Laboratorio di Economia e Management
Scuola di Studi Superiori Sant'Anna

Piazza dei Martiri della Libertà 33- I-56127 PISA (Italy)
Tel. +39-050-883-341 Fax +39-050-883-344
Email: lem@sssup.it Web Page: http://www.sssup.it/~LEM/

LEM
Working Papers

L'economia degli standard e la diffusione delle tecnologie.

L'Open Source non è un assurdo economico

Andrea Bonaccorsi[*]
Cristina Rossi[†]

[*] *Scuola Superiore S.Anna, Pisa, Italy*
[†] *Scuola Superiore S.Anna, Pisa, Italy*

I2001/02 **Maggio 2002**

L’economia degli standard e la diffusione delle tecnologie.

L’Open Source non è un assurdo economico

Andrea Bonaccorsi
Cristina Rossi

Scuola Superiore Sant'Anna
Piazza Martiri della Libertà, 33
56127 Pisa, Italy

Introduzione

Come ha osservato Raymond (1999, 224), uno dei padri fondatori del movimento Open Source, “Linus Torvalds e i suoi compagni [hanno fornito] centinaia di megabyte di programmi, documenti ed altre risorse, persino intere suite di strumenti Internet, programmi di desktop publishing, supporto grafico, editor, giochi e così via”. D’altro lato, “nel mercato dei Web Server il 20% di Microsoft sembra sparire se paragonato al 53% di Apache” (Di Bona, Ockman, Stone, 1999, 9)

Dal punto di vista della teoria economica, il fenomeno dell’Open Source sembra sfidare alcune idee consolidate. L’affermazione dell’Open Source costituisce un’importante innovazione nel processo di produzione del software in contrapposizione al tradizionale modello proprietario, proprio del mondo commerciale. Da qui il sorgere di una serie di domande non banali, idealmente collocabili lungo l’asse delle tre fasi schumpeteriane del processo innovativo dove una nuova idea nasce (invenzione), trova applicazione economica trasformandosi in un nuovo prodotto o processo (innovazione vera e propria) ed è poi introdotta tra consumatori ed imprese (diffusione).

Per ciascuna delle tre fasi il fenomeno dell’Open Source genera puzzle teorici assai interessanti. Possibili linee di ricerca sono le seguenti:

– Perché i programmatori scrivono codice Open Source se nessuno li paga per farlo?

- Come è possibile che centinaia e centinaia di soggetti sparsi in tutto il mondo riescano a coordinarsi, al punto di produrre programmi composti da milioni di righe di codice[1], in assenza di una struttura gerarchica fondata sulla proprietà?
- Perché i programmi Open Source si diffondono in un mondo dominato dagli standard proprietari?

La prima domanda ha per oggetto la fase dell'*invenzione*: è certamente comprensibile che un singolo soggetto abbia inventato il concetto di *free software* e lo abbia proposto alla comunità dei programmatori, ma resta da spiegare il fenomeno della continua invenzione, da parte di centinaia di individui, di programmi offerti gratuitamente. Qual è il sistema di incentivi che regola la fase iniziale dell'innovazione? Quali sono le motivazioni che spingono gli agenti a comportarsi in questo modo?

La seconda domanda mette a fuoco il problema del *coordinamento*, che è centrale per trasformare un'invenzione in innovazioni economicamente vantaggiose e commercializzabili. Il funzionamento spontaneo e decentrato della comunità Open Source costituisce una sfida a molte nozioni di coordinamento correnti. Come è possibile allineare gli incentivi di molti individui senza ricorrere sistematicamente a diritti di proprietà e contratti di subordinazione? Come è possibile specificare i compiti di ciascuno senza ricorrere alla organizzazione gerarchica?

La terza domanda, infine, si concentra sulle condizioni per la *diffusione* dell'Open Source. Cosa determina la diffusione della nuova tecnologia in presenza di uno standard già affermato, e quindi di potenziali rendimenti crescenti di adozione della tecnologia precedente? Gran parte della teoria della diffusione di tecnologie di rete prevede l'emergenza di standard dominanti e, in seconda istanza, la coesistenza di più standard se la dinamica di diffusione parte nello stesso momento per le tecnologie in competizione. Nel caso dell'Open Source, al contrario, si ha diffusione di uno standard alternativo mentre esiste *già* uno standard dominante.

L'obiettivo del presente lavoro è di spiegare quelli che, a prima vista, sembrano un "non senso", utilizzando in modo originale alcuni strumenti forniti dalla scienza economica, quali la teoria dell'azione collettiva e la teoria della diffusione di tecnologie in presenza di esternalità di network. Si mostrerà come una lettura non convenzionale del fenomeno

[1] Secondo J. Sanders (1998), nel 1998 il kernel del sistema operativo Linux era composto da più di un milione e mezzo di righe di codice. Nella prima versione, implementata da Linux Torvalds nel 1991, le linee erano 10.000.

può dare una spiegazione plausibile. Allo stesso tempo si mostrerà come il software aperto è destinato non solo a convivere con il software proprietario, ma anche a originare una serie articolata di nuove forme di organizzazione economica (contratti, licenze, tipologie di imprese) nelle quali elementi open e proprietari si combineranno variamente.

1. Software Open Source e motivazioni individuali

Nel 1984 con la Free Software Foundation e l'elaborazione della General Public Licence, Stallman propone una idea innovativa, che sarà poi ribadita nel 1997 nella Open Source Definition[2]: chiunque deve avere la possibilità di utilizzare, studiare, modificare ed integrare il codice dei programmi per elaboratore ridistribuendo poi liberamente il risultato della sua attività.

Secondo i teorici di un nuovo modello di sviluppo del software, definito *bazar* (Raymond, 1999), i contributi di migliaia di programmatori che lavorano in modo totalmente volontario, scambiandosi idee e file attraverso la rete Internet, giungono in assenza di autorità centrale a realizzare programmi complessi. Gli sviluppatori, sparsi in tutto il mondo, lavorano ai vari progetti in modo asincrono, spesso sacrificando il tempo libero e i fine settimana: la struttura top down, gerarchicamente organizzata e pianificata, condivisa dalla quasi totalità dei processi produttivi, è qui abbandonata a favore di un modello bottom up, non coercitivo ed ampiamente decentralizzato, anche se sottoposto a regole condivise di comportamento.

Ed è proprio per questo che il movimento Open Source sembra essere un concorrente temibile per le imprese che dominano il mercato del software. Queste, infatti, si trovano a fronteggiare una minaccia che proviene dall'esterno: non si tratta di un nuovo concorrente che fa le cose nello stesso modo, anche se più velocemente o in modo più efficiente, ma di un nuovo paradigma, nettamente contrapposto a quello tradizionale. Nel paradigma proprietario gli sviluppatori, retribuiti e concentrati nello stesso luogo, lavorano contemporaneamente alla scrittura di un codice che sarà protetto da una licenza commerciale che ne vieterà la distribuzione se non in forma binaria e a fronte di

[2] Open Source Definition, http://www.opensource.org

forti limitazioni nell'utilizzo, che ne impediranno tra l'altro la duplicazione e ogni tipo di modifica.

Fornendo ai programmatori la possibilità di lavorare liberamente sul codice sorgente, la filosofia alla base del movimento Open Source facilita sia la correzione degli errori sia l'adattamento ad esigenze e piattaforme hardware diverse. Se a ciò si aggiunge l'assenza delle pressioni imposte alle software house dai loro stessi annunci di versioni più aggiornate ed efficienti non sorprende che il risultato possa essere un software qualitativamente migliore. L'affidabilità è, infatti, indicata tra le caratteristiche che fanno del software Open Source "an easy sell" (Liebovitch, 1999). La sua resistenza ai crash può essere misurata in mesi ed anni piuttosto che in giorni o settimane: il Web Server Apache equipaggiato con sistema operativo Linux, installato nel 1997 nei laboratori della Compaq Computer Corporation ha resistito per ben 267 giorni prima di "crashare" e lo ha fatto solo in seguito ad un temporale che ha privato di energia elettrica lo stabile dove esso era collocato. A ciò si affianca una notevole portabilità: "oggi Linux gira su qualunque cosa, dai PalmPilot alle workstation Alpha, ed è il sistema operativo più soggetto a porting fra quelli disponibili per PC" (Torvalds, 1999, 112)

Appare quindi del tutto naturale l'interrogativo proposto da Glass (1999, 104): "*Non capisco chi sono quei pazzi che vogliono scrivere, leggere e fare revisione di tutto quel codice senza ricevere nessuna ricompensa"*. Le motivazioni di chi scrive in Open Source, sin dall'inizio ampiamente analizzate dai padri fondatori del movimento, hanno cominciato a suscitare crescente interesse da parte della teoria economica a mano a mano che il fenomeno usciva dalle università e dai centri di ricerca, diveniva "tremendously successful" (Lerner e Tirole, 2001, 819) e creava nuovi e spesso fortunati modelli di business.

Negli scritti che hanno fatto di lui uno dei più famosi *Open Source evangelist*, Raymond ricollega gli incentivi di chi scrive codice open ai valori propri della *hacker culture.* Gli sviluppatori Open Source sarebbero gli eredi dei c.d. *Real Programmer* del secondo dopoguerra che "provenivano dai settori dell'ingegneria e della fisica…[e] programmavano in FORTRAN e in un'altra mezza dozzina di linguaggi ormai dimenticati" (Raymond, 1999). Formatisi nelle Università o nei centri di ricerca delle grandi società di software, l'Artificial Intelligence Lab del MIT di Boston, il

Laboratorio di Intelligenza Artificiale dell'Università di Stanford, il Dipartimento di Informatica di Berkeley, lo Xerox Parc, gli hacker vivono la programmazione come una forma d'arte provando una soddisfazione estetica nello scrivere codice, simile a quella che la musica dà ad un compositore e il dipingere ad un pittore.

A fronte di questo occorre tuttavia ridimensionare alcuni famosi miti sul software libero: un approfondito studio dei più fortunati progetti Open Source rivela come non sempre i programmatori lavorano gratuitamente e operano in un ambiente totalmente anarchico con lo scambio di file e gli user group sulla rete Internet quali uniche forme di coordinamento. Una survey condotta dall'Institut fuer Psychologie dell'Università di Kiel, in Germania, (Hermann, Hertel e Niedner, 2001) su un campione di 141 sviluppatori del kernel del sistema operativo Linux, evidenzia come ben il 43% riceva una qualche retribuzione mentre il 20% percepisce un salario "for their Linux programming on a regular basis". Allo stesso modo nei grandi progetti è sempre possibile individuare un "well-respected leader" (Lerner e Tirole, 2001) o un "core development group" (Mockus et al., 2000) che hanno il compito di individuare i principali problemi da risolvere, scegliere tra le molteplici soluzioni proposte dagli sviluppatori e decidere la versione ufficiale del codice prodotto dal gruppo di lavoro. Da alcune parti è stato osservato (Bezrukov, 1999) come la struttura organizzativa del gruppo degli sviluppatori del kernel Linux sia gerarchicamente organizzata attorno alla figura carismatica di Linux Torvalds che agisce come una sorta di "dittatore benevolente". Allo stesso modo non sempre i programmatori sono migliaia: mentre la Linux kernel mailing list conta oltre 3500 iscritti i contributori effettivi non raggiungono i 500.

Tutto ciò, tuttavia, non ridimensiona la necessità di comprendere a fondo la struttura degli incentivi di chi lavora a progetti Open Source. Un'attenta analisi evidenzia come sia possibile ricondurre il problema al classico calcolo costi-benefici, qualora se ne evidenzino chiaramente gli elementi costitutivi.

Sul lato dei costi, si osserva come spesso l'investimento in infrastrutture necessario alla partecipazione al movimento è quasi nullo. La maggior parte dei programmatori possiede un computer per ragioni di studio o di lavoro o addirittura scrive codice Open Source sul luogo di lavoro. La sopra citata survey sugli sviluppatori del kernel Linux individua un 38% di soggetti che scrivono codice free durante il regolare orario di

lavoro anche se questo non fa ufficialmente parte delle loro mansioni. Allo stesso modo il collegamento alla rete Internet non costituisce per i più un problema. La presenza di contributi provenienti da tutto il mondo è poco più che un'idea romantica: la maggior parte delle LOC aggiunte ai progetti è dovuta a soggetti europei e statunitensi che hanno quindi ampia disponibilità di connettività.

I principali investimenti hanno allora ad oggetto il tempo e le risorse intellettuali. A questo proposito gli studi condotti evidenziano come anche l'investimento in termini di tempo sia contenuto. In un' analisi su oltre 1700 partecipanti ad un importante newsgroup per la risoluzione dei problemi legati al server Apache, Lakani e von Hippel (2000) trovano che oltre l'ottanta per cento dei soggetti che rispondono alle richieste di aiuto impiega un tempo medio di 5 minuti per fornire la soluzione al quesito cui ha deciso di rispondere. La ragione di ciò è semplice: gli agenti conoscono già la risposta così come conoscono il funzionamento dei newsgroup, dei principali strumenti di posta elettronica, dei programmi che zippando i file ne riducono la dimensione permettendo di farli circolare agevolmente sulla rete. La stragrande maggioranza degli sviluppatori ha avuto modo di lavorare con Unix da cui derivano molti programmi Open Source e può quindi far fruttare agevolmente le proprie competenze.

Ai bassi costi si contrappone poi una serie non trascurabile di benefici.

In primo luogo la già citata *gratificazione intrinseca* della programmazione software come esperienza estetica ("fun to program"). La motivazione edonistica (o anche ludica) è comunemente annoverata dalla sociologia e dalla psicologia sociale tra le determinanti della partecipazione ad un qualsivoglia movimento sociale così come l'altruismo è riconosciuto essere alla base di molti comportamenti umani. In questo ambito può trovare spiegazione la cessione gratuita di beni e servizi, che secondo la antropologia della "gift economy" (Mauss, 1959) è, da un lato un modo per creare e mantenere relazioni sociali, dall'altro racchiude l'obbligo tacito alla reciprocità. Ciò rimane vero anche quando la cessione avviene non a favore di soggetti conosciuti ma verso una comunità di sconosciuti dai quali, tuttavia, si spera di avere in futuro aiuto e supporto sulla base di una di tacita convenzione di reciprocità instauratasi in forza delle contribuzioni precedenti.

In aggiunta si devono considerare due ulteriori benefici: *reputazione* e *autoproduzione*.

La possibilità di guadagnare una *reputazione* soprattutto attraverso la competizione con gli altri sviluppatori per lo sviluppo della più elegante ed efficiente soluzione informatica e quindi di segnalare il proprio talento è chiaramente interpretabile alla luce della teoria economica. In presenza di asimmetrie informative sulla qualità dei lavoratori, la disponibilità a programmare gratuitamente può essere interpretata come segnalazione del talento (Lerner e Tirole, 2001). La visibilità che un giovane e creativo programmatore può avere attraverso la partecipazione alla comunità open source è superiore a quella raggiungibile in una grande impresa di software.

Elementi di reputazione sono da sempre il motore della comunità scientifica. La forte gratificazione intellettuale che i programmatori derivano dallo scrivere codice è del tutto analoga a quella derivante da una scoperta scientifica. Il movimento Open Source ricalca la struttura degli incentivi propria della ricerca scientifica trasferendola alla produzione di tecnologie aventi un potenziale valore commerciale Il nuovo modello di sviluppo del codice diviene così "la naturale estensione della cultura scientifica occidentale" (Stone, 1999). Il processo di scoperta scientifica prevede la condivisione dei risultati, così come i dettami dell'Open Source prevedono la condivisione del codice sorgente. Condividere i risultati permette al ricercatore di migliorarli grazie ai feedback forniti dai colleghi, dai quali egli può ricevere riconoscimento per il proprio lavoro e quindi prestigio. Analogamente condividendo il codice sorgente, si ricevono dagli altri membri del gruppo suggerimenti che consentono di perfezionarlo. Il fatto che i risultati siano sotto gli occhi di tutti, conferisce, poi, un prestigio che è tanto maggiore quanto più la comunità è ampia. D'altro canto lavorare in progetti Open Source comporta una ricaduta di prestigio e visibilità che dà ai programmatori la possibilità di essere notati dalle aziende di software. Ciascuno dei padri fondatori del movimento "ha raggiunto una reputazione tale da permettergli di pagare l'affitto e mantenere dei figli" (Di Bona, Ockman, Stone, 1999, 14). Senza contare poi che molti degli sviluppatori sono studenti di computer science che utilizzano la partecipazione a progetti Open Source come base per la loro tesi di laurea o di dottorato acquisendo un'esperienza di programmazione che potranno utilizzare al loro ingresso nel mondo del lavoro.

Il nuovo paradigma di sviluppo racchiude in se un'immensa possibilità di *learning*: avere a disposizione il codice sorgente di un programma consente di studiarlo approfonditamente. Tra le motivazioni che spingono i partecipanti ai newsgroup

dedicati alla risoluzione di problemi di suite Open Source, quella dell'apprendimento derivante dalla lettura delle risposte alle domande poste dagli utenti è senza dubbio una delle principali (Lakhami e von Hippel, 2001).

Molti dei progetti Open Source, infine, sono nati come autoproduzione, per soddisfare una domanda in assenza dell'offerta corrispondente, in altre parole per "fill an unfilled market" (Green, 1999). Un tipico caso è rappresentato dal linguaggio Perl. Tale linguaggio fu elaborato da un amministratore di sistema della Burroughs, che aveva sentito la necessità di qualcosa che unisse alle caratteristiche di velocità e immediatezza proprie dei linguaggi di shell di Unix, la possibilità sviluppare programmi più complessi, tipica dei linguaggi di alto livello[3]. Non riuscendo a trovare questo "qualcosa" l'aveva creato distribuendolo poi a tutta la comunità Open Source. Oggi Perl è un linguaggio comunemente accettato e ad esso si devono la maggior parte dei contenuti interattivi della pagine Web

A quelle descritte si aggiungono una serie di motivazioni ancillari che vanno dal sospetto con cui viene valutata l'opera di "*costumerizzazione del computer*" (Ullman, 1998) operata da Microsoft, alla *instrumentability* (Hertel, 2002) ossia all'idea che il proprio contributo sia fondamentale per la riuscita di un progetto, al grande valore attribuito agli obiettivi del progetto stesso.

Anche inquadrando la struttura degli incentivi in una semplice analisi costi benefici, restano i problemi legati al free rider e all'esternalità positiva che sono chiaramente individuabili nella produzione dell'informazione. Rendendo il codice sorgente accessibile a tutti, il movimento Open Source ha dato al bene software le caratteristiche proprie dei beni collettivi (Bessen, 2001). Di tali beni, una volta che sono forniti, possono liberamente beneficiare anche coloro che non hanno contribuito alla produzione o lo hanno fatto in misura inferiore al necessario. Per ciascun individuo, quindi, la strategia di risposta ottima è quella di non contribuire per niente sfruttando poi la contribuzione altrui per avere benefici senza sostenere alcun costo. Chiaramente se il suddetto ragionamento è condiviso da tutti, il bene non sarà fornito: è il problema del *free riding*, ciò che è vantaggioso dal punto di vista individuale porta a risultati collettivamente subottimali.

[3] Si pensi ad esempio al linguaggio C.

Sorretto da un'ineccepibile costruzione teorica, il *free riding* totale (ossia con tutti i soggetti che scelgono una contribuzione nulla) si osserva assai raramente, come è noto, sia nel mondo reale sia negli esperimenti di laboratorio in cui si richiede ai soggetti di contribuire alla fornitura di beni collettivi (si veda tra gli altri Fox, Guyer e Hamburger, 1975).

Nel caso specifico, tuttavia, vi è da spiegare come siano evitati comportamenti autointeressati di free riding e si mantenga una cooperazione sostenuta nel tempo anche in comunità di grandi dimensioni.

Una possibile spiegazione sta nel considerare il software prodotto al di fuori dei canali commerciali come una semplice istanza del più generale *problema dell'azione collettiva* alla luce del quale rileggere tutte le motivazioni precedentemente analizzate. Le dimensioni assunte dal fenomeno risultano assai interessanti dal punto di vista teorico, perché sembrano sfatare la nota tesi di Olson (1965) di una relazione inversa tra ampiezza del gruppo e probabilità che un'azione collettiva sia condotta con successo. L'argomento è che all'aumento della dimensione del gruppo diminuisce la probabilità che un comportamento di defezione possa essere individuato con corrispondente aumento dell'incentivo a godere del bene senza contribuire a produrlo.

In realtà, l'analisi di Olson non tiene conto di un insieme di variabili il cui effetto è, invece, determinante e come tale non può essere trascurato, prima tra tutte l'eterogeneità dei soggetti partecipanti all'azione collettiva. Hardin (1982), autore di uno dei testi più autorevoli sull'argomento, sottolinea la centralità della *eterogeneità* di risorse ed interesse: un piccolo insieme di soggetti molto motivati e dotati delle risorse necessarie è in grado di fornire il bene anche senza la cooperazione altrui, costituendo il c.d. *sottogruppo efficace*. Per il movimento Open Source si potrebbe pensare che il sottogruppo efficace teorizzato da Hardin, la cui presenza è essenziale per la riuscita dell'azione collettiva, sia composto dagli hacker. Le loro "maggiori risorse" sono rappresentate dall'incontestabile *know how* posseduto nel campo dell'informatica, mentre il più accentuato interesse deriva sia dalla soddisfazione intellettuale insita nella scrittura del codice sia dalle necessità di lavoro o ricerca accademica sia dalla volontà di guadagnarsi una reputazione e segnalare il proprio talento.

L'azione di fornitura è resa, inoltre, più efficace dal fatto che il bene codice sorgente è non rivale nel consumo. I benefici da esso apportati sono, infatti, invarianti rispetto al

numero dei suoi utilizzatori: chiunque può scaricare il sorgente dalla rete, compilarlo e utilizzarlo senza che questo riduca la possibilità di farlo per altri soggetti. Questo contribuisce a confutare la tesi di Olson: Chamberlain (1974) dimostra che, nel caso di beni collettivi non rivali, l'ammontare fornito cresce al crescere della dimensione del gruppo.

Le conclusioni di Hardin danno conto di come il software Open Source sia prodotto dagli hacker "più interessati", ma non di come alla sua produzione collaborino anche altri soggetti, realizzando una cooperazione che si sta facendo sempre più estesa. A questo proposito si può allora fare riferimento alla teoria della Massa Critica, proposta inizialmente da Schelling (1978), sviluppata in sociologia da Marwell, Oliver e Prahal (1993) e ripresa in economia da Witt (1997) e Huberman (1998). La presenza dei soggetti molto interessati fa sì che si superi la fase iniziale del processo di fornitura del bene collettivo, quella in cui per la maggior parte dei soggetti il costo della contribuzione eccede il beneficio. Superata tale fase, sempre più soggetti trovano vantaggioso contribuire. Si innesca così un circolo virtuoso che fa sì che il fenomeno si autosostenga procedendo verso il nuovo equilibrio dove tutti i membri del collettivo scelgono la cooperazione.

In quest'ottica il ruolo dei soggetti molto interessati non è quello di fornire interamente il bene, ma di creare le condizione necessarie perché la sua produzione diventi più agevole. Il loro compito è quello "essere i primi a cooperare", favorendo il superamento della fase di *start up* dell'Azione Collettiva.

Questo schema interpretativo sembra spiegare bene la dinamica dei progetti Open Source nati con un piccolo gruppo di soggetti che lavorano per raggiungere un particolare obiettivo. Se il gruppo riesce a trovare soluzioni valide in questa fase iniziale, pubblica i propri risultati su Internet pubblicizzando su mailing lists e newsgroup l'esistenza del progetto. A questo punto il progetto può imboccare due strade: attrarre un numero crescente di contributori o non destare alcun interesse. Nel primo caso si avrà un progetto Open Source ben strutturato, si pensi al caso di Linux, nel secondo il progetto andrà a morire secondo lo schema del "dying seminar" descritto da Schelling (1978).

2. *Il problema del "non sexy work" e l'emergenza di modelli di business ibridi*

Lo schema fin qui presentato può essere sufficiente per spiegare le attività di sviluppo che presentano caratteri intrinseci di gratificazione assimilabili alla produzione artistica e scientifica e effetti di segnalazione del talento, ma non è assolutamente convincente per gran parte delle attività di programmazione c.d. "non sexy", quali le interfacce grafiche, la redazione dei manuali tecnici, l'attività di supporto nei newsgroup. Nonostante su quest'ultimo punto si siano raccolti dati empirici (Lakhani, von Hippel, 2001) che evidenziano motivazioni non dissimili da quelle di chi risolve complessi problemi di programmazione, il problema della motivazione a svolgere attività di sviluppo a basso valore tecnologico all'interno di comunità nelle quali i valori della creatività, della produzione artistica e della reputazione sono fondanti è uno dei più difficili per l'intero paradigma. È appena il caso di menzionare che queste attività, mentre hanno un basso contenuto innovativo, sono viceversa fondamentali ai fini dell'adozione del software da parte degli utenti. Si pensi ad esempio alle due più famose interfacce grafiche per Linux: KDE e Gnome, ideate nel 1998, hanno messo a disposizione degli utenti Linux l'ambiente desktop mouse driver che da qualche tempo si è imposto come standard nel mondo commerciale favorendo sia l'adozione del sistema operativo di Torvalds da parte dei soggetti meno *skilled* sia il passaggio ad esso di parte degli utenti Window e Mac.

L'analisi economica della diffusione ha chiarito da tempo, infatti, che, mentre le discontinuità tecnologiche sono responsabili dell'apertura di nuove traiettorie, la diffusione nel sistema economico dipende da una serie cumulata di modeste innovazioni incrementali, in molti casi di nessun interesse tecnologico intrinseco.

Questo problema sembra essere risolto dalla nascita di nuovi modelli di business "ibridi". Esiste, infatti, complementarità tra lo sviluppo di programmi da parte della comunità degli sviluppatori Open Source e la nascita di nuove imprese che producono software libero, lo cedono gratuitamente ai clienti e poi spostano il valore dalla licenza al service facendo pagare servizi aggiuntivi di packaging, consulenza, manutenzione, aggiornamento e training. Questo modello di business è largamente accettato in tutta la comunità di sviluppatori e utenti, attraverso una regolazione minuziosa, non priva di sviluppi giuridici innovativi, del rapporto tra prestazioni gratuite e prestazioni a

pagamento. Moltissime imprese di successo già affermate sul mercato adottano varianti di questo modello di business. Tra queste Red Hat non solo ha resistito all'esplosione della bolla speculativa che nella primavera del 2000 ha cancellato molte imprese tecnologiche, ma continua a fare profitti. La missione aziendale di quest'impresa, che ha fornito software per le missioni spaziali della Nasa, è paragonata a quella di un costruttore di auto. Così come nell'industria dell'auto i vari pezzi prodotti da diverse industrie sono assemblati per creare un prodotto che si adatti alle esigenze del consumatore, Red Hat riunisce e compila il codice sorgente del sistema operativo Linux ottenendo un eseguibile pronto per l'installazione che distribuisce assieme ad applicazioni accessorie (una tra tutte il programma Gimp per la gestione e la modifica delle immagini) e una ricca documentazione in un CD che ha un prezzo inferiore ai cento dollari. A Red Hat si stanno recentemente affiancando sia nuovi soggetti che forniscono servizi di supporto per software Open Source sia i giganti dell'industria del software che, preso atto che il futuro vedrà l'inevitabile coesistenza dei due paradigmi, decidono di rendere compatibili i loro prodotti con le piattaforme Open Source, di rilasciare il codice sorgente di alcuni software e, addirittura di partecipare a progetti Open Source. Così se l'Apple rilascia il codice sorgente del Mac Server OS X, Sun ha prodotto in Open Source la Suite StarOffice per opporsi allo strapotere di MS Office di Microsoft e nel dicembre 2000 IBM ha annunciato di voler investire un miliardo di dollari nello sviluppo di Linux per promuovere la diffusione delle sue piattaforme di e-business.

I descritti modelli di business ibridi, che sono assolutamente essenziali alla diffusione dell'Open Source, sono resi possibili anche dalla proliferazione delle licenze di cessione del software che è possibile far rientrare nello schema dell'Open Source. Rispettati, infatti, i criteri fondamentali dell'Open Source, ossia il pieno accesso al codice sorgente, la piena usabilità del software da esso originato, la libertà di modificarlo e ridistribuire i risultati delle modifiche, chiunque è libero di elaborare una propria licenza OSS. Accanto alla più famosa (e più estrema) GPL sotto la quale sono rilasciati Linux e oltre l'ottanta del cento della produzione Open Source, è così stato elaborato un vasto insieme di licenze alcune delle quali, come la BSD (Berkeley Software Development), permettono l'appropriazione delle modifiche effettuate su codice Open Source. Altre licenze poi, pur non essendo Open Source perché proteggono il diritto d'autore

dell'impresa produttrice, prevedono la pubblicazione del codice sorgente e lo rendono utilizzabile da determinate categorie di soggetti. E' il caso della Apple Public Source Licence di Apple e della Sun Comunity Source Licence di Sun.

Oltre che per il ruolo giocato nella diffusione, l'importanza dei modelli ibridi sta nel fatto che essi risolvono simultaneamente due problemi: da un lato rassicurano i potenziali adottanti circa l'ampia disponibilità di servizi complementari, che sono percepiti come decisivi per l'affidabilità e l'effettiva usabilità dei programmi software, dall'altro forniscono incentivi monetari per lo svolgimento delle attività di sviluppo "non-sexy", per le quali le motivazioni tipiche della comunità Open Source non sono sufficienti. Senza contare poi che l'esistenza di soggetti commerciali che impiegano software Open Source è una garanzia della futura sopravvivenza del software stesso. Un fattore chiave per la decisione di adozione è la sicurezza che il software continuerà ad essere prodotto e sviluppato evitando così agli utenti cambiamenti di applicativi e piattaforme che comportano alti costi di switching in termini di infrastrutture e training del capitale umano.

La coesistenza di modalità a pagamento e gratuite è una chiave di lettura della affermazione del movimento Open Source. Appartiene quindi alla stessa dinamica di diffusione del software libero la necessità di "ibridare" forme di remunerazione diverse all'interno di modelli di business innovativi. Sotto questo profilo, lo studio dettagliato delle licenze GPL e delle sue numerose varianti sopra citate, come pure dei contratti in uso presso le nuove imprese che sull'Open Source vivono e fanno profitti rappresenta un'interessante sfida per la ricerca economica.

3. *L'Open Source in cifre: la rilevanza empirica del nuovo paradigma di produzione del software*

La dimensione del fenomeno Open Source può essere valutata sia osservando la quantità e la natura del software prodotto sia la sua diffusione nel sistema economico e sociale. Per quanto riguarda questa seconda dimensione, un ruolo particolare è rivestito dall'adozione da parte della pubblica amministrazione. Il settore pubblico, infatti, possiede il peso necessario a far sì che i nuovi standard liberi possano imporsi in sostituzione di quelli proprietari attualmente dominanti.

Studi condotti nel 2001 (Ghosh, Prakash, 2001; Schimitz 2001) segnalano la presenza di circa 12.000 progetti Open Source attivi. Considerato che ad ogni progetto lavorano in media venti programmatori e che si stima che ciascuno di essi contribuisca in media a due progetti si ottiene una comunità di sviluppatori composta di circa 120.000 unità che hanno prodotto oltre 1000 megabyte di codice per un totale di circa 25 milioni di linee. Il numero di progetti posti su Internet è in continua crescita: a fronte di molti progetti che sono destinati a morire, ve ne sono altri che raccolgono un crescente successo. Tra i più citati (oltre al sistema operativo Linux e al web server Apache) vi sono BIND, il programma che opera la risoluzione del DNS e Sendmail per lo scambio di posta elettronica. Questi due software hanno rappresentato nel loro mercato le cosiddette *killer application*: nessun concorrente commerciale è riuscito (o ha voluto) ad intaccare la loro indiscussa supremazia.

Caratteristica peculiare della produzione in Open Source è l'elevata concentrazione dei contributi, si calcola, infatti, che al primo 10% degli autori sia dovuto oltre il 70% del codice totale con ben il 20% fornito dai primi dieci autori, principalmente università e centri di ricerca. A ciò fanno riscontro grosse differenze nella produttività degli autori stessi. Mentre solo 25 autori partecipano a più di 25 progetti, la stragrande maggioranza ne segue uno solo. La medesima struttura concentrata si ritrova all'interno dei singoli progetti. Secondo Mock e al. (2000) ai 15 più prolifici sviluppatori di Apache si deve l'aggiunta di quasi il 90% delle linee di codice mentre uno studio condotto sul progetto GNOME (Koch, Schneider, 2002) evidenzia la presenza di 301 programmatori, a 52 dei quali si deve l'ottanta per cento del codice. La stessa evidenza si ritrova nello svolgimento del "non sexy work": degli oltre 7.000 soggetti che hanno fornito help on line per il Web server Apache, solo 100 hanno fornito il 50% delle risposte.

Passando poi ad esaminare la diffusione dell'Open Source software, degna di nota è la crescita di Linux la cui quota di mercato aumentava nel 1998 di oltre il 200 per cento. Nel 1999 la SuSe, la più importante società tedesca di distribuzione del sistema operativo di Torvalds, incrementava i suoi profitti del 350%. L'esplosione della bolla speculativa nella primavera del 2000, ha rallentato ma non arrestato la tendenza: il numero di utenti Open Source continua a crescere.

Esistono, tuttavia, forti differenze tra il mercato dei server e quello dei client (con particolare riferimento al caso dei desktop) che sono evidenziabili sia a livello generale,

sia osservando l'adozione delle imprese e del settore pubblico. Mentre sul lato client la Microsoft ha potuto sfruttare la posizione dominante guadagnata con MS Dos e la famiglia dei Windows[4], ciò non si è verificato nel caso dei server Web: Netcraft calcola che attualmente circa il 61% degli Internet Web server impieghi Apache. Buona è anche la diffusione nel caso dei general-purpose server (LAN o Intranet aziendali). Alle stime di Gartner Group (2001) che attribuiscono a Linux il 10% dei server esistenti negli Stati Uniti si affiancano quelle di IDC (2001) che parlano, invece, del 27%[5]. Dal lato server si registra anche la maggior adozione da parte della PA europea che equipaggia con software Open Source l'8% dei suoi server a fronte dell'1% dei suoi client (Schmitz, 2001). Le punte maggiori si registrano in Germania e Francia dove intere organizzazioni sono passate all'Open Source[6]. Per quanto riguarda il mondo business Forrester (2001) con un'indagine condotta tra le 2.500 Top Company americane, trova che il 56% di esse usa qualche forma di Open Source, la maggior parte del quale riguarda i server web (33%). Sorprendente il dato relativo ai software per la gestione di database: il 42% è Open Source a fronte dell'1% dell'impiego nei normali computer da ufficio. Ottime le prospettive di sviluppo evidenziate: si calcola che, entro il 2004, il venti per cento degli attuali investimenti in licenze sarà spostato ai servizi e al training del personale.

Simile, anche se su scala minore, la situazione nelle piccole e medie imprese italiane. Una ricerca condotta per conto di SuSe Italia da Congenio s.r.l. e presentata a SMAU 2001, evidenzia come su 414 PMI industriali 16 utilizzino Linux come sistema operativo dal lato server mentre una soltanto lo fa sul lato client dominato da Windows con una quota che sfiora il 90%. E' interessante notare, tuttavia, che quasi l'ottanta per cento degli amministratori di sistema delle intervistate dichiara di conoscere Linux e il movimento Open Source in generale con punte che sfiorano il novanta per cento se ci si concentra sulle medie imprese. Tra i fattori che spiegano l'utilizzo di Linux sul lato server predominano la stabilità e la sicurezza mentre tra le motivazioni dell'utilizzo di Windows sul lato client la facilità d'uso è dominante. Oltre il 25% dei soggetti vede la

[4] Il sistema operativo Windows 3.1 che ha raccolto a pieno l'eredità di Lisa della Apple e di X Windows di Xerox Parc nello sviluppo di interfacce grafiche mouse driver, è apparso ad inizio anni Novanta le due interfacce grafiche per Linux, KDE e Gnome compaiono alla fine del medesimo decennio.

[5] Le prestazioni del software Open Source sono particolarmente buone del campo dei database server con MySql e PostgreSql.

[6] Il governo francese è particolarmente attivo nel promuovere la diffusione di software Open Source. Nel 1999 soluzioni Open Source erano adottate dal Ministero della Difesa, dell'Educazione, della Ricerca, delle Finanze.

possibilità di un mutamento di sistema operativo dal lato client e questo fornisce indicazioni circa la futura diffusione dell'Open Source in quest'ambito. In particolare la quasi totalità degli utilizzatori di Linux dal lato server è favorevole al suo impiego anche sul lato client. Del resto la piccola e media impresa sembra rappresentare il terreno di elezione per la diffusione del software libero: liberando le risorse prima deputate all'acquisto delle licenze, infatti, è possibile migliorare il parco macchine e il training del personale permettendo l'adozione di soluzioni ITC tecnologicamente avanzate anche in realtà di ridotte dimensioni che spesso possiedono un budget limitato.

4. *Il Problema del Coordinamento: Modularità del Software, Reusability e Ruolo della Rete Internet*

E' inevitabile che un economista, osservando i sorprendenti risultati raggiunti dal movimento Open Source si chieda: "how do coordinate the efforts of potentially hundreds of developers wordwide?" (Hecker, 1999). Questo a prima vista sembra impossibile, anche se in realtà accade: i progetti Open Source esistono e, i "relatively loosely coordinated software development products" (Hecker, 1999, 50) riescono a sfidare la concorrenza degli equivalenti commerciali. Raymond (1999, 224) osserva ironicamente che prima dell'esperimento Open Source "tutta la tradizione della progettazione del software era dominata dalla legge di Brooks secondo cui un progetto al quale abbiano contribuito migliaia di sviluppatori non può essere che un accozzaglia di codice instabile e traballante".

In cerca di una possibile spiegazione, alcuni protagonisti del movimento sono giunti a riprendere il concetto di "ordine spontaneo"(Perkins, 1999), paragonando il meccanismo di coordinamento spontaneo e decentrato a quello che sta alla base dell'incontro tra domanda e offerta nei mercati e al funzionamento della mano invisibile. Si tratta di una spiegazione che non può essere accettata in prima battuta, ma che è necessario arricchire con considerazioni che riguardano, da un lato, la tecnologia di produzione del software, dall'altro i nuovi strumenti di comunicazione resi possibili dalla rete Internet.

Per quanto riguarda il primo aspetto, Glass (1999) osserva come "object orientation seemed destined to make software development a matter of fastening Lego components

together". Non costituendo un'entità monolitica i programmi possono essere divisi in moduli da ricomporre e riutilizzare in vario modo. La quasi totalità degli sviluppatori del kernel Linux lavora in modo ordinato ai subsystem del kernel in cui i file hanno lo scopo di far dialogare il computer con le periferiche o di permettergli la lettura di file video o audio.

I concetti di *modularity* e *reusability* (Bonaccorsi, Rossetto, 1999) stanno quindi alla base dello sviluppo del software e sono resi ancora più effettivi dalla condivisione di un protocollo comune (linguaggio di programmazione) dove gli errori sono individuati e corretti attraverso il meccanismo della compilazione.

Un chiaro esempio delle suddette caratteristiche è l'utility *Grep* (acronimo di *Generalised Regular Expression Parser*), scritta in linguaggio C, che permette di individuare un file contenente un particolare testo: in sintesi una versione più sofisticata del *Find* di MS-DOS. *Grep* è una componente di Linux ma è stata scritta prima della sua nascita e può essere impiegata all'interno di altri sistemi operativi.

Secondo Linus Torvalds (1999), nella modularità sta appunto la chiave del successo dell'Open Source. Parlando di Linux egli afferma che "quello che ci serve è un sistema il più possibile modulare. Il modello di sviluppo Open Source lo vuole, per non avere gente che lavori simultaneamente sugli stessi punti" e la fonte della modularità di Linux è nel suo "kernel monolitico".

Il coordinamento è reso possibile quindi da una oculata scelta tecnica che in parole semplici è quella di "tenere il kernel piccolo e limitare al minimo il numero di interfacce e di altri vincoli al suo sviluppo futuro" in modo tale che i programmatori possono aggiungere moduli senza "pestarsi i piedi l'uno con l'altro" (Torvalds, 1999, 117-118).

Tuttavia anche queste premesse non sono a illuminare il problema del coordinamento, in quanto manca un tassello fondamentale: come avviene la comunicazione tra i programmatori? L'esplosione del movimento Open Source, che trova nel sistema operativo di Torvalds la sua massima espressione, ha coinciso, di fatto, con la sorprendente diffusione di Internet e con l'espansione dell'industria degli Internet Service Providers, che ha permesso la fornitura di connettività al grande pubblico a prezzi sempre più bassi. Attualmente la rete è il luogo virtuale dove i programmatori si incontrano rendendo effettivo il nuovo modello di sviluppo: "once an Open Source file is posted to the Web, talented programmers world-wide can download it and begin

tinkering" (Sanders, 1998, 80). In un certo senso il Word Wide Web è diventato il regno incontrastato del fenomeno Linux: si calcola che le pagine Web dedicate al sistema operativo di Torvalds sono "in the neighborhood of three millions"(Wallich, 1999, 44).
Siamo quindi di fronte ad un fenomeno di coordinamento che sfrutta profonde innovazioni intervenute:

- a livello dei singoli task, che sono resi più facilmente interfacciabili dalla evoluzione della tecnologia software;
- a livello della architettura, che è resa modulare in generale dall'evoluzione dei linguaggi di programmazione e in particolare dalle scelte strategiche di Linux sul kernel;
- a livello dei costi di comunicazione, che sono abbattuti e resi trascurabili dall'avvento di Internet.

Sotto queste condizioni il coordinamento gerarchico non è strettamente necessario. Al contrario, esso rischierebbe di deprimere le motivazioni intellettuali, estetiche e di divertimento che sembrano intrinseche alla comunità dei programmatori. Il coordinamento viene assicurato da forme di leadership basata sulle competenze di programmazione, che somigliano più alla legittimazione della comunità scientifica che alla catena di comando di una impresa.

5. Il problema della diffusione: software Open Source ed esternalità di rete

I meccanismi alla base della diffusione dell'Open Source sono ancora poco chiari persino ai suoi padri fondatori. Constatando che Linux, "è usato per il controllo dei sistemi robotizzati e ha volato a bordo dello Shuttle", Torvalds (1999) afferma: "ho avvertito sì la transizione dall'essere l'unico utente di Linux all'essercene un centinaio, ma quella da cento utenti ai milioni mi è sfuggita".
Poiché la vasta letteratura sui processi di diffusione delle tecnologie, coinvolge, oltre all'economia, discipline quali la sociologia o la geografia (Rogers, 1995), diviene importante circoscrivere l'ambito di analisi facendo riferimento alle caratteristiche del bene oggetto di diffusione e dei soggetti chiamati a domandarlo. Accettando la definizione di Varian e Shapiro (1999, 3), secondo cui "*è informazione tutto ciò che può essere digitalizzato, ovvero rappresentato come sequenza di bit*", il software è

senz'altro informazione. Dal punto di vista economico la produzione di informazione si caratterizza per avvenire con alti costi fissi e costi marginali trascurabili: tutta la massa del costo è concentrata nella prima copia i cui costi di riproduzione sono, invece, estremamente ridotti. Dal lato della produzione il software presenta forti economie di scala: quante più copie si producono e si vendono, tanto più l'alto costo della prima sarà assorbito dal bassissimo costo delle successive.

Dal lato della domanda, invece, si osserva il fenomeno dell'esternalità di network, nota anche come "demand-side economy of scale" (Bensen e Farrell, 1994). Questo tema ha dato vita ad un'immensa mole di contributi da quando, nel 1985, Katz e Shapiro ne hanno fornito la più esauriente definizione: "The utility that a user derives from consumption of the good increases with the number of other agents consuming the good". Accanto ad essa gli autori forniscono anche un'esaustiva classificazione delle fonti di esternalità: così l'esternalità può essere diretta o indiretta a seconda che derivi dall'appartenenza "fisica" dei soggetti ad una rete o da "market-mediated effects" (Farrell e Saloner, 1985) nel caso di beni complementari. L'esempio più citato per il caso di esternalità diretta è quello delle "reti di comunicazione" quali il fax o il telefono la cui l'utilità per i potenziali adottanti dipende in maniera cruciale dal numero di altre famiglie o imprese che fanno parte delle corrispondenti reti. Chiaramente vi è l'esternalità di network diretta anche quando i consumatori appartengono una rete che non è fisica, ma virtuale come nel caso degli utenti di uno stesso sistema operativo.

L'esternalità indiretta caratterizza, invece, il cosiddetto paradigma hardware-software in cui due o più beni complementari sono uniti a formare un sistema che per operare ha necessita dell'apporto di tutte le componenti. L'esternalità positiva deriva dal fatto che "un bene complementare diventa più economico e disponibile in maniera più veloce al crescere del mercato del bene con esso compatibile" (Farrell e Saloner, 1985, 71). E' questo il caso dei computer e dei programmi per elaboratore, del giradischi e dei dischi, delle videocassette e dei videoregistratori il cui mercato è stato teatro di una delle battaglie commerciali più discusse dalla teoria economica: quella tra il VHS di Matsushita e il Beta della Sony.

A queste due fonti principali se ne affianca una terza che si verifica per i beni durevoli quando "la qualità e la disponibilità dei servizi collegati al bene…dipende dal numero di unità del bene che sono state vendute"(Katz e Shapiro, 1985, 424): si pensi ai servizi di

manutenzione forniti dalle case automobilistiche, tanto più affidabili e facilmente reperibili quanto maggiore è il numero di auto di quella casa in circolazione.

Gli utenti dei programmi per elaboratore sono sottoposti principalmente al primo tipo di esternalità: l'esempio classico è quello dello scambio di file. Chi possiede lo stesso programma per la gestione di un dato tipo file può leggere e modificare i file ricevuti da altri soggetti che, a loro volta, potranno compiere le medesime operazioni. Da ciò deriva l'importanza di condividere un particolare tipo di software: il sistema operativo al quale è deputata la gestione del "file system" dell'elaboratore.

Le esternalità di network hanno rilevanti conseguenze sulla struttura del processo di diffusione che, nel caso particolare, corrisponde al problema dell'affermazione di uno standard. In particolare il processo diffusivo risulta esposto ad esiti di path dependence e di lock-in (Arthur, 1989, 1994, 1996). Situazioni in cui, cioè, un piccolo vantaggio iniziale a favore di uno standard è sufficiente perché esso si imponga, conquistando tutto il mercato che resta, in seguito, bloccato su di esso per la presenza di elevati costi di switching.

I concetti di lock-in e path-dependence sono stati recentemente oggetto di un acceso dibattito rispetto al quale. proprio la natura delle tecnologie di Open Source consente di formulare alcune precisazioni rilevanti.

Liebowitz e Margolis (1996), sottolineano come l'analisi degli standard da parte della teoria economica enfatizzi il rischio di incorrere nella "trappola" ben esemplificata dalle conclusioni di David (1985) riguardo alla battaglia Qwerty- Dvorak: "Se ci sono solo tastiere QWERTY, i dattilografi studieranno solo le tastiere QWERTY, ma se i dattilografi studiano solo QWERTY, ci saranno solo tastiere QWERTY". Per spiegare come questo si verifichi, David fa riferimento ai lavori di Arthur sull'operare dei rendimenti crescenti nel processo di diffusione di due tecnologie in competizione tra loro, per cui, più una tecnologia è adottata, più alta è la probabilità che lo sarà in futuro. I rendimenti crescenti sono immediata conseguenza delle esternalità di network, tanto che Witt sostiene che i due termini possono essere utilizzati come sinonimi (1997).

Data allora la presenza nel processo di diffusione del software delle esternalità di network dirette descritte in precedenza, il meccanismo del lock-in sembra essere un esito inevitabile: se un software riesce a guadagnare una fetta consistente di mercato, si innesca un circolo virtuoso per cui i consumatori avranno ancora più incentivo a

adottarlo, aumenterà l'offerta di prodotti complementari (applicativi, manutenzioni) e detto software dominerà il mercato divenendo lo standard.

In Arthur et al. (1983) il concetto è formalizzato con la teoria delle Urne di Polya. Lo schema è quello del campionamento con ripetizione da un'urna contenente palline rosse e blu, nella quale ogni volta che si estrae una pallina di un dato colore se ne aggiunge un'altra del medesimo colore. In ogni periodo t, la probabilità di aggiungere palline rosse è una funzione crescente della proporzione di palline rosse che è presente nell'urna, mentre lo stesso vale per le palline blu. Si dimostra che, quando il numero di palline tende all'infinito, la proporzione di palline dello stesso colore converge in probabilità ad uno. Spiegato il lock-in, resta però da spiegare cosa lo innesca: Arthur fa riferimento alla presenza di "small events" (1989, 116) che possono "by chance" dare ad una delle due tecnologie il necessario vantaggio. Se così stanno le cose, non esiste nessuna difesa dalla possibile affermazione di tecnologia inefficiente a scapito di un'altra che, se adottata, sarebbe sicuramente superiore. E' appunto quello che David sostiene essere accaduto per le macchine per scrivere QWERTY, di fatto inferiori per prestazioni alle di DSK di August Dvorak.

Lo sviluppo dell'Open Source contraddice questa previsione, in quanto esso guadagna terreno sullo standard dominante rappresentato da Microsoft facendo pensare che tra i due si arriverà ad un equilibrio in cui i due paradigmi convivono fianco a fianco. Occorre quindi cercare di spiegare come un nuovo standard possa affermarsi in presenza di un altro già consolidato e come due standard possano convivere. In un articolo del 1997, Witt osserva come la dimostrazione matematica del lock-in dipenda dalle assunzioni iniziali, mutando le quali l'autore riesce a dare conto di come "a newly introduced technology can successfully disseminate in a market despite existing network externality" (Witt, 1997, 753).

Le ipotesi di Arthur sono difficilmente difendibili nel caso del software. L'autore assume innanzitutto che le due tecnologie in competizione arrivino sul mercato nello stesso momento. Si tratta della "virgin market condition" (Witt, 1997, 762), certamente non sostenibile nel caso del software dove il sistema operativo Linux si trova a competere con il predominio decennale dei sistemi basati e derivati da MS-DOS. Allo stesso modo, non è mai vero che il mercato potenziale di una tecnologia è infinito ("the scope of the market for any product is limited" (Witt, 1997, 763)) e che le scelte della

sua adozione non sono rivedibili. Questo vale in particolare nel caso dei programmi per elaboratore che hanno un mercato in crescita ma necessariamente limitato (il loro utilizzo non è indispensabile in ogni attività umana!) e non prevedono investimenti in capitale fisso tali da determinare proibitivi costi di revisione delle scelte. Difficilmente sostenibile è poi l'indipendenza delle scelte individuali ipotizzata da Arthur. L'influenza degli altri soggetti del sistema sulle scelte di adozione delle tecnologie è ampiamente esaminata dalla teoria economica, a partire dai primi studi di matrice epidemiologica, in cui la diffusione avviene per contagio dei non adottanti da parte degli adottanti (Mansfield, 1961; Bass, 1969, 1980), fino ai recentissimi modelli di interazione locale[7] in cui ogni le decisioni di ognuno sono influenzate da quelle di un sottoinsieme di soggetti, definiti a lui vicini sulla base di un'appropriata metrica. Alle interazioni locali fanno riferimento Dalle e Jullien (1999) per spiegare la diffusione del sistema Linux in sostituzione di Window NT. Secondo gli autori per un utente di un sistema operativo non è rilevante tanto *il numero totale degli altri soggetti che lo utilizzano quanto il numero di coloro che lo fanno all'interno del gruppo con cui normalmente l'individuo interagisce*. Emerge qui una diversa fonte di diversità tra i soggetti classificata da Manfredi et al. (2000) come "eterogeneità sociale". Le differenze tra gli agenti non riguardano soltanto le loro caratteristiche intrinseche ma anche la rete delle relazioni sociali che sono in grado di attivare al fine di ottenere informazioni che li aiutino nel loro processo di scelta. Le caratteristiche di tale rete sono diverse a seconda che i nodi siano rappresentati da utenti di software Open Source oppure da chi impiega software commerciale. Un fenomeno diffuso nell'Open Source è, infatti, la cosiddetta "Advocacy" teorizzata dalle figure di spicco del movimento. Si tratta di una sorta di marketing one-to-one, in base al quale gli utenti dei programmi Open Source sono invitati a convincere altri membri del loro collettivo di riferimento a fare altrettanto, e ad abbandonare il mondo commerciale. L'Advocacy ha una crescita esponenziale: tra gli scopi vi è quello di trasformare un soggetto in un nuovo adepto del movimento e quindi in un potenziale "avvocato". Il ruolo di un "Open Source advocate" è allora assai simile a quello dei "diffusion agent" discussi da Witt, che non solo diffondono informazioni circa la nuova tecnologia ma anche provano a convincere un gruppo di potenziali adottanti a farlo simultaneamente in modo da contrastare le diseconomie che,

[7] Per una rassegna dei modelli di interazione locale si veda Fagiolo G. (1998)

in presenza di esternalità di network, penalizzano chi sceglie per primo una nuova tecnologia. Il processo di diffusione del software Open Source sembra quindi soddisfare le ipotesi che Witt dimostra essere alla base dell'esistenza di Masse Critiche nel processo di *spread* della nuova tecnologia e che permettono appunto l'avvicendarsi degli standard. Secondo Schelling (1978) il termine è mutuato dalla fisica ed indica la quantità di combustibile radioattivo necessaria ad innescare la fissione nucleare. Trasponendo questo concetto all'ambito economico e sociale, si parla di effetto *Massa Critica* quando determinate variabili che caratterizzano un processo superano una data soglia per cui il fenomeno esplode facendo sì che il sistema abbandoni l'equilibrio stabile su cui si trovava posizionato per passare ad un nuovo equilibrio altrettanto stabile. Nei modelli di diffusione di tecnologie la suddetta variabile è rappresentata dal numero di adottanti. Le simulazioni di Dalle e Jullien analizzano il fenomeno della Massa Critica, definita *"thresholds effects"* (1999, 14) nel sentiero di diffusione di Linux generato dal modello. Osservano che la sua emergenza dipende dal valore assunto da un particolare parametro a, che è una misura della *"preferenza degli utenti per la standardizzazione"* all'interno della loro rete di riferimento: più grande è α, più i potenziali utenti sono propensi a adottare lo standard attuale, ossia Window NT. Tale parametro è quindi anche una misura della forza del "proselitismo" degli utenti di Open Source: minore è a, maggiore è la loro forza all'interno di ogni rete locale. Le simulazioni evidenziano la presenza di *"thresholds effects"* quando il valore di a è sufficientemente basso.

Altri autori sono concordi nel riconoscere l'importanza rivestita in quest'ambito dalle aspettative che possono riguardare sia le performance della tecnologia (Huberman, 1998) sia l'ampiezza finale della rete di coloro che la utilizzano. Analizzando la diffusione di due tecnologie in competizione tra loro, le cui performance crescono al crescere del numero di chi le adotta, Huberman conclude che il tempo di "attesa" prima dell'insorgenza della Massa Critica decresce quanto più sono ottimistiche le aspettative dei soggetti. La transizione dall'equilibrio iniziale in cui tutti impiegano la vecchia tecnologia a quello in cui la nuova tecnologia ha soppiantato la vecchia, è poi tanto più veloce quanto più i soggetti sono eterogenei. Le sue simulazioni mostrano che, in presenza di un piccolo gruppo di soggetti dotati di propensione ad innovare doppia della

media del gruppo, la Massa Critica emerge quasi subito. Di nuovo risultata evidenziato il ruolo degli hacker e della loro cultura nel movimento Open Source.

6. Conclusioni

Il confronto tra il paradigma proprietario e quello libero non deve necessariamente essere svolto in termini di superiorità e di minaccia alla sopravvivenza.

L'industria privata del software ha dimostrato, fin dalla affermazione della traiettoria microelettronica, di essere capace di guidare processi di progressiva disintegrazione verticale e autonomizzazione dalla industria dell'hardware (Nelson, Winter, Malerba e Orsenigo, 2001; Torrisi, 2001) e di generare notevoli aumenti di produttività e riduzione dei costi di sviluppo (Cusumano, 1998). La disponibilità di software a basso costo, sia su package che su programmi customizzati è certamente uno degli elementi che ha contribuito alla diffusione delle ICT nei paesi avanzati e, in particolare, alla lunga ondata di crescita degli anni '90. Il paradigma proprietario ha consentito dunque imponenti processi di innovazione.

In questo senso è certamente eccessivo sostenere che l'Open Source metta radicalmente in discussione la stessa esistenza di software a pagamento. L'emergenza del software libero come paradigma di produzione non significa necessariamente la fine del software proprietario, ma piuttosto la possibilità di un nuovo equilibrio di coesistenza competitiva. Appare difficile prevedere gli esiti finali di questa dinamica in termini di quote di mercato dei due paradigmi.

Tuttavia alcuni elementi fanno ritenere che il paradigma proprietario debba progressivamente convergere verso una accettazione della necessità di rendere trasparente almeno in parte i codici sorgente, entrando di fatto entro un *modello di business ibrido*. Infatti, da un lato alcuni grandi operatori IT (si pensi a IBM) si sono mossi verso una legittimazione esplicita dell'Open Source e lo sviluppo di soluzioni compatibili sia con standard Microsoft che con Open Source, dall'altro molti grandi utilizzatori industriali e di servizi (governi, grandi imprese), soprattutto in Europa, hanno iniziato a imporre clausole di trasparenza dei codici sorgente nei capitolati di acquisto. A queste tendenze si deve aggiungere la continua nascita di piccole imprese di software che sviluppano soluzioni Open Source, trovando remunerazione nei servizi

aggiuntivi. Ciò allarga la base installata del software libero e inizia a creare effetti di esternalità di rete, pur in presenza di uno standard dominante.

La diffusione di Open Source ha inoltre avuto dinamiche radicalmente differenziate in funzione della presenza di *first mover advantage*. Nel settore dei sistemi operativi lato client, sui quali Microsoft era già leader dominante quando Linux ha iniziato la sua vicenda, la quota di mercato di Window supera il 50% su gran parte dei mercati geografici. Al contrario, nel settore dei server Web, dove la tecnologia si è affermata più tardi e il vantaggio di Microsoft non era consolidato, il sistema Open Source Apache è leader incontrastato. Ciò suggerisce che chi gode del vantaggio della prima mossa e aggrega rapidamente effetti di esternalità di rete è nella posizione migliore per dominare il mercato. L'equilibrio competitivo finale si collocherà in un qualche punto intermedio tra i due monopoli, ad una distanza variabile a seconda della area applicativa e della sua storia di competizione.

A conclusioni coerenti con questo impianto giunge anche la recente letteratura economica che si è occupata del rapporto tra path dependence e natura dell'equilibrio asintotico di diffusione. Nella prima famiglia di modelli à la Arthur e David, infatti, l'equilibrio dinamico prevedeva la dominanza di una tecnologia su quella concorrente, anche se tecnologicamente superiore. Il risultato era evidentemente rilevantissimo dal punto di vista teorico (affermazione di traiettorie inefficienti e lock-in), ma poco plausibile in una grande varietà di situazioni empiriche. Alcune caratteristiche della formalizzazione e delle assunzioni di base sono tuttavia responsabili di questo esito, mentre modifiche nei modelli possono portare a equilibri di coesistenza tra tecnologie, variamente configurati (Witt, 2000). In generale, la coesistenza appare una situazione più generale (Bassanini e Dosi, 2000).

Analizzandone le peculiarità, questo lavoro dimostra come il movimento Open Source non sia "un assurdo" ma possa essere ricondotto all'interno della scienza economica, attraverso i più recenti contributi delle teorie dell'azione collettiva, del coordinamento in assenza di autorità centrale e della diffusione di tecnologie in presenza di esternalità di network. Ciò ha permesso di fare luce non solo su come il fenomeno nasce e trova vasta applicazione economica, ma anche sui meccanismi alla base della sua sempre più massiccia diffusione. Le questioni aperte sono tuttavia ancora molte. Occorre sicuramente una più approfondita analisi dei meccanismi di coordinamento con

particolare riferimento al funzionamento esatto dei progetti Open Source, al fine di comprendere il ruolo in essi rivestito sia dai “leader” del progetto sia delle “figure minori” che svolgono il “non-sexy work” Proprio nella ricerca di una spiegazione del perché il “non-sexy work” viene svolto, sta una delle sfide più interessanti verso la totale comprensione economica dell’Open Source. Mentre sia la sociologia sia la teoria economica degli incentivi, riescono a dare conto di come un ricercatore universitario finlandese abbia potuto scrivere il kernel Linux senza essere pagato, maggiori sono le difficoltà nel comprendere cosa spinge un programmatore a lavorare alla realizzazione di una noiosa interfaccia grafica che renda lo stesso Linux user-friendly. Questo punto è fondamentale per due ragioni: da una lato, la tendenza dei programmi Open Source a diventare sempre più user-friendly permetterà la loro diffusione in fasce sempre più ampie della popolazione, dall’altro i programmi user-friendly e l’assistenza all’utente sono il core business di molte delle nuove società che, riuscendo a fare profitti con l’Open Source, dimostrando che nel caso del software il termine *libero* ha il significato di *free speech* e non di *free beer*.

Certamente il fenomeno Open Source non è l’unico della nuova società dell’informazione a creare problemi economici interessanti. La trasformazione del software da bene privato a bene collettivo, realizzata da Torvalds e dal movimento successivo sembra inverare la previsione che Kenneth Arrow aveva formulato su *American Economic Review* nel 1994, avvertendo che in contesti ad elevata innovazione ad intensità di conoscenza i beni non hanno più una natura fissa ma partecipano in grado diverso della natura di bene privato e appropriabile e di bene pubblico e universale.

Il fatto che la teoria economica sia attrezzata per lavorare meglio sui casi semplici, con teorie specializzate per i beni privati (i mercati) o i beni pubblici (la finanza pubblica), e sia in affanno sui casi complessi, è ovviamente un suo problema. Fortunatamente, non è un problema del mondo reale, che procede speditamente anche senza la teoria economica.

Bibliografia

Arrow K. (1994) *Methodological Individualism and Social Knowledge.* American Economic Review.

Arthur W. B. (1989) *Competing Technologies, Increasing Returns, and Lock-in by Historical Events*. The Economic Journal 99, pages 116-131.

Arthur W. B. (1994) *Increasing Returns and Path Dependence in the Economy*. University of Michigan Press, Ann Arbor.

Arthur W. B. (1996) *Increasing Returns and the New Word of Business*. Harvard Business Review, pages 100-109.

Arthur W. B., Ermeliov Y., Kaniovski Y. (1983) *On Generalised Urn Schemes of Polya Kind.* Cybernetics 19, pages. 61-71.

Bass M. F. (1969) *A New Product Growth Model for Consumer Durables*. Management Science 15(5), pages 215-227.

Bass M. F. (1980) *The Relation between Diffusion Rate, Experience Curves, and Demand Elasticity for Consumer Durable Technological Innovations.* Journal of Business 53(3), pages 551-567.

Bassanini A., Dosi G. (2000) *Heterogenous Agents, Complementaries, and Diffusion. Do Increasing Returns Imply Convergence to International Technological Monopolies?* LEM Working Paper

Bensen S. M., Farrell J. (1994) *Choosing How to Compete: Strategies and Tactics in Standardisation*. Journal of Economics Perspectives 8(2), pages 117-131.

Bessen J. (2001) *Open Source software: free provision of complex public goods.* NBER Working Paper

Bezroukov N., (1999) *Open Source software as a special type of academic research (critique to vulgar Raymondianism). First Monday 4.*

Bonaccorsi A. Rossetto S. (1999) *Modular Design under Market Uncertainty*. Paper Presented to the University of Chicago International Conference on Real Options, July.

Chamberlain J. (1974) *Provision of collective goods as a function of group size*. American Political Science Review 68, pages 707-716.

Cogenio s.r.l. (2001) *Indagine circa la propensione delle PMI industriali a migrare verso sistemi operativi alternativi: il caso Linux.* Report presentato a SMAU 2001.

Dalle J. M., Jullien N. (1999) *NT vs. Linux or Some Explanation into Economics of Free Software*. Paper Presented at "Applied Evolutionary Economics", Grenoble, June 7-9.

David P. (1985) *CLIO and the Economics QWERTY*. American Economic Review 75, pages 332-337.

Di Bona C., Ockman S., Stone M., Eds. (1999). *OPENSOURCES. Voci dalla Rivoluzione Open Source.* Apogeo, Milano.

Economic Review, 84 (2) pages 1-9

Fagiolo G. (1998) *Spatial Interactions in Dynamic Decentralised Economies.* In Cohendet P., Llerena, P., Stahn, H. and Umbhauer, G. (Eds.), The Economics of Networks: Interaction and Behaviours. Springer Verlag, Berlin.

Farrell J., Saloner G. (1985) *Standardization, Compatibility, and Innovation*. Rand Journal 16, pages 70-83.

Forrester (2001) *Open Source cracks the code.*

Fox J., Guyer M., Humburger H (1975) *Group size and Cooperation.* Journal of Conflict Resolution 19, pp 503-519.

Gartner Group (2001) *An end-user analysis. Linux server market share: where will be in 2001 and how will grow?*

Ghosh R., Prakash V. V.(2001) *The Orbiten Free Software Survey.* First Monday

Glass R. L. (1999) *Of Open Source, Linux and Hype.* IEEE Software 16(1), pages 126-128.

Glass R. L. (2000) *The sociology of Open Source: Of Cults and Cultures*. IEEE Software 17(3), pages 104-IBC.

Green L. G. (1999). *Economics of Open Source Software*. http://www.badtux.org/eric/editorial/economics.html.

Hardin R. (1982) *Collective Action*. John Hopkins University Press, Baltimore.

Hecker F. (1999) *Setting Up Shop: The Business of Open-Source Software.* IEEE Software 16(1), pages 45-51.

Hermann G., Hertel S., Niedner S. (2001) *Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel*. University of Kiel Working Paper

Hertel G. (2002) *Management of virtual teams based on social psycology models.* In E.H. White (Editor), Pabst Publisher, Lengerich.

Hurberman B., Loch C. (1998) *A Punctuated Equilibrium Model of Technology Diffusion.* Xerox Palo Alto Research Centre, Palo Alto.

IDC (2001) Report available at ***http://dailynew.yahoo.com/h/zd/20010611/new_report_questions_linux_server_claims_1.html***

Katz M. L., Shapiro C. (1985) *Network Externalities, Competition, and Compatibility*. American Economic Review 75, pages 424-444.

Koch S., Schneider G. (2002) *Effort, co-operation and co-ordination in an Open Source software project: GNOME.* Information System Journal 12, pages 27-42.

Lakani K., von Hippel E. (2000) *How Opwn Source works: "Free" user-to-user assistance.* MIT Sloan School of Management Working Paper #4417.

Lerner J., Tirole J. (2001) *The Open Source movement: key research questions*. European Economic Review 45, pages 819-826.

Liebovitch E. (1999) *The Business Case for Linux*. IEEE Software 16(1), pages 40-44.

Liebowitz S. J., Margolis S. E. (1990)*The Fable of Keys.* Journal of Law and Economics 33, pages 1-26.

Liebowitz S. J., Margolis S. E. (1996) *Market Process and Selection of Standards.* Harward Journal of Law and Technology 9, pages. 283-318.

Malerba F., Nelson R., Orsenigo L., Winter S. (2001*) History-Friendly models: An overview of the case of Computer Industry.* Journal of Artificial Societies and Social Simulation 4(3)

Manfredi P., Bonaccorsi A., Secchi A. (2000). *Heterogeneity in New Product Diffusion.* LEM Working Paper.

Mansfield E. (1961) *Technical Change and the Rate of Imitation*. Econometrica 29, pages 741-766.

Marwell G., Oliver P., Prahal R. (1993) *The Critical Mass in Collective Action. A Micro-social Theory*. Cambridge University Press, Cambridge.

Mauss M. (1959) *The Gift. The form and the reason for exchange in archaic societies.* Routledge, London

Mockus A., Fielding R., Herbsleb J. (2000) *A case study of Open Source software development: the Apache server*. In Proceedings of the 22nd International Conference on Software Engineering, pages 263-272.

Olson M.(1965) *The Logic of Collective Action.* Cambridge University Press, Cambridge Massachusetts.

Perkins G. (1999) *Culture Clash and the Road of Word Dominance*. IEEE Software 16(1) pages 80-84.

Raymond E. S. (1999) *The Cathedral and the Bazaar*. http://www.redhat.com/redhat/cathedral-bazar/.

Roger E.M. (1995) *Diffusion of innovations*. Free Press, New York, 4th edition.

Rossi C. (1998) *Il Problema dell'Azione Collettiva nella Teoria Economica: Una Rassegna di Alcuni Recenti Contributi*. Tesi di laurea non pubblicata, Università di Pisa

Sanders J. (1998) *Linux, Open Source and Software's Future*. IEEE Software 15(5), pages 88-91.

Schelling T. (1978) *Micromotives and macro behavior*. New York, Norton

Schimitz P.E. (2001) *Study into the use of Open Source software in the public sector*. An IDA Study (Interchange of Data between Administrations).

Stone M. (1998) *Science of the New Renaissance*. http://www.edventure.com/release1/1198.html.

Torrisi S. (1998) *An international study of software industry*. Edward Elgar Publishing

Torvalds L. (1999) *Il vantaggio di Linux.* In Open Sources: Voci dalla Rivoluzione Open Source, a cura di Chris Di Bona, Sam Ockman, Mark Stone. Apogeo, Milano.

Ullman E. (1998) *The Dumbing Down of Programming.* 21st Salon, 13 May, http://www.salonmagazine.com.

Varian H. L., Shapiro C. *Information Rules.* Etas Libri, Milano.

Wallich P. (1999) *Cyber View The Best Things in the Cyberspace Are Free.* Scientific American March, page 44.

Witt U. (1997) *Lock-in vs. Critical Masses. Industrial Changes under Network Externalities*. International Journal of Industrial Organisation, 15 pages 753-772.

Witt U. (2000) *Path-dependence in Institutional Change*. In: K. Dopfer (ed.), The Evolutionary Principles of Economics. Cambridge, Cambridge University Press