, pregaci di farle (e pagaci profumatamente)."** [?]

Il bivio

- *Mi trovai di fronte ad una* **difficile scelta morale** *.[?]*
- *La scelta facile sarebbe stata quella di unirsi al mondo del software proprietario.[?]*
- *Un'altra possibile scelta, semplice ma spiacevole, sarebbe stata quella di abbandonare l'informatica.[?]*
- *In tal modo le mie capacità sarebbero state sprecate.[?]*

Ribellione

- *Allora cercai un modo in cui un programmatore potesse fare qualcosa di buono. [?]*
- *Mi chiesi:* **“Posso scrivere nuovi programmi, per rendere ancora possibile l’esistenza di una comunità?** *[?]*

La soluzione

- *La risposta era semplice: innanzitutto occorreva un* **Sistema Operativo**. *Questo è difatti il software fondamentale. [?]*

- *Con un sistema operativo si possono fare molte cose; senza, non è possibile far funzionare il computer.*

GNU

- Nacque così, nel 1983, il progetto **GNU**.
- Il nome **GNU** fu scelto secondo una tradizione hacker, come acronimo ricorsivo che significa: **GNU's Not Unix** – (GNU non è Unix)

Ambizioni

- L'idea era ambiziosa, il progetto GNU consisteva nello sviluppo di un sistema operativo alternativo a quelli proprietari e di tutta una serie di applicazioni utili.

- Questo sistema sarebbe stato compatibile con **UNIX** ma, a differenza di UNIX, **era libero**.

Libertà

- Cosa si intende per software libero?
- Richard Stallman riconobbe **quattro libertà fondamentali** applicabili al software.

Prima libertà

Libertà d'uso

- Libertà di eseguire il programma per qualsiasi scopo.

Seconda libertà

Libertà di modifica

- Libertà di studiare il programma e modificarlo.

Terza libertà

Libertà di distribuzione delle copie

- Libertà di ridistribuire copie del programma

Quarta libertà

Accesso al codice sorgente

- Libertà di migliorare il programma e di distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio.

Permesso d'autore

- *Avevamo bisogno di evitare che il software GNU fosse trasformato in software proprietario. [?]*
- *Il metodo che usammo fu il* **Permesso d'autore**. *[?]*
- *L'idea del permesso d'autore consiste nel dare a chiunque il permesso di eseguire, copiare, modificare e distribuire le versioni modificate, ma senza dare il permesso di aggiungere restrizioni.[?]*
- *In tal modo, le libertà essenziali che definiscono il software libero sono garantite a chiunque ne abbia una copia, e diventano* **diritti inalienabili**.*[?]*

Copyleft

- Il permesso d'autore (**copyleft**) usa le leggi sul diritto d'autore (**copyright**), ma le capovolge per ottenere lo scopo opposto: invece che un metodo per privatizzare il software, diventa infatti un mezzo per mantenerlo libero.
- Una particolare implementazione del permesso d'autore che utilizziamo per la maggior parte del software GNU e' la **GNU General Public Licence** (licenza pubblica generica GNU), abbreviata in **GPL**.

Hurd

- Nel 1990 il sistema GNU era quasi completo, l'unica parte significativa ancora mancante era il kernel.
- Il kernel del sistema GNU fu battezzato con il nome di **HURD**.

Linux

- GNU Hurd non era pronto per un uso non sperimentale.
- Nel 1991 **Linus Torvalds** sviluppò un kernel compatibile con Unix e lo chiamò **Linux**.
- La combinazione di Linux con il sistema GNU produsse un **sistema operativo libero completo**.
- *Chiamammo* **GNU/Linux** *questa versione del sistema, per indicare la sua composizione come una combinazione del sistema GNU col kernel Linux. [?]*

Open Source

- Secondo la filosofia del software libero **l'Open Source è una questione di libertà e non di prezzo** .
- Per parlare di OpenSource iniziamo a ragionare su quello che permette di fare una bicicletta.
- Quali legami hanno una bicicletta e il concetto di libertà del software?

Con una bicicletta posso:

- Utilizzarla liberamente per spostarmi.
- Ripararla quando si rompe.
- Smontarla e rimontarla per capire come funziona.
- Cambiare dei componenti per adattarla alle mie necessità o per potenziarla.
- Mostrarla a chiunque.

Bicicletta libera

- Con la bicicletta posso fare qualsiasi cosa mi venga in mente e che sia consona al suo utilizzo.
- E' fondamentale notare che chi mi ha venduto la bicicletta non ha il potere di decidere cosa posso o cosa non posso fare con essa.
- Nel momento in cui ho acquistato la bicicletta non ho più nessun vincolo nei confronti del venditore.

Il software che usiamo

- Attualmente buona parte del software che l'utente medio utilizza **non gode di libertà simili** a quelle che prima esponevamo per la nostra bicicletta.

"Ruota di Bicicletta",
di Marcel Duchamp,
Museo Nazionale di Arte Moderna, Roma.

Le non libertà

- Non è possibile utilizzarlo liberamente per qualsiasi scopo richiesto dall'utente.
- Non è consentito studiare come funziona il codice e modificarlo per adattarlo alle proprie necessità.
- Non è permesso copiare il software a qualcun altro che ne ha bisogno.
- E' vietato migliorare il programma utilizzato e rendere pubbliche le modifiche.

Forti limitazioni

- Colui che ha scritto il software ha il diritto di **porre vincoli** a chi lo ha acquistato.
- L' ideatore del software può decidere come può essere utilizzato o se e in che modo può essere redistribuito.
- I software non liberi non permettono all' utente di accedere al codice sorgente.

Bicicletta vs Software

- Tra la bicicletta ed il software vi è, dal punto di vista giuridico, una sostanziale differenza.
- La bicicletta ha un proprietario il quale ne dispone a suo piacimento.
- **Il software non è un oggetto fisico**, quindi non ha un proprietario.
- Il software è un'opera di ingegno e come tale risponde alle leggi del **diritti d'autore**.

Scelta

- Utilizzare software libero è una scelta etica.
- Lo sviluppo si basa sugli stessi principi fondanti della comunità scientifica:
  - Libero scambio delle informazioni.
  - Condivisione di idee e risultati.
  - Libero utilizzo della conoscenza.

Sostenibilità

Un esempio pratico

- Le grandi imprese del software, quando rilasciano un prodotto, tendono a coprire le esigenze, in termine di funzionalità sviluppate nel software, della maggior parte dei clienti potenziali:
  - empiricamente si è misurata la percentuale del 80% (fonte IDC).
  - il 20% dei clienti finali, nel caso in cui abbiano bisogno di una determinata funzione devono adattare le loro esigenze al programma.

Domanda:

L'adattamento del software alle esigenze del cliente in quale misura può essere realizzato e con quali mezzi?

Una risposta poco convincente

Il cliente può rivolgersi:

- Ai creatori del software:
  - la risposta sarà sicuramente funzione del suo potere economico nei confronti del produttore.
- Alle piccole e medie *software house* in grado di ritagliare, come un artigiano, un prodotto software sulle esigenze specifiche del cliente nei limiti imposti dalla struttura del software.

La realtà de facto:

- Sviluppi che utilizzano funzionalità già presenti nel software proposto.
- Servizio di installazione, assistenza e formazione.

C'è qualcosa di sbagliato!!!

Indipendentemente dal tipo di azione emergono due fatti essenziali:

- Le software house guadagnano dai servizi aggiuntivi (installazione, personalizzazione, formazione).
  - Sul software che forniscono non hanno però alcuna possibilità di modifica.
- Il cliente finale spende soldi per l' acquisto di software fornito con una licenza che non permette di fare nessuna modifica per adattare il prodotto ai suoi bisogni ed ottimizzarlo.

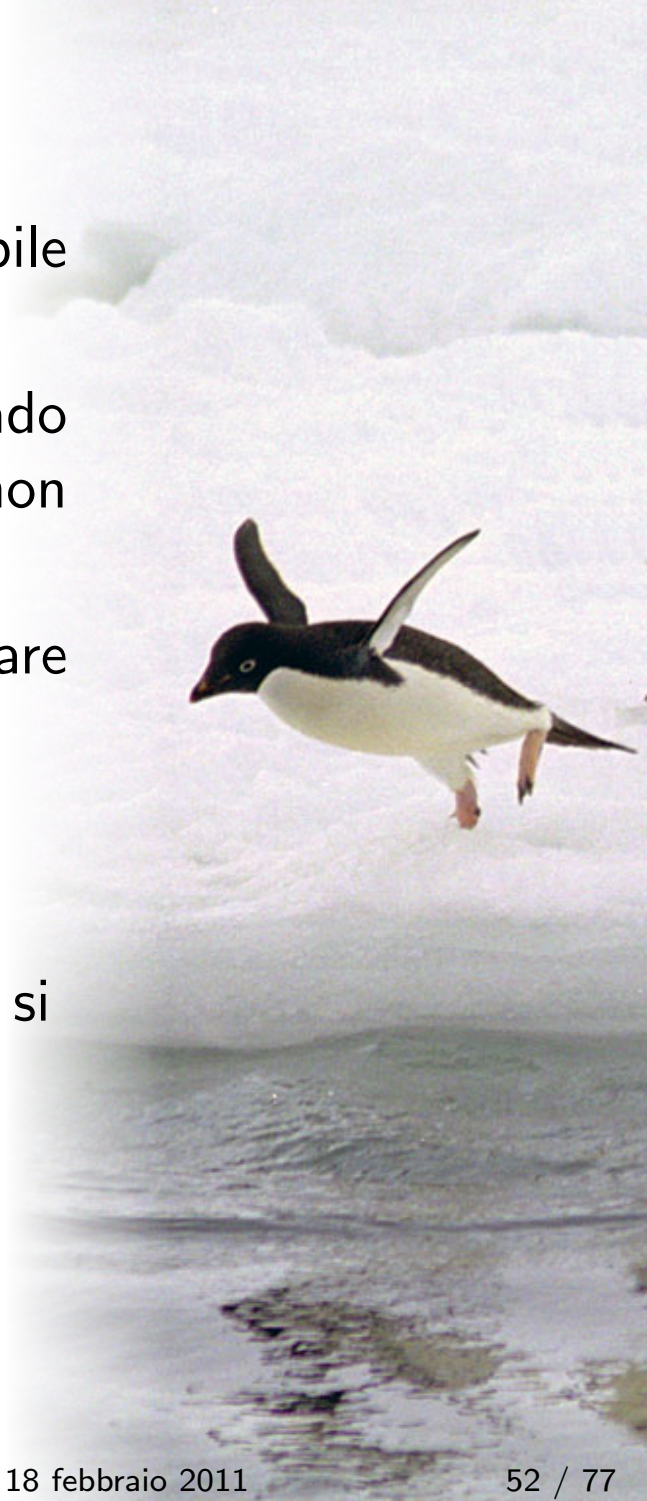
La conclusione realistica:

Il rivenditore ed il suo cliente quindi sono in balia delle scelte della casa produttrice del software e da essa dipendono per le scelte future.

La natura del software come bene

Il software è per sua natura un bene non escludibile e non rivale :

- non escludibile perché il produttore è in grado di impedire di usare il bene a coloro che non ne hanno pagato il prezzo.
- non rivale, in quanto più utenti possono usare contemporaneamente lo stesso software.



Osservazione

L'efficienza economica vorrebbe che i beni non rivali fossero distribuiti gratuitamente, o meglio si paghi solo per il supporto materiale e le attività strettamente collegate.

Conclusione:

Non è conveniente produrre il bene software.

La nascita di un dilemma

Dilemma:

- Se il prezzo di un bene (in questo caso un programma software) è più alto del costo di produrne una copia in più, qualcuno, che lo userebbe, non lo farà perché non è conveniente.
- Se il prezzo non è più alto del costo di produzione, il produttore non potrà coprire i costi necessari per sviluppare il software.

Effetto Collaterale:

Non vi saranno incentivi economici per l'attività di innovazione.

Le soluzione artificiosa al dilemma

Una vecchia proposta:

- Necessità di introdurre i brevetti.
  - L'esistenza dei brevetti dovrebbe stimolare la produzione di nuove invenzioni, che diventeranno poi di pubblico dominio allo scadere del brevetto.

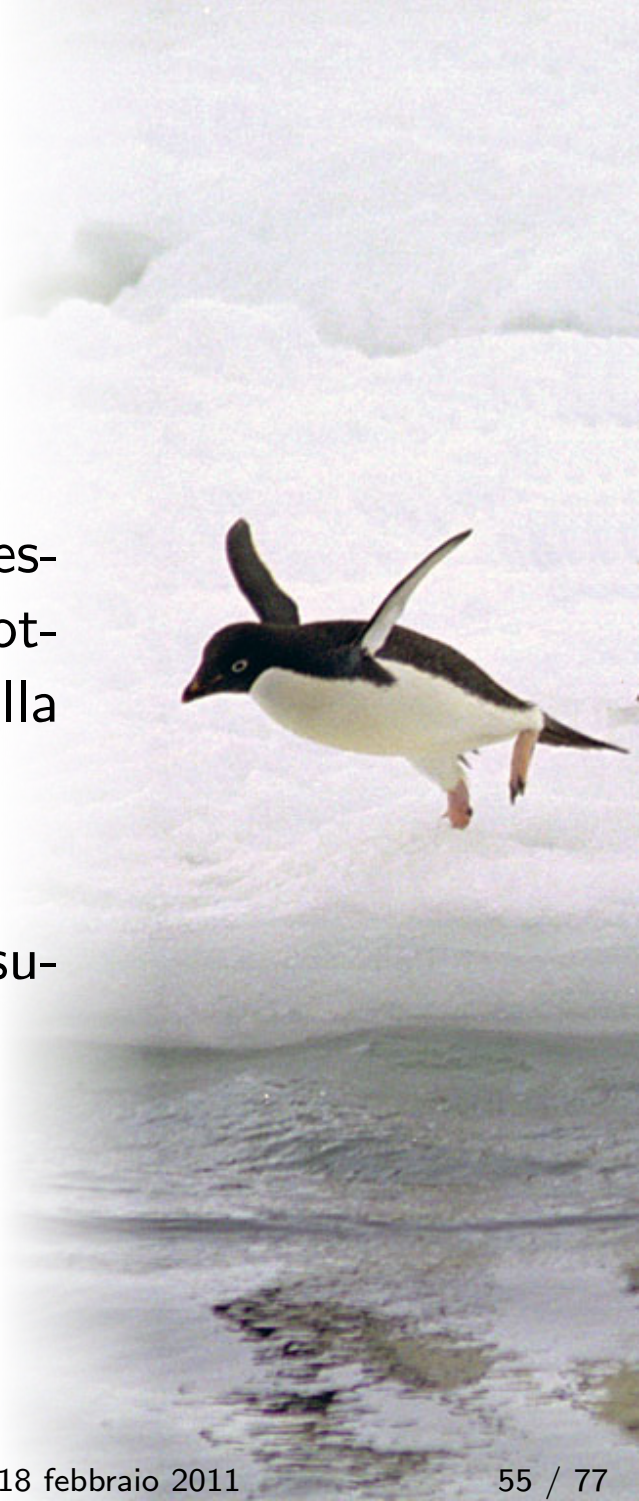
Conseguenze reali

- Limitazione della concorrenza tra le imprese, senza la quale i teoremi dell'economia del benessere non valgono.
- Siamo quindi in presenza di un fallimento di mercato.

Ancora peggio:

Il mercato del software proprietario gode di rendimenti di scala crescenti che orientano il mercato verso monopoli perché:

- Nel caso del software, in cui i prodotti sono essenzialmente valore aggiunto congelato, si ottengono rendimenti crescenti al crescere della produzione.
- Effetti di network.
- Effetti di apprendimento da parte dei consumatori.



Ripensando al valore del software...

Quanto vale o quanto costa esattamente un prodotto software?

Il software come tutte le altre classi di beni strumentali ha due tipi di valore economico:

- un valore d'uso, ovvero il suo valore economico in quanto strumento (bene intermedio).
- un valore di vendita, in quanto articolo commerciabile (bene finito).

Uno spaccato di realtà

Un circolo vizioso molto diffuso

Le grandi aziende tradizionali di sviluppo software mantengono una strategia di marketing tendente ad avere prezzi di acquisto molto alti e contributi per l'assistenza molto bassi o quasi nulli:

- i profitti provengono dalla vendita.
- l'assistenza è vista come un costo non come un ricavo.

Conclusione:

- Il valore d'uso e il valore di vendita del software proprietario non sono sinergici.
- Nasce un circolo vizioso: il paradigma del software proprietario diventa non sostenibile.

Il punto di svolta

Un analisi onesta è ammettere che il costo di acquisto di un software non si riduce al solo costo di licenza. Esiste un un indicatore utile:

- TCO : la somma di tutte le spese ed i costi associati all'acquisto e all'uso di equipaggiamenti, materiali e servizi.

Un cambiamento di fronte:

La misura del valore di un software tramite il TCO.

- pone al centro del calcolo dei costi le attività e non i prodotti.
- retribuisce ogni attività in maniera sostenibile, anche quella del cliente.

Il TCO di un prodotto software

Nel caso di un prodotto software per ottenere il TCO bisogna tenere conto delle seguenti attività:

- La fase di pianificazione e di valutazione della soluzione software più appropriata e conveniente per la propria azienda.
- La valutazione degli impatti sugli altri sistemi.
- La valutazione degli impatti sull'efficienza operativa.
- Il training per gli utenti presenti in azienda.
- La valutazione dei costi dei servizi accessori, che comprendono la manutenzione evolutiva del programma, gli aggiornamenti del software, il servizio di supporto.
- La valutazione dei costi incidentali.

L'approccio Open Source

- Il Software Libero ha il suo modello di sviluppo: il progetto Open Source.
- Il Software Libero e il progetto Open Source non obbediscono alle leggi del mercato.
- La struttura dei costi/ricavi di una azienda fondata sul software libero o Open Source non può essere tradizionale.

La struttura dei costi/ricavi di una azienda Open Source

- La soglia di ingresso nel mercato è bassa.
- Il costo di produzione è basso in quanto il bene è reperibile liberamente.
- L'offerta sul mercato non sono i beni ma i servizi.
- La reale struttura dei costi del ciclo vitale di un progetto è sostenuta tramite una struttura di prezzi per l'offerta di determinati servizi.
- Scambio di un valore continuo fra produttore e consumatore tramite il paradigma servizio/costo.
- Il valore di vendita viene a mancare.

Riflessioni

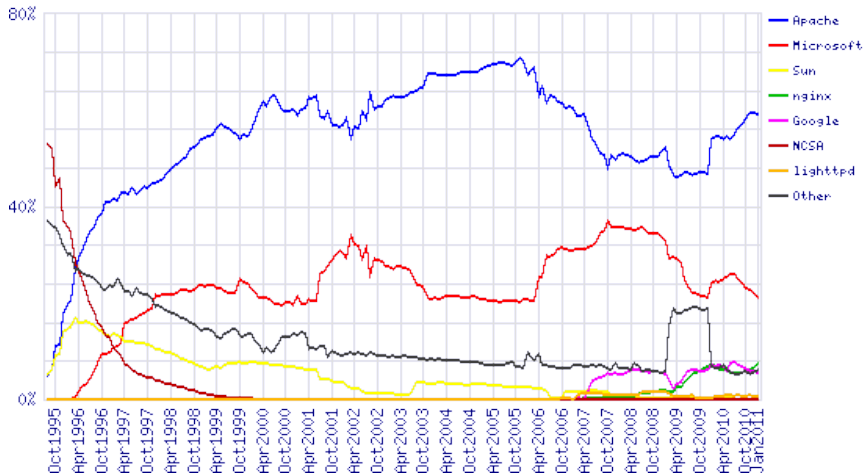
Abbiamo appena detto che il solo valore d'uso sia sufficiente per finanziare lo sviluppo dell'Open Source in modo sostenibile.

- La storia del web server Apache generalizza un modello in cui gli utenti di software trovano vantaggioso sostenere lo sviluppo dell'Open Source.
  - Il risultato è un prodotto migliore di come potrebbe essere altrimenti, a un costo inferiore.

Riflessioni

Abbiamo appena detto che il solo valore d'uso sia sufficiente per finanziare lo sviluppo dell'Open Source in modo sostenibile.

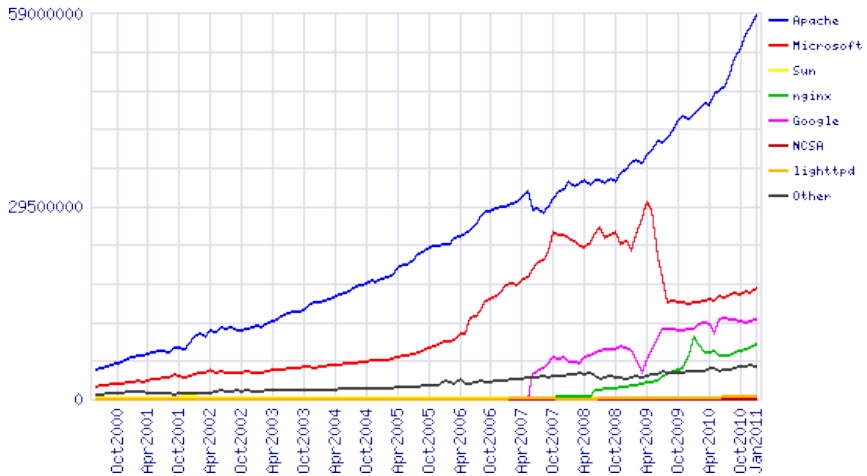
- La storia del web server Apache generalizza un modello in cui gli utenti di software trovano vantaggioso sostenere lo sviluppo dell'Open Source.
  - Il risultato è un prodotto migliore di come potrebbe essere altrimenti, a un costo inferiore.



Riflessioni

Abbiamo appena detto che il solo valore d'uso sia sufficiente per finanziare lo sviluppo dell'Open Source in modo sostenibile.

- La storia del web server Apache generalizza un modello in cui gli utenti di software trovano vantaggioso sostenere lo sviluppo dell'Open Source.
  - Il risultato è un prodotto migliore di come potrebbe essere altrimenti, a un costo inferiore.



Considerazioni:

L'Open Source serve non tanto ad abbassare i costi, quanto a ripartire il rischio perché:

- La disponibilità del codice sorgente.
- La presenza di una comunità che agisce in collaborazione, finanziata da più fonti di reddito indipendenti.

Fornisce una garanzia di sopravvivenza che ha un valore economico intrinseco o, comunque, un valore sufficiente per finanziarla.

Ancora:

L'Open Source rende piuttosto difficile trarre un valore di vendita diretto dai software.

- Non si tratta di una difficoltà tecnica:
  - La difficoltà sta piuttosto nella natura del contratto sociale che si trova alla base dello sviluppo Open Source.
  - Consentire di percepire un valore di vendita diretto rende impossibile legalmente la biforcazione dei progetti.
  - la possibilità di biforcare un progetto Open Source rende possibile tutte le evoluzioni possibili di un software.

Obiettivo

Nessuno si trovi in una posizione privilegiata per fare profitti.

L'Open Source moltiplica le occasioni

L'Open Source permette di costruire mercati nei servizi legati al software. E' possibile in questo modo:

- Creare un valore di vendita indiretto.
- Migliorare il software Open Source in quanto tale.
- Generare conoscenza e sviluppo .

Un elenco non esaustivo:

- Vendita di servizi.
- Perdita della leadership.
- Liberare il software, vendere il marchio
- Liberare il software, vendere il contenuto
- Vendita di documentazione di alto valore professionale

I vantaggi dell' Open Source

Alle imprese che comprendono i fattori economici, culturali e volendo organizzativi che risiedono in una effettiva strategia Open Source, questo modello offre buone aspettative:

- Evoluzione del prodotto.
- Aumento della qualità e della affidabilità del prodotto.
- Mantenimento del prodotto.
- Reclutamento e fidelizzazione dei collaboratori.
- Facilità a creare e proporre standard liberi.

Quando l'opensource ha vinto

Le discriminanti che fanno propendere per l'Open Source sono:

- l'esigenza di affidabilità, stabilità e misurabilità.
- la correttezza del design e dell'implementazione non possono essere verificate efficacemente se non tramite la revisione reciproca e indipendente.
- il software è di vitale importanza per il controllo dell'impresa da parte dell'utente.
- il software istituisce o abilita una comune infrastruttura di calcolo e comunicazioni.
- i metodi di punti o i loro equivalenti funzionali fanno parte delle comuni conoscenze ingegneristiche.

Ancora...

Un progetto Open Source costituisce

- un Valore Aggiunto di natura positiva che aiuta lo sviluppo economico di tutta la società.

Inoltre:

- La capacità di soddisfare i bisogni di informazione reciproca.
- La velocità con cui avvengono lo scambio di informazioni.

Costituisce un sistema aperto verso l' esterno in cui il contributo di ognuno può crescere e migliorare.

Una buona notizia

Una frase emblematica:

L'Open Source permette ad ognuno di dare un mattone per ricevere in cambio una casa.

Un progetto Open Source è un bene pubblico e può essere affetto da free riding:

- i singoli individui sono tentati di pagare il prezzo di un bene, scaricandolo su qualcun'altro.

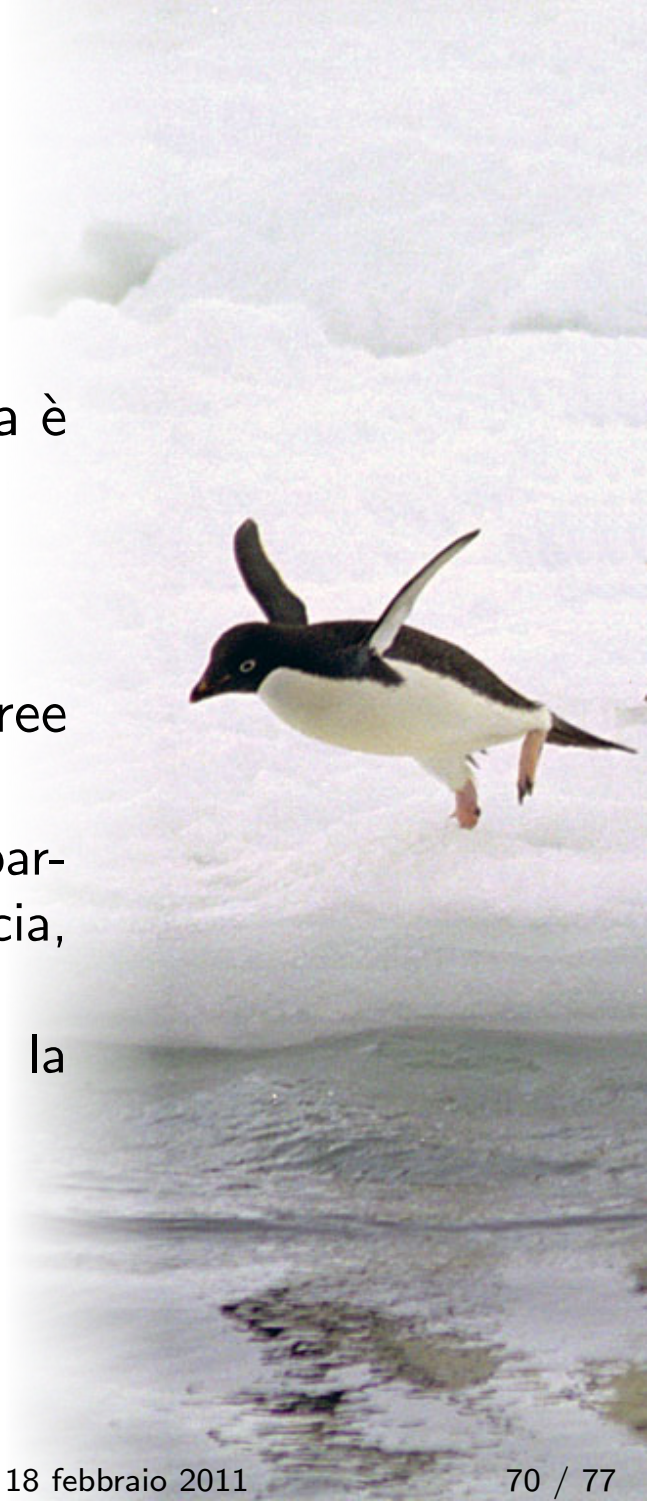
Una buona notizia (II)

Il caso reale:

La realtà empirica prova che la linea di tendenza è opposta al modello del *free riding*.

Il motivo sostanziale:

- l'utilizzo del software, anche se affetto da free riding, non ne diminuisce il valore
  - aumenta la possibilità che arrivino nuovi partecipanti (informazione, competenza, fiducia, reputazione. . .)
  - incrementa la possibilità di migliorare la qualità del software



Un'idea:

Creare o collaborare ad un progetto Opens ource è una bella maniera di imparare.

- Studenti e Insegnanti possono contribuire in vari modi allo sviluppo dell'Open Source:
  - Creando esempi di utilizzo e tutorial.
  - Scrivendo documentazione.
  - Partecipando alle mailing list.
  - Testando il software e comunicare i bug.
  - Scrivendo codice.
  - Pensando nuovi tipi di software.
  - Creando nuovi progetti.

Conclusione:

Cose è più simile all'insegnamento dell'imparare facendo.

Il percorso:

- Selezionare un progetto utile.
- Conoscere le persone che lo usano e lo sviluppano.
- Contribuire in qualche maniera.
- Motivare gli studenti.

Risultato:

L'Open Source è un ottima opportunità:

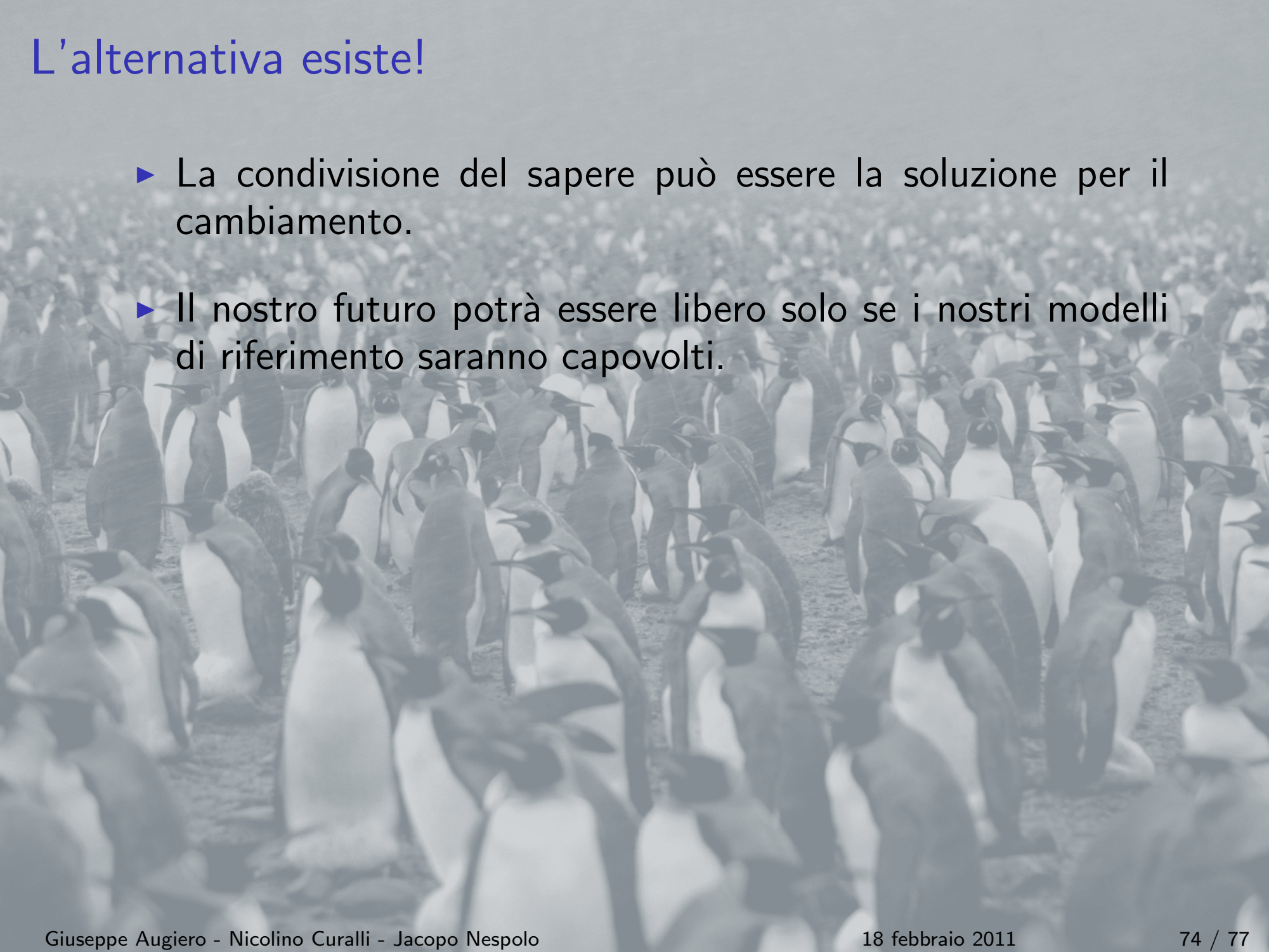
- per gli insegnanti.
- per gli studenti.
- per le scuole.

se tutti sono coinvolti attivamente.


Un'alternativa esiste?

L'alternativa esiste!

- La condivisione del sapere può essere la soluzione per il cambiamento.
- Il nostro futuro potrà essere libero solo se i nostri modelli di riferimento saranno capovolti.



Una alternativa è già realtà

Linux e il mondo dell'Open Source sono:

- ritorno alla **Creatività.**
- una **Comunità.**
- una **Rivoluzione.**

Tante ne sono possibili

"Se tu hai una mela, e io ho una mela,
e ce le scambiamo,
allora tu ed io abbiamo sempre una mela per uno.

Ma se tu hai un'idea, ed io ho un'idea,
e ce le scambiamo,
allora abbiamo entrambi due idee."

George Bernard Shaw

Il pinguino che voleva volare

Sapere, Software e Sostenibilità per una Società Libera

impaginato con LaTeX!

CC BY NC SA

Giuseppe Augiero
giuseppe@augiero.it
Nicolino Curalli
n.curalli@elabor.biz
Jacopo Nespolo
j.nespolo@gmail.com
GULP - Pisa

La Limonaia,
16 febbraio 2011