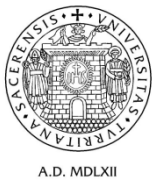
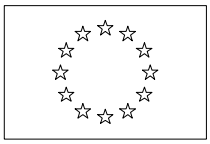
MIUR





F.S.E.

Università degli Studi di Sassari
Dottorato di Ricerca in Diritto ed Economia
dei Sistemi Produttivi

PEER PRODUCTION ED OPENNESS: ASPETTI ISTITUZIONALI, PROFILI GIURIDICI ED IMPLICAZIONI STRATEGICHE

Coordinatore:
Chiar.mo Prof. Michele M. Comenale Pinto

Tutor:
Chiar.mo Prof. Luca Ferrucci

Tesi di dottorato della
Dott.ssa Margherita Piredda

Anno Accademico 2007-2008

Indice

Introduzione

In questa tesi ho cercato di approfondire il tema del modello di creazione dell'innovazione offerto dal Free/Libre and Open Source Software, distillando la letteratura economica ed in parte giuridica pubblicata nel corso degli ultimi dieci anni.

Il primo capitolo ripercorre l'evoluzione storica del movimento FLOSS, mettendo in luce come in realtà esso affondi le radici sia in nel modello aperto di condivisione della conoscenza tipico del mondo scientifico, sia nella cultura hacker, e come i diversi steps in cui tale evoluzione si articola siano stati determinati da rilevanti innovazioni tecnologiche.

Nel secondo capitolo ho affrontato i problemi giuridici sollevati dal modello FLOSS; in particolare ho analizzato la tutela del software, a partire dalla disciplina del diritto d'autore vigente negli Stati Uniti, in Europa ed in Italia, passando poi a mettere in evidenza il processo di transizione verso una tutela brevettuale dei programmi informatici, attuata dagli Stati Uniti. Dopo aver offerto un rapido excursus sulla letteratura economica in tema di efficacia della tutela brevettuale nell'appropriazione delle rendite da innovazione, ho analizzato il modello di tutela della proprietà intellettuale offerto dalle licenze Free/Libre e Open Source.

Nel terzo capitolo ho analizzato il fenomeno del Free/Libre ed Open Source Software come forma creativa di "consumer resistance", filone letterario mutuato dalla consumer theory, che afferma la crescente "agency" del consumatore, in una visione postmoderna dell'attenuazione del confine tra momento della produzione e

momento del consumo nella creazione del valore manifestatasi inizialmente nelle cosiddette brand communities, per poi concretizzarsi, grazie alla creazione di relazioni all'interno delle cosiddette *gift economies*, in vere proprie comunità di innovazione in cui si passa da forme di produzione simbolica alla realizzazione di veri e propri artefatti ad opera del consumatore.

Nel quarto capitolo ho analizzato il fenomeno del software open source come modello privato di creazione di beni pubblici, evidenziando aspetti quali le motivazioni dei partecipanti ai progetti open source, nonché gli aspetti organizzativi legati al coinvolgimento di nuovi utenti in una sorta di comunità di pratica in assenza, almeno in prima approssimazione, di una forma di organizzazione del lavoro riconducibile alla tassonomia preesistente basata sulla dicotomia tra mercato e gerarchia.

Nel quinto ed ultimo capitolo ho infine evidenziato i processi di progressiva integrazione delle imprese nei progetti open source, in un modello che sembrerebbe riconducibile al nascente paradigma dell'*open innovation*, e il modo con cui l'open innovation diventa un modello di innovazione vincente, attraverso i diversi business models oggi esistenti per la cooptazione di fonti di innovazione terze rispetto all'impresa, rappresentate, in questo caso, dalle comunità degli utenti e sviluppatori nate attorno ai progetti open source.

Il fine ultimo di questa dissertazione è mostrare come esistano situazioni in cui l'innovazione non è il risultato dell'investimento in ricerca e sviluppo, realizzato dall'impresa e da questa tutelato attraverso i diritti sulla proprietà industriale, ma sia il risultato dello sforzo cooperativo e non remunerato economicamente di utenti finali

del prodotto, nonché mostrare come il risultato ottenuto sia tutt'altro che effimero, rendendo possibile definire le comunità di utenti come fonti di innovazione terze all'impresa.

Capitolo 1

Evoluzione del Free/libre and open source software

1.1 Introduzione

Sono ormai passati 17 anni, da quando, il 5 ottobre 1991, Linus Torvalds, giovane studente di informatica dell'Università di Helsinki, postava sul newsgroup comp.os.minix l'annuncio (fig. 1.1) della creazione di un kernel di sistema operativo molto simile a Minix, chiedendo l'aiuto di chiunque volesse partecipare alla creazione di quello che sarà il software GNU/Linux.

Fig. 1.1: La genesi di Linux – Il messaggio di Linus Torvalds

```
Message-ID:
1991Aug25.205708.9541@klaava.helsinki.fi
From: torvalds@klaava.helsinki.fi (Linus Benedict Torvalds)
To: Newsgroups: comp.os.inix

Subject: What would you like to see most in minix?

Summary: small poll for my new operating system
Hello everybody out there using minix-I'm doing a (free)
operating system (just a hobby, won't be big and professional
like gnu) for 386 (486) AT clones. This has been brewing since
april, and is starting to get ready. I'd like any feedback on
things people like/dislike in minix, as my OS resembles it
somewhat
Any suggestions are welcome, but I won't promise I'll implement
them :-)

Linus
```

Fonte: Wynants & Cornelis (2005)

Attualmente, il software open source rappresenta un mercato in notevole crescita: nel 2006 il fatturato mondiale da esso derivante aveva raggiunto la cifra di 1,8 miliardi di dollari (IDC, 2007), cifra

destinata a raggiungere i 5,8 miliardi di dollari nel 2011, con un tasso di crescita del 26%. Queste cifre sono rilevanti, se si considera che la quasi totalità del software open source è distribuita gratuitamente, e le cifre sopra riportate si riferiscono al mercato derivante dalle distribuzioni a pagamento, prevalentemente destinate al segmento business.

Il software a codice sorgente aperto, per essere tale, deve uniformarsi alla cosiddetta *Open Source Definition*. La Open Source Definition è, sostanzialmente, una carta dei diritti degli utenti di computers (Perens, 1999), in quanto individua i diritti dell'utente che una licenza open source deve garantire per poter essere definita Open Source, che consistono, in prima approssimazione, nel:

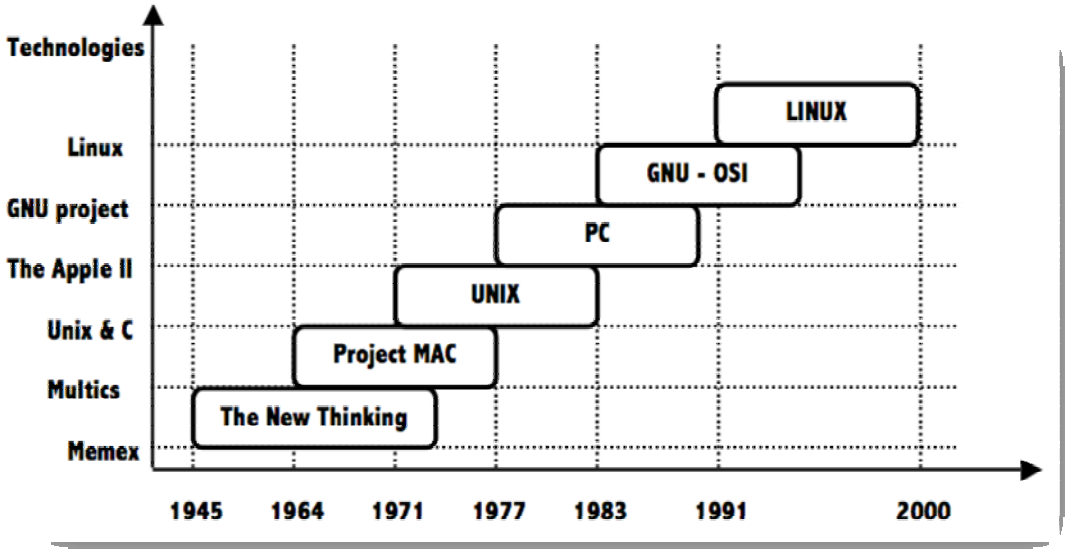
- Diritto di fare copie del programma e di distribuirlo;
- Diritto di accesso al codice sorgente del software, condizione necessaria per poterlo modificare;
- Diritto di apportare migliorie al programma.

Secondo Benussi (2006) è possibile individuare sei diverse fasi nell'evoluzione del floss, ciascuna delle quali fortemente connotata da un evento storico rilevante, che riassume le maggiori caratteristiche del periodo e che rappresenta il meccanismo in grado di determinare la variazione rispetto al periodo precedente (Benussi, 2006). E'possibile, quindi, creare una mappa cronologica dell'evoluzione del Floss (fig. 1.2)

Il primo stadio descrive la nascita di una nuova era che ha influenzato i successivi sistemi innovativi, determinata dallo sviluppo delle tecnologie informatiche. Questo periodo è iniziato dopo la

Seconda Guerra Mondiale, grazie al contributo seminale di Vannevar Bush (1945), e si estende durante i primi vent'anni dello sviluppo delle nuove tecnologie informatiche. Tale periodo ha caratterizzato nuove sfide tecnologiche e scientifiche, ed in particolare la creazione di una nuova industria basata sui computers, e i suoi effetti sono ancora evidenti nell'odierno settore del software. La creazione dell'UNIVAC e del sistema SAGE rappresentano i più rilevanti progetti che hanno aperto la strada alle future innovazioni.

Fig. 1.2: Mappa cronologica del FLOSS



Fonte: Benussi (2006)

La seconda fase corrisponde alla creazione e all'evoluzione di uno dei progetti leader nell'ambito informatico: il progetto MAC al Massachussets Institute of Technology, che ha creato il sistema compatibile *time sharing* (Compatible Time Sharing System o CTSS) e il sistema Multics.

Il terzo periodo descrive la nascita del software su cui l'Open Source si basa: UNIX. Esso è caratterizzato dallo sviluppo del sistema operativo UNIX, dalle sue origini a fini scientifici alla sua maturità, corrispondente alla creazione della Sun Microsystems, una delle più importanti ed innovative società open source-oriented.

Il quarto step si identifica con la creazione del sistema tecnologico del personal computer, e con la nascita del relativo mercato. Nuove imprese iniziarono a produrre personal computers, tra tutte la Apple Macintosh e, allo stesso tempo, nuove società, come la Microsoft, iniziarono ad elaborare softwares proprietari specifici per i PC. Rapidamente i nuovi sistemi, basati sulla combinazione pc-software proprietario, diventarono il prodotto di maggior valore nel settore ICT.

Il quinto stadio evolutivo si identifica con la creazione del progetto GNU e della Free Software Foundation. Inoltre, durante questo periodo, inizia una prima diffusione su larga scala di Internet e si assiste alla creazione del World Wide Web. Nel frattempo fu creata la Open Source Initiative al fine di sostenere il crescente successo tecnico, economico e sociale del movimento Free/Open Source.

Infine, il sesto periodo si identifica con la nascita del sistema operativo GNU/Linux e i cambiamenti che il nuovo modello di creazione e diffusione del software ha provocato sul movimento Open Source. Questo periodo ha inizio nel 1991 quando Linus Torvalds postò on-line il link per il download del suo kernel ed è terminato nel 2001 con l'avvento del cosiddetto Web 2.0.

1.2 Il nuovo modo di pensare: 1945-1969

La genesi dell'atmosfera intellettuale che ha generato la consuetudine della condivisione dei programmi informatici è dovuta alle nuove pratiche collaborative sviluppate dai ricercatori che operavano nei laboratori pubblici e privati in Europa e negli Stati Uniti dopo la Seconda Guerra Mondiale.

Tab. 1.1: Fondi per la ricerca – U.S.A. 1930-1998 (milioni di $ al valore del 1992)

Anno	Governo	Industria	Università	Altri no-profit	Totale
1930	248	1.195	210	59	1.712
1940	614	2.077	280	94	3.063
1955	17.977	12.902	453	318	31.650
1960	39.185	20.281	666	538	60.670
1970	53.559	26.944	1.099	894	82.498
1975	49.534	34.543	1.544	1.122	86.743
1980	43.070	37.084	1.810	1.273	83.237
1985	48.022	50.133	2.175	1.469	101.799
1991	63.035	95.030	3.505	3.372	164.942
1995	59.375	102.994	3.816	3.679	169.864
1998	59.083	125.469	4.342	3.717	192.611

Fonte: Ns. adattamento da Chesbrough (2003)

Durante la guerra, infatti, molti scienziati, come la comunità di fisici raccolti nel progetto Manhattan, furono costretti a collaborare a progetti statali per la creazione principalmente di nuove armi, ed il successo – devastante in termini di effetti pratici – del progetto Manhattan rafforzò, a livello politico, la consapevolezza che una supremazia a livello tecnologico portasse automaticamente alla supremazia a livello geo-politico, e determinò un'enorme flusso di risorse finanziarie per le attività di ricerca.

La notevole disponibilità di risorse finanziare a sostegno di nuovi progetti di ricerca, unita alla consapevolezza del rapido progresso tecnologico raggiungibile, generarono due importanti conseguenze: la comune comprensione dell'importanza della scienza per la società e la spinta a produrre, diffondere e gestire crescenti quantità di dati.

Uno degli obiettivi più sfidanti era il perfezionamento delle tecnologie per la comunicazione, al fine di superare il problema dell'elaborazione dei dati, e la soluzione proposta fu la condivisione dell'energia necessaria per far fronte alle difficoltà connesse con elaborazioni e calcoli sempre più complessi.

Quindi il concetto della condivisione delle risorse tecnologiche può essere visto come la nascita di un nuovo tipo di processo di sviluppo all'interno delle comunità scientifiche e si accompagna a notevoli investimenti statali nella ricerca scientifica.

Inoltre questo diffuso sforzo scientifico e tecnologico ha avuto un altro effetto cruciale sulla comunità scientifica che può essere definito come "connected-community effect", ed è dovuto al fatto che, durante la guerra, gli scienziati abbiano dovuto vivere insieme per molto tempo al fine di interagire con gli altri e condividere idee per la risoluzione di problemi. Questa esperienza di interazione proseguì dopo la Seconda Guerra Mondiale. In particolare, risultò essere particolarmente efficiente l'idea di condividere risorse per aumentare la velocità di calcolo e per migliorare l'accuratezza delle analisi. Bisognò quindi creare dei nuovi sistemi per la conservazione e l'organizzazione dell'informazione, e la comunità scientifica accettò subito tale sfida.

Il concetto di ipertesto fu la prima risposta al bisogno di "thinking connected", e si intuì la necessità di creare migliori tecnologie per la comunicazione. Sulla stessa linea, anni dopo, si capì l'importanza di creare sistemi di connessione tra i computers in grado di resistere ad eventuali attacchi esterni.

La struttura più adatta era senza dubbio rappresentata da un sistema distribuito, in contrapposizione alle infrastrutture centralizzate per la comunicazione esistenti al tempo. Questa nuova struttura influenzò tutti i futuri sviluppi delle communication technologies, e generò nuove opportunità; ad esempio, i maggiori centri di ricerca acquisirono la possibilità di collegare i computers dislocati nei propri laboratori con database esterni, e, quindi, di migliorare la propria capacità di gestire crescenti flussi di informazioni.

Durante gli anni Sessanta, poi, si cominciò ad elaborare e a condividere i programmi per i computers.

Nel 1961 il Massachussets Institute of Technology acquistò il primo PDP-1, prodotto dalla DEC; a partire da tale data, i ricercatori cominciarono a creare comunità attorno a questi computers per la creazione, miglioramento e condivisione dei software necessari per il loro funzionamento. Iniziarono quindi a sviluppare una sottocultura legata all'elaborazione dei programmi, che può essere vista come un prototipo della cultura hacker.

La cultura hacker è basata sull'assunto che la comunicazione aperta e libera è la chiave per l'innovazione. Pochi anni dopo, nel 1969, iniziò il progetto ARPAnet. Lo sviluppo di questa nuova infrastruttura per la comunicazione coinvolse un grande numero di forze intellettuali e indusse a ricercare nuove opportunità per la

comunicazione. Nello stesso anno, UNIX venne creato da uno scienziato dei Laboratori Bell della AT&T, Ken Thompson. L'UNIX originale può essere considerato come un nipote del CTSS, uno dei primi sistemi time-sharing o a condivisione di tempo.

UNIX è un sistema operativo che si caratterizza per essere:

- Multiutente: più utenti possono interagire contemporaneamente, da terminali diversi, con il sistema, che evita interferenze tra le attività dei vari utenti. All'interno del sistema ogni utente è individuato da uno *username*, e inoltre più utenti sono suddivisi in gruppi, ciascuno dei quali individuato univocamente (*groupname*). In ogni sistema è identificato un utente *root*, che rappresenta l'amministratore di sistema e che, in generale, non ha alcuna limitazione nell'accesso alle risorse del sistema stesso.
- Multiprogrammato: il suo nucleo può supportare la contemporanea esecuzione di più processi gestiti a divisione di tempo.
- Gestione della memoria virtuale: il sistema della gestione della memoria in UNIX si basa su paginazione e segmentazione. Queste caratteristiche consentono ad ogni processo di indirizzare un'area di memoria di dimensioni eventualmente superiori a quella della memoria centrale effettivamente disponibile.
- Portabile: grazie all'impiego del linguaggio C nella realizzazione del sistema, esso gode di un'elevata portabilità.
- Aperto: oggi realizza alcuni dei più diffusi protocolli di comunicazione per Internet.

- Ambiente di sviluppo per programmi C: Unix ha uno stretto legame con il linguaggio C (Wikipedia, 2008).

Sia la creazione di UNIX che di ARPAnet rappresentano un nuovo modo di concepire i computers, che si apprestavano a non essere più delle mere macchine, ma elementi di una rete, di una comunità.

1.3 Il Progetto MAC: 1964-1975

La necessità di realizzare un sistema operativo in grado di rendere possibile la condivisione da parte di più utenti dello stesso computer mediante la condivisione a tempo è stata la determinante del progetto MAC, condotto da ricercatori dell'M.I.T., progetto da cui sono poi derivati il laboratorio per l'intelligenza artificiale (MIT Artificial Intelligence Laboratory) e il MIT Laboratory for Computer Science.

Le invenzioni elaborate da queste strutture del MIT costituiscono le fondamenta del FLOSS, basti pensare al fatto che Richard Stallman, creatore del Free Software Foundation, ha iniziato la sua carriera all'interno del progetto MAC (Stallman, 1999). Una delle caratteristiche più rilevanti del modello di sviluppo delle conoscenze rappresentato dal MAC è senza dubbio lo sforzo di creare una comunità epistemica supportata da un network tecnologico.

Indirettamente, con il progetto MAC, è stato ottenuto un altro risultato, ovvero la creazione di un sistema di stoccaggio di dati online che è diventato un modo per consentire alle persone lo scambio e la condivisione di dati e di programmi (Benussi, 2006).

In tal modo si è creato non sono uno strumento pubblico utile ai ricercatori, ma anche una information community, in quanto il sistema

a condivisione di tempo ha generato conoscenze e capacità al servizio della comunità, a cui ciascun ricercatore poteva accedere secondo le proprie esigenze.

Il sistema operativo a condivisione di tempo elaborato dal Progetto MAC fu Multics, acronimo di MULTIplexed Information and Computing Service, iniziato a partire dal 1963. Il progetto inizialmente coinvolse il Computer Department della General Electric, che stava realizzando la parte hardware, e i laboratori della Bell, che decisero di lavorare con il MIT con la speranza che Multics sarebbe diventato il sistema operativo base per i laboratori.

La prima versione di Multics apparve nel 1969 e venne utilizzata per fini accademici, mentre la sua gestione venne affidata dal Progetto MAC al MIT Processing Center. Alla fine del 1971, Multics divenne operativo 24 ore su 24, sette giorni su sette, e serviva una comunità di circa 500 utenti.

Nel 1975, il nuovo direttore del Mac Project, Michael Dertouzos, cambiò il nome del progetto in Laboratory for Computer Science, terminando l'esperienza pioneristica della struttura del MIT.

L'importanza del Multics è legata alla straordinaria influenza che esso ha esercitato sui successivi sistemi operativi, il più importante dei quali è senza dubbio Unix. Multics, insieme all'eredità lasciata dal progetto ARPAnet, ha alimentato altre innovazioni, come il protocollo TCP/IP, il progetto GNU e successivamente il sistema operativo GNU/Linux.

1.4 Il linguaggio: 1971-1982

Può essere utile, al fine di comprendere appieno il significato del concetto di open source e perché si parla di software free/libre and

open source software in relazione al sistema operativo GNU/Linux, riassumere brevemente la storia dell'antenato di GNU, ovvero UNIX[1].

L'influenza di Unix su GNU/Linux è soprattutto nell'architettura e quindi nella diversa organizzazione del lavoro degli sviluppatori; in particolare, l'utilizzo di un diverso stile nella scrittura del codice influenza il disegno del sistema operativo così come il suo successivo sviluppo.

La filosofia che sta dietro Unix è che "piccolo è bello", nel senso che non è necessario creare un'interfaccia complessa al fine di realizzare un sistema articolato, ma si può creare un sistema complesso dall'interazione di elementi semplici[2]. Seguendo questa filosofia, gli inventori di Unix hanno creato il concetto delle "pipes". Una pipe è un meccanismo di comunicazione interprocesso che permette all'output di un processo di essere utilizzato come input di un altro. In questo modo, una serie di comandi può essere collegata ("piped"). In molte shell Unix questa caratteristica viene rappresentata da una barra verticale. Usando il meccanismo di pipe di Unix un utente può creare un programma composto da un numero teoricamente illimitato di piccole e specializzate utilities.

Grazie alle pipes viene introdotto nella programmazione una concezione modulare dell'architettura del software, che sarà la caratteristica fondante di GNU/Linux e di tutti i software open source.

Unix è stato ideato a partire dal 1969 da Ken Thompson, un giovane scienziato che, alle dipendenze dei laboratori Bell, era stato coinvolto nel progetto Multics, insieme a Dennis Ritchie e Doug

[1] Anche se sembrerebbe contraddittorio, visto che GNU è l'acronimo ricorsivo di GNU is Not Unix.
[2] Benkler (2002) evidenzia la cruciale importanza della modularità del software nel rendere possibile l'interazione a distanza e non coordinata tra gli sviluppatori nei progetti FLOSS.

McIlroy, che ugualmente lavoravano ad un progetto ispirato da Multics.

Una delle prime applicazioni di Unix fu su un word processor elaborato dal dipartimento brevetti dei laboratori Bell su un nuovo marchio, il PDP-11.

Il progetto ebbe come risultato la creazione di un intero nuovo sistema operativo, che riscosse un notevole successo, del tutto inatteso dai laboratori Bell e dalla AT&T. Il sistema Unix originario fu scritto in assembler e le sue applicazioni con un linguaggio interpretato chiamato B, che non fu in grado di realizzare i compiti assegnati. Quindi Dennis Ritchie aggiunse alcune modifiche critiche e iniziò lo sviluppo di un nuovo linguaggio chiamato C.

Thompson e Ritchie riuscirono a scrivere un nuovo sistema operativo in soli tre anni.

Si verificò quindi un evento che influenzò in maniera determinante lo sviluppo di Unix e diede vita alla prassi della condivisione del codice sorgente tra gli sviluppatori. Tale avvenimento fu una sentenza antitrust[3], che impose alla AT&T la scelta tra il nascente settore dell'informatica e quello della telefonia, con la conseguente scelta della società di uscire dal settore informatico. Ai laboratori della Bell fu imposto di concedere le licenze d'uso per Unix ad un prezzo nominale per impedire alla AT&T di renderlo un prodotto chiuso. In conseguenza di ciò, Ken Thompson ebbe il permesso di spedire i nastri contenenti il codice sorgente di Unix a chiunque fosse interessato ad averli. Anche se la

[3] Si tratta del decreto del Dipartimento di Giustizia americano dell'8 gennaio 1982, poi parzialmente modificato dal Giudice Harold H. Green della Corte Distrettuale con decreto del 24 agosto 1982 (Baker & Yandle, 1994)

AT&T aveva concesso il programma sotto diversi tipi di licenza, non riuscì a sfruttarne appieno le potenzialità economiche e quindi fu l'unico sistema operativo che, al tempo, veniva distribuito con il codice sorgente.

La disponibilità del codice sorgente si rivelò essere il meccanismo necessario per il coinvolgimento di sviluppatori esterni ai laboratori Bell, tanto che nel 1978 comparve il primo user group di Unix. A quel tempo, Unix veniva utilizzato come sistema operativo per l'intero sistema informativo della Bell e università di tutto il mondo iniziarono ad adottarlo ed utilizzarlo.

Nello stesso anno venne fondata la prima società basata su Unix, la Santa Cruz Operation (SCO). Cinque anni prima un altro importante attore era entrato a far parte del mondo di Unix: l'università di Berkeley. Nel 1974, infatti, Thompson vi tenne un ciclo di lezioni, trovando programmatori assolutamente entusiasti delle nuove rivoluzionarie caratteristiche di Unix. In particolare, in un laboratorio guidato da Bill Joy, venne creata una particolare versione di Unix, chiamata Berkeley Software Distribution (BSD), nel 1977 (Mc Kusick, 1999).

Nel 1980 si verificò un altro evento determinante per l'evoluzione di Unix. La Defence Advanced Research Project Agency identificò nella versione BSD di Unix elaborato dall'università della California il sistema operativo in cui implementare i nuovi protocolli TCP/IP.

Il risultato fu la BSD versione 4.2, rilasciata nel 1983, la prima versione di Unix ad avere l'implementazione dei protocolli TCP/IP.

Nel 1983 venne fondata la Sun Microsystems, con lo scopo di creare computers operanti su Unix capaci di operare in un network, e per raggiungere tale obiettivo si decise di utilizzare la versione BSD elaborata a Berkeley.

Nel 1982 Unix dimostrò di essere un sistema stabile e portabile ed un efficace linguaggio di programmazione. La crescente reputazione di Unix spinse la AT&T a riprendere il controllo sulla sua versione del software, riportandolo sui binari del modello proprietario grazie anche alle minacce di cause legali a chi fino a quel momento aveva liberamente modificato il codice sorgente, con la conseguente scomparsa dei contributi al sistema operativo fino a quel momento offerti dalle università.

Il ritorno di Unix al modello proprietario, l'unico modello che a quel tempo si credeva consentisse lo sfruttamento commerciale, comportò anche la frammentazione del mercato, mediante la creazione, da parte delle numerose nuove software companies, di versioni di Unix differenziate e tra loro incompatibili.

Nel 1983 Larry Wall inventò la patch, un programma eseguibile che consentiva agli sviluppatori di apportare cambiamenti incrementali e di diffonderli senza dover scambiare l'intero file sorgente. Fu uno degli strumenti più importanti a sostengono dello sviluppo cooperativo di software. L'importanza di questa innovazione spinse Larry Wall a creare un nuovo linguaggio di programmazione, Perl, oggi uno dei più importanti nel mondo open source.

1.5 La nascita del personal computer: 1977-1991

Questo periodo si caratterizza per diversi avvenimenti fondamentali per l'informatica e anche per il FLOSS, e che sono la

nascita del personal computer, con la definizione di uno standard da parte dell'IBM, con il suo pc IBM (1984), la definitiva separazione tra hardware e software, e il conseguente sviluppo di alcune software companies, tra cui spicca la Microsoft, il passaggio del software dall'iniziale modello aperto, tipico del modus operandi delle comunità scientifiche, al modello proprietario, ritenuto l'unico in grado di consentire l'ottenimento di profitti alle nascenti società del settore del software. Questo periodo termina nel 1991 con la creazione del world wide web da parte di Tim Barners-Lee.

Si può invece rinvenire l'inizio esatto di tale nuova fase evolutiva con la data di fondazione della Apple Computer da parte di Steve Jobs, Steve Wozniak e Mike Makkula, avvenuta il 3 gennaio 1977.

Propedeutica alla creazione della Apple era stata l'introduzione, da parte della Intel, del microprocessore 8080, nel 1974 e, successivamente, nel 1975, l'invenzione, da parte di una piccola società del New Mexico, del microcomputer Mits/Altair 8800.

Nel 1977 venne creato il sistema operativo specifico per l'architettura del microprocessore Intel 8080, il CP/M. Grazie all'introduzione di questo sistema operativo, una comunità di sviluppatori creò il primo software per un pc. Il CP/M non divenne però lo standard dominante nell'ambito del pc, perché nel 1977 venne lanciata sul mercato una nuova macchina rivoluzionaria, la Apple II, creata da Steve Jobs e Steve Wozniak. Il nuovo pc era basato sull'idea rivoluzionaria di rendere il personal computer fruibile dalla massa, vendendo insieme al pc una serie di applicativi tutti basati sulla

graphical user interface, rendendo effettivo il concetto per cui "what you see is what you get".

Fino al 1981, i leader del settore pc erano Apple, Commodore e Tandy, mentre le altre imprese agivano da follower. Il più importante di questi era IBM, che reagì alla sfida tecnologica nel 1981 con il lancio del "project chess", finalizzato alla realizzazione di quello che diventerà il pc-IBM, macchina che impose il proprio standard da quel momento in poi.

Il leader del progetto era Philip Don Estridge, il cui compito era la realizzazione di un personal computer che potesse essere usato facilmente dalla massa. Anche il pc-IBM adottò un'architettura aperta, e vennero acquistati componenti e softwares al di fuori della IBM. Una delle conseguenza di questa scelta fu l'adozione di un nuovo sistema operativo, il PC-DOS, creato da una nuova società di Seattle, la Microsoft Corporation. Un altro evento cruciale fu la possibilità per la Microsoft, grazie al consenso della IBM, di licenziare a terzi produttori il proprio sistema operativo.

Cominciarono a diffondersi cloni del pc IBM, anch'essi operanti con l'MS-DOS, che furono tollerati dalla IBM sia perché si registrò un'incredibile domanda di pc a cui essa non avrebbe potuto far fronte, sia perché la presenza di cloni del proprio pc contribuivano alla sua imposizione come standard; ovviamente questa situazione ebbe conseguenze molto positive anche per la Microsoft.

Tra il 1981 ed il 1984 si ebbero altre innovazioni determinanti per il software. Nel 1983 la Lotus di Mitch Kapor, su commissione della Apple, rilasciò il primo foglio di calcolo, il Lotus 1-2-3, a cui si aggiunsero i primi applicativi multimediali. Contemporaneamente, la

Lotus rilevò diverse software companies in modo da estendere il proprio portafoglio prodotti, che, negli anni '90, venivano venduti insieme nella Lotus Smartsuite.

L'altra innovazione rilevante registrata in questo periodo fu il Macintosh Apple. Dopo l'introduzione di Apple II, grazie al contributo di Jef Raskin, venne ideato e lanciato un nuovo computer, plug and play, formato da un case con lettore di floppy e da un monitor in bianco e nero di 9 pollici. L'obiettivo era la realizzazione della migliore interfaccia utente mai realizzata prima, cosa che venne fatta con l'interfaccia del Mac, che fu la più importante innovazione introdotta dalla Apple e tutt'ora il principale vantaggio competitivo dei computer Macintosh.

Il nucleo del progetto non era però originale della Apple, ma era il risultato di un progetto sulla GUI condotto nei laboratori Xerox a Palo Alto Ca., e mai sfruttato economicamente dalla Xerox, al pari di altre innovazioni[4] .

Il Macintosh Apple venne lanciato nel 1984 con una famosa campagna pubblicitaria[5] e il target market del nuovo pc era ancora una volta la massa dei consumatori, visto che il pc era ancora uno strumento utilizzato da un ristretto numero di utenti, obiettivo che non venne raggiunto neanche con questo prodotto.

Nello stesso tempo, anche Microsoft, grazie all'ottenimento di alcuni prototipi di Macintosh, aveva intuito le enormi potenzialità commerciali della GUI e, nel 1985, lanciò Microsoft Windows, che

[4] Per una dettagliata analisi dei problemi gestionali del portafoglio brevetti creato dal PARC di Xerox si rimanda aChesbrough (2003) e a Rivette & Kline (2000)

[5] Lo spot, trasmesso durante il Super Bowl, può essere visto su YouTube: http://www.youtube.com/watch?v=OYecfV3ubP8

aveva molti elementi del Macintosh OS, dando inizio ad una lunga battaglia giudiziaria, iniziata con una citazione ricevuta dalla Apple nel 1988 e conclusasi con un accordo tra le parti solo nel 1997.

Un'altra importante innovazione segnò l'evoluzione del mondo dell'informatica e della comunicazione: il world wide web.

Il web ha origine nei laboratori del CERN, a Ginevra, quando negli anni '80, Tim Berners-Lee e Robert Cailliau inziarono a lavorare al progetto ENQUIRE al fine di realizzare il modo per accedere a databases localizzati su diversi server al CERN. Attorno al dicembre 1990, Berners-Lee disponeva dei due strumenti fondamentali per il web: un web editor e il primo server web. Con la pubblicazione del primo annuncio sul newsgroup alt.hypertext, il 6 agosto 1991, nacque il web.

Il web è basato sul modello organizzativo dell'ipertesto, ma l'intuizione fondamentale di Berners-Lee fu di legare il concetto di ipertesto a quello di un'architettura a rete distribuita, sviluppata da Arpanet sin dagli anni '60. Come conseguenza, egli sviluppò un sistema per l'identificazione univoca delle risorse nella rete, attraverso un uniform resource identifiers[6].

Il world wide web era non proprietario, rendendo possibile sviluppare server e client indipendentemente, senza restrizioni legate all'ottenimento di un'eventuale licenza. Il 30 aprile 1993 il CERN annunciò che il world wide web sarebbe stato a disposizione di chiunque.

[6] Uno **Uniform Resource Identifier** è una stringa che identifica univocamente una risorsa generica che può essere un indirizzo Web, un documento, un'immagine, un file, un servizio, un indirizzo di posta elettronica, ecc. L'URL è un URI, o più comunemente chiamato *indirizzo web**Specificata fonte non valida.***.

Il world wide web fu concepito come un'architettura aperta, basata su standards aperti, gratuitamente e liberamente disponibile per chiunque. Esso si compone di tre elementi: l'URL (Uniform Resource Locator) che localizza ogni pagina contenente l'informazione e dove può essere trovata; l'HTTP (Hyper Text Transfer Protocol) che specifica come browser e server si scambiano le informazioni; l'HTML (Hyper Text Mark-up Language) che è un metodo di codifica delle informazioni al fine del loro utilizzo su diversi dispositivi.

Al fine di gestire il web e l'evoluzione dei suoi tre componenti, venne creato nel 1994, su iniziativa di Berners-Lee, il World Wide Web Consortium (W3C), con sede al M.I.T., che assunse anche l'onere di rendere accessibile il web ai diversamente abili e garantire la libertà di accesso al web a tutti[7].

1.6 Il progetto GNU e l'Open Source Initiative: 1983-1998

La storia del Free Software è indubbiamente legata alla personalità di Richard Stallman, uno sviluppatore che, durante la prima metà degli anni '80, reagì coraggiosamente alla commercializzazione di Unix creando un nuovo sistema operativo, che garantiva a chiunque l'accesso al codice sorgente del software. Il progetto di Stallman rispondeva alla domanda di una larga parte della comunità scientifica che, fino a quel momento, aveva utilizzato e modificato il software senza limitazioni e che, con il software proprietario, si vedeva privata di questo diritto[8].

[7] Sito istituzionale del W3C http://www.w3.org/

[8] Interessante vedere come Stallman (2003) racconta il motivo determinate la sua scelta: «Quando cominciai a lavorare nel laboratorio di Intelligenza Artificiale del MIT [Massachusetts Institute of Technology] nel 1971, entrai a far parte di una comunità in cui ci si scambiavano i programmi,che esisteva già da molti anni. La condivisione del software non si limitava alla nostra comunità; è una

Come risposta a questa situazione, Stallman inaugurò nel 1984 il progetto GNU (acronimo ricorsivo di GNU's Not Unix) per la realizzazione di un nuovo sistema open, con la contemporanea creazione, nel 1985, del Free Software Foundation. La FSF può essere vista come l'intento di istituzionalizzare la filosofia hacker, e il concetto di "free" va ovviamente inteso come libero e non gratuito (Stallman, 1999).

Anche nel caso di GNU, l'obiettivo era la creazione di un sistema operativo che fosse adattabile a qualsiasi computer e quindi garantire la sua portabilità. Allo stesso tempo, la libera distribuzione del codice sorgente insieme al software rendeva possibile avviare un

---

cosa vecchia quanto i computer, proprio come condividere le ricette è antico come l'arte culinaria. Ma noi lo facevamo più di chiunque altro.
Il laboratorio di Intelligenza Artificiale usava un sistema operativo a partizione di tempo (timesharing) chiamato ITS (IncompatibleTimesharing System) che il gruppo di hacker del laboratorio aveva progettato e scritto in linguaggio assembler per il Digital PDP-10, uno dei grossi elaboratori di quel periodo. Come membro di questa comunità, hacker di sistema nel gruppo laboratorio, il mio compito era quello di migliorare il sistema. Non chiamavamo il nostro software "software libero", poiché questa espressione ancora non esisteva, ma proprio di questo si trattava. Ogni volta che persone di altre università o aziende volevano convertire il nostro programma per adattarlo al proprio sistema e utilizzarlo, gliene davamo volentieri il permesso. Se si notava qualcuno usare un programma sconosciuto e interessante, gli si poteva sempre chiedere di vederne il codice sorgente, in modo da poterlo leggere, modificare, o cannibalizzarne alcune parti per creare un nuovo programma.
La situazione cambiò drasticamente all'inizio degli anni '80, con la dissoluzione della comunità hacker del laboratorio d'Intelligenza Artificiale seguita dalla decisione della Digital di cessare la produzione del computer PDP-10. Nel 1981 la Symbolics, nata da una costola del laboratorio stesso, gli aveva sottratto quasi tutti gli hacker e l'esiguo gruppo rimasto fu incapace di sostenersi. Quando nel 1982 il laboratorio di Intelligenza Artificiale acquistò un nuovo PDP-10, i sistemisti decisero di utilizzare il sistema timesharing non libero dellaDigital piuttosto che ITS. Poco tempo dopo la Digital decise di cessare la produzione della serie PDP-10. La sua architettura, elegante e potente negli anni '60, non poteva essere estesa in modo naturale ai maggiori spazi di intervento che andavano materializzandosi negli anni '80. Questo stava a significare che quasi tutti i programmi che formavano ITS divenivano obsoleti. Ciò rappresentò l'ultimo chiodo conficcato nella bara di ITS; 15 anni di lavoro andati in fumo. I moderni elaboratori di quell'epoca, come il VAX o il 68020, avevano il proprio sistema operativo, ma nessuno di questi era libero: si doveva firmare un accordo di non-diffusione persino per ottenerne una copia eseguibile. Questo significava che il primo passo per usare un computer era promettere di negare aiuto al proprio vicino. Una comunità cooperante era vietata. La regola creata dai proprietari di software proprietario era: «se condividi il software col tuo vicino sei un pirata. Se vuoi modifiche, pregaci di farle».

processo di revisione distribuita di GNU, per il suo continuo aggiornamento da parte degli utenti stessi.

Per impedire che si verificasse quello che era già accaduto con Unix, che, alla fine degli anni '70, era ritornato sotto il controllo della AT&T dopo quasi un decennio durante il quale era stato continuamente migliorato gratuitamente dai suoi utenti, Stallman ideò una particolare licenza d'uso del software, la GNU Public License. Essa, insieme al progetto GNU, diede vita al fenomeno del Free Software.

Il progetto GNU iniziò nel 1983 con un messaggio che Stallman postò al newsgroup del Mit Lab[9], cercando aiuto, sia ai produttori di computer per l'ottenimento di elaboratori, sia agli sviluppatori per contributi in denaro e lavoro, al fine di creare un sistema operativo diverso da Unix da mantenere di dominio pubblico.

Al fine di impedire l'appropriazione da parte di terzi, Stallman elaborò la nuova licenza GNU Public Licence[10], definita anche copyleft, che garantiva quattro livelli di libertà:

- Libertà di eseguire il programma, per qualsiasi fine;
- Libertà di studiare come funziona il software, e di adattarlo ai propri bisogni, con libero accesso al codice sorgente;
- Libertà di distribuire copie per aiutare il proprio vicino;

[9] http://www.gnu.org/gnu/initial-announcement.html

[10] La versione autorizzata italiana della GNU GPL v. 3 può essere letta a questo link http://www.gnu.org/gnu/initial-announcement.html

- Libertà di migliorare il programma e obbligo di rilasciare pubblicamente le proprie modifiche, alle stesse condizioni a cui è stato ottenuto il programma originale, permettendo a terzi l'accesso al codice sorgente.

In estrema sintesi, la licenza rende obbligatoria la diffusione del codice sorgente con il codice oggetto, consentendo a chiunque la modifica del sistema operativo ma, qualora il software modificato venga utilizzato al di fuori dell'uso personale, obbliga l'utente che ha apportato le modifiche a divulgarle sotto licenza GNU/GPL. Quest'ultima disposizione della licenza determina la sua cd. "natura virale": tutte le derivazioni del programma possono essere distribuite solo sotto la stessa licenza. La natura virale della licenza è una caratteristica fondamentale, che impedisce a terzi che vogliano comportarsi da free riders l'incorporazione di parte del programma in un software proprietario, così come era avvenuto in passato per tutti i miglioramenti apportati in ambienti universitari ad Unix e altri programmi. In questo modo il software protetto dalla licenza GPL è "free", nell'accezione inglese inteso come libero, anche se non necessariamente gratuito: "free as free speech, not as a free beer".

Il concetto di open source software nacque durante un meeting tenuto nel febbraio 1998 a Palo Alto, California, a cui parteciparono diversi sviluppatori, tra cui Eric Raymond, riuniti in un team assunto dalla Netscape per elaborare una strategia di difesa del software Navigator rispetto alla crescente diffusione del browser Explorer della Microsoft.

In particolare la Netscape voleva individuare il miglior modo di aprire il codice sorgente del programma e l'idea degli sviluppatori fu quella di testare su Navigator il modello dello sviluppo aperto del software.

Tale modello era già stato testato in prima persona da Raymond con fetchmail, in cui il modello "bazaar", cioè condiviso e non gerarchizzato (in contrapposizione al modello proprietario della "cattedrale") era, a detta di Raymond, più efficiente nel realizzare un software il più possibile privo di difetti, in virtù del principio (o "legge di Linux") per cui "given enough eyeballs, all bugs are shallow" (Raymond, 1999) in aperta contraddizione con la legge di Brooks, per cui, invece, aggiungere sviluppatori ad un progetto in fase di implementazione ne rallenta in realtà lo sviluppo.

Raymond individuò l'attitudine antagonistica del movimento Free Software, legato alla natura virale della sua licenza, rispetto al mondo del software commerciale, e quindi si pensò di coniare un nuovo termine, l'open source, appunto, per indicare delle caratteristiche diverse che tale software avrebbe dovuto avere e che vennero definite ufficialmente con l'Open Source Definition[11].

---

[11] "Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria: 1. Free Redistribution: The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale. 2. Source Code: The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed. 3. Derived Works: The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software. 4. Integrity of The Author's Source Code:

La creazione del movimento Open Source avvenne ufficialmente tra il 1998 ed il 1999. Il 22 gennaio 1998 Netscape annunciò che avrebbe rilasciato il codice sorgente per Navigator e, nel successivo febbraio, il team di sviluppatori su cui Netscape aveva fatto affidamento per questo nuovo progetto, coniò il termine Open Source con successivo lancio del sito web del progetto.

Il concetto di open source era seguito ad un animato dibattito all'interno della comunità hacker in merito ai possibili legami del software libero/aperto con il mercato, ed in particolare alla possibilità che le società operanti nel settore del software potessero far proprie le modifiche apportate al software aperto, al fine della loro commercializzazione, mediante l'attribuzione di una licenza d'uso meno virale della GNU GPL o della LGPL.

Il 22 giugno 1998 IBM annunciò che avrebbe venduto e sostenuto Apache, un rinomato Web server open source, come parte della sua suite Websphere, mentre a luglio l'Economist

---

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software. 5. No Discrimination Against Persons or Groups: The license must not discriminate against any person or group of persons. 6. No Discrimination Against Fields of Endeavor: The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research. 7. Distribution of License: The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties. 8. License Must Not Be Specific to a Product: The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution. 9. License Must Not Restrict Other Software: The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software. 10. License Must Be Technology-Neutral: No provision of the license may be predicated on any individual technology or style of interface." Vedi Open Source Definition (http://www.opensource.org/docs/osd).

dedicò un editoriale a Linux, riportando i risultati positivi delle performances del software.

La combinazione tra Linux e i metodi open source cominciò ad assumere una propria sostenibilità anche da un punto di vista economico.

Il fenomeno cominiciò a destare le preoccupazioni del principale concorrente, la Microsoft, che in un memorandum, il famoso Halloween Document[12], individuava, per mezzo dei propri dirigenti, i possibili mezzi – legali e non – per contrastare il nascente movimento open source.

Fig. 1.3: FUD – Pubblicità Microsoft su C't (Ottobre 2000)

Trad. it.: "Un sistema operativo open non ha solo benefici"
Fonte: Wynants & Cornelis (2005)

[12] Il memorandum era circolato nel mese di agosto 1998 ed era stato elaborato dai dirigenti della Microsoft. Divenne involontariamente pubblico il 1° novembre 1998, scatenando polemiche sui metodi non proprio corretti che Microsoft intendeva utilizzare per fronteggiare Linux. Vedasi http://www.catb.org/esr/halloween/index.html.

La strategia adottata dalla Microsoft venne definita FUD, ovvero "fear, uncertainty and dubt"[13]. Nonostante questa strategia, la quota di mercato di Linux alla fine del 1998 era aumentata del 212%.

Il 1999 fu un altro anno di successi per il software open source: a gennaio HP ed SGI annunciarono che avrebbero utilizzato Linux sulle proprie macchine, mentre a febbraio fu la IBM ad annunciare il ricorso a Linux sui propri pc, a supporto di Lotus, nonché la sua partecipazione a Red Hat.

Nel mese di marzo la Apple rilasciò un nuovo sistema operativo aperto, Darwin, che divenne il cuore del nuovo sistema operativo Mac OS X, mostrando la sua nuova apertura al software open source[14].

Alla fine degli anni '90 quindi il termine open source divenne la definizione comune di un movimento che non raggruppava più solo hackers, ma molte grandi software companies.

1.7 L'era di Linux: 1991-2001

Nei primi anni '90, Stallman e la FSF non avevano ancora completato il kernel. Il kernel è il cuore di ogni sistema operativo: la sua funzione è dare ai programmi l'accesso a risorse come la memoria del pc, la ram, la rete etc. Il completamento di GNU come sistema operativo fu possibile grazie al kernel Linux.

Linus Torvalds era uno studente di informatica all'Università di Helsinki, agli inizi degli anni '90, che aveva avuto modo di conoscere

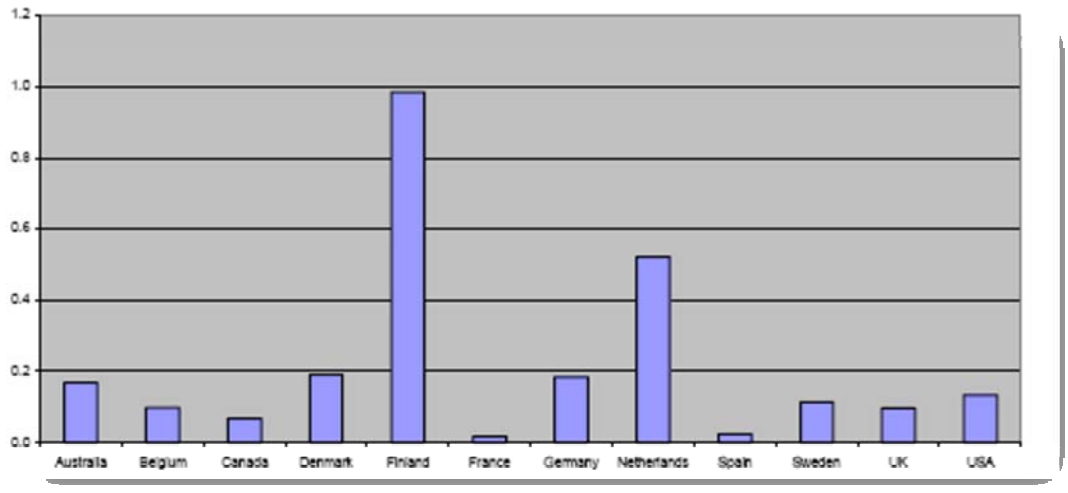
[13] Un esempio fu la rivendicazione ad opera di SCO di proprio copyright su alcune parti di Linux, con conseguente citazione in giudizio di Torvalds: si sospettò subito che dietro questa rivendicazione ci fosse la Microsoft, cosa che venne confermata nel 2003 quando il venture capital BayStar ammise che Microsof aveva finanziato il loro investimento in SCO, chiedendo in cambio l'uso del copyright contro Linux (Wynants M., J. Cornelis , 2005 p. 53).

[14] Tutt'ora, ad ogni nuova release di Mac OS X, la Apple pubblica il suo codice sorgente qui: http://developer.apple.com/opensource/index.html.

il sistema operativo Minix, un clone libero di Unix, creato da Andrew Tanenbaum. Nel 1991 aveva poi seguito un ciclo di lezioni tenute al Politecnico di Helsinki da Richard Stallman. A quel tempo Stallman, che già da alcuni anni lavorava con la Free Software Foundation al nuovo sistema operativo libero GNU, si trovava ad affrontare il problema della realizzazione del nucleo di ogni sistema operativo, ovvero il kernel[15]. Data l'importanza del problema, diversi sviluppatori, tra cui lo stesso Torvalds, erano impegnati a ricercare una soluzione efficiente al problema.

L'intuizione vincente di Torvalds fu il ricorso al nascente web al fine di creare una rete di sviluppatori, dislocata potenzialmente in qualsiasi parte del mondo, che lavorasse al progetto, iniziata con un messaggio di richiesta di collaborazione postato il 25 agosto 1991.

Fig. 1.4: Provenienza dei primi sviluppatori chiave di Linux



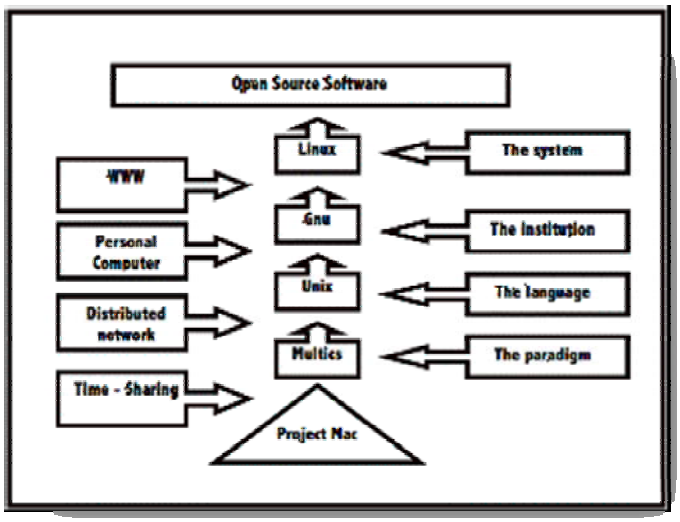
Distribuzione per milione di abitanti in ciascuno Stato delle persone coinvolte in base al Credits File del Marzo 1994

Fonte: Tuomi (2000)

[15] Si tratta di un software avente il compito di fornire ai processi in esecuzione sull'elaboratore un accesso sicuro e controllato all'hardware. Dato che possono esserne eseguiti simultaneamente più di uno, il kernel ha anche la responsabilità di assegnare una porzione di tempo-macchina e di accesso all'hardware a ciascun programma (multitasking) (fonte: Wikipedia 2008)

Se è vero che la parte più rilevante del lavoro di creazione di un nuovo software, ovvero la scrittura del codice, richiede un notevole sforzo da parte di un numero non eccessivo di sviluppatori, è anche vero che le successive fasi, ovvero la revisione del codice e il test sono compiti che possono essere svolte da un numero anche rilevante di persone, che si limitano a riportare il feedback tecnico derivante dall'uso del software, al gruppo di sviluppatori, attraverso il ricorso ad efficienti protocolli di comunicazione.

Fig. 1.5: Gli elementi fondanti del FLOSS



Fonte: Benussi (2006).

Dall'evoluzione del FLOSS è possibile trarre alcune conclusioni (Benussi, 2006):

1) Il fenomeno open source non è sostanzialmente nuovo ma è un'evoluzione path-dependant di una lunga tradizione nella gestione della conoscenza tecnologica. In particolare, essa dipende da un modus operandi, seguiti dai primi pioneristici hacker che ricalca il modello aperto della scienza (David, 2004).

2) La crucialità del web nel consentire la realizzazione di una rete mondiale di sviluppatori che lavorassero ad un progetto comune, senza dover più scontare limiti geografici.

3) La nascita dell'open source come codificazione della cultura hacker, in particolare la creazione delle licenze open source in buona parte sulla pioneristica licenza GPL elaborata da Stallman per GNU.

4) Infine, il fatto che il FLOSS ha una propria storia autonoma, in continua evoluzione.

Capitolo 2

2.1 Il problema della tutela giuridica del software tra brevetto e copyright

La tutela giuridica del software in ambito nazionale e internazionale è sempre stata argomento controverso tanto dal punto di vista giuridico, che da quello economico e addirittura etico/filosofico.

Dal punto di vista giuridico, superato un approccio iniziale basato sull'applicazione dei principi generali in materia di responsabilità civile (contrattuale, extracontrattuale, concorrenza sleale, segreto industriale) ed inquadrato il software nella categoria dei beni immateriali, le possibili forme di tutela hanno sostanzialmente oscillato fra la protezione tramite diritto d'autore (copyright) e la tutela tramite brevetto (patent), istituti appartenenti alla categoria dei diritti di proprietà intellettuale, strumenti attraverso i quali "il bene pubblico idea diviene per un certo periodo di tempo proprietà privata del creatore".

Sulla scelta fra l'una o l'altra forma di protezione hanno esercitato notevoli condizionamenti profili di carattere tecnico quali, l'affermazione o negazione del carattere meramente intellettuale dei programmi in quanto tali e la possibilità di configurare o meno un profilo espressivo dei programmi diverso da quello propriamente contenutistico. Una certa rilevanza hanno senza dubbio esercitato aspetti di carattere politico-giuridico stante le diverse filosofie di fondo sottese alle due forme di tutela che implicano differenti modi di acquisizione dei diritti, differente durata del diritto, differenti

strumenti di difesa da eventuali violazioni (il diritto d'autore protegge la forma dell'espressione creativa a prescindere dal contenuto in essa racchiuso, mentre all'opposto il brevetto permette lo sfruttamento della creazione riguardo al suo contenuto, i diritti sulle invenzioni industriali nascono al momento del conseguimento del brevetto, il diritto d'autore al momento stesso della creazione dell'opera).

2.2 La scelta iniziale a favore del copyright: l'esperienza degli Stati Uniti

Sin dalla metà degli anni ottanta in ambito internazionale il copyright ha iniziato ad essere riconosciuto come principale mezzo di protezione del software. In particolare negli Stati Uniti con l'emanazione del Computer Software Copyright Act del 1980 il congresso americano ha accolto completamente le raccomandazioni della CONTU (The National Commission on New Technological Uses of Copyrighted Works) pubblicate nel 1978 dopo tre anni di studi sulle proposte di modifica della legge sul diritto d'autore, volte:

- a rendere esplicito che i programmi nella misura in cui sono originali, sono tutelati dalla legge sul diritto d'autore;
- estendere la protezione a tutti gli usi dei programmi non proibendo solo la riproduzione ma anche l'utilizzazione;
- assicurare al legittimo possessore di copie di programmi la facoltà di usarle o adattarle per uso proprio.

Tuttavia nell'ordinamento statunitense, accanto alla tutela tramite copyright, si è presto affiancata una tutela tramite brevetto e a partire dalla celebre decisione State Street Bank del 1998[16], ogni

[16] State Street Bank & Trust Co. v Signature Financial Group Inc., 149 F. 3d 1374 (1998).

remora verso la brevettazione di business methods appare essere stata accantonata anche perché nel sistema statunitense, difformemente da quello introdotto dalla Convenzione sul Brevetto Europeo (CBE) non esistono divieti legislativi di brevettare alcune invenzioni in dipendenza del loro oggetto: la legge stabilisce infatti solo quali siano i requisiti positivi di brevettazione.

2.3 La scelta comunitaria: la Direttiva n. 91/250 CE

In Europa una svolta decisiva si è avuta con l'approvazione della Direttiva n. 91/250 CEE relativa alla tutela giuridica dei programmi per elaboratore, atto emesso dopo un lungo e serrato dibattito e basato sui seguenti principi cardine:

- i programmi quali opere letterarie e purché originali sono tutelati in base alle leggi sul diritto d'autore;
- la tutela non si estende ai principi, alla logica, agli algoritmi e al linguaggio di programmazione nonché alle cosiddette interfacce qualora costituiti da un complesso di idee e principi;
- i diritti spettano a chi ha creato il programma:se questo è realizzato nel corso di lavoro subordinato o di un contratto d'opera essi spettano al datore di lavoro o al committente;
- i diritti esclusivi comprendono: la riproduzione in qualsiasi forma, compresi caricamento, visualizzazione, esecuzione, trasmissione e memorizzazione; l'adattamento; la distribuzione sotto qualsiasi veste giuridica a meno che non si sia venduto il programma;
- sono violazioni dei diritti esclusivi il possesso e il commercio di copie illecite da parte di chi sia al corrente o abbia motivo di

sapere che si tratta di copie illecite. Violazioni sono pure il commercio e il possesso di prodotti che facilitano la rimozione dei dispositivi tecnici di protezione del programma.

Punto particolarmente controverso oggetto di serrato dibattito sul quale la Direttiva ha preso posizione è quello della decompilazione o reverse engineering di un programma per elaboratore, e cioè il processo e le tecniche di trasformazione di un programma da codice oggetto in codice sorgente (procedimento che può richiedere una o più copie o adattamenti del programma originale).

La Direttiva consente il procedimento allorché sia indispensabile per ottenere le informazioni necessarie e consentire l'interoperabilità di un programma per elaboratore creato autonomamente con altri programmi, a condizione che dette attività siano eseguite dal licenziatario o da altra persona che abbia il diritto di utilizzare la copia del programma o per loro conto da persona abilitata a tal fine.

2.4 La disciplina italiana: il D.Lgs. 29.12.1992 n. 518 e le sue applicazioni.

In Italia il D.L.vo del 29 dicembre 1992, n. 518 che, in sede di recepimento della Direttiva n. 91/250/CEE, ha modificato la L. n. 633 del 1941 estendendo la tutela propria del diritto d'autore ai programmi per elaboratore ha stabilito: "In particolare sono comprese nella protezione:(omissis) I programmi per elaboratore, in qualsiasi forma espressi purché originali quale risultato di creazione intellettuale dell'autore. Restano esclusi dalla tutela accordata dalla presente legge le idee e i principi che stanno alla base di qualsiasi elemento di un programma, compresi quelli alla base delle sue interfacce. Il termine

programma comprende anche il materiale preparatorio per la progettazione del programma stesso".

Gli artt. 64 ter e 64 quater della legge sul diritto d'autore stabiliscono quattro casi in cui non si può impedire all'utilizzatore del programma di realizzare copia dell'opera, anche in assenza dell'autorizzazione del titolare del programma:

- copia necessaria all'uso del programma;
- copia effettuata per lo studio del programma;
- copia di riserva;
- copia per decompilare il programma per ottenere l'interoperabilità con altri programmi

2.5 La convenzione di Monaco del 1973 e la giurisprudenza della CBE

Sulla scia di un approccio giuridico contrario alla brevettabilità del software in quanto tale si è mossa anche la Convenzione sul brevetto europeo del 5 ottobre 1973 meglio nota come convenzione di Monaco. Nel testo della Convenzione alla previsione, contenuta all'art. 52 secondo cui un'invenzione per essere brevettabile deve presentare i tradizionali requisiti di novità, attività inventiva e idoneità ad un'applicazione industriale si accompagna l'elenco non tassativo, contenuto all'art. 52 dei trovati che non sono considerati come invenzioni in ragione del loro oggetto, fra i quali sono compresi i metodi matematici, i piani e metodi per attività commerciali e intellettuali, i programmi per elaboratori, le presentazioni di informazioni. Tuttavia, l'art. 52 stabilisce che la brevettabilità è

esclusa soltanto ove la domanda di brevetto concerna quei ritrovati considerati in quanto tali.

Stante la notevole dose di ambiguità delle norme, è spettato alla giurisprudenza della Commissione di ricorso dell'Ufficio Europeo Brevetti prendere posizione in ordine alla brevettabilità o meno delle invenzioni attuate per mezzo di elaboratori.

Fin dalla seconda metà degli anni '80, la Commissione di ricorso ha stabilito che invenzioni relative a metodi commerciali realizzati attraverso un software sono brevettabili solo ove esse apportino un contributo tecnico e ciò per una serie di considerazioni che, come notato hanno anticipato di circa quindici anni le attuali proposte della Commisione CE in materia di brevettabilità del software.

In due decisioni rese nel 1999, entrambe relative a domande di brevetto presentate da IBM, la Commissione di ricorso ha anzitutto ricordato che, quantunque il carattere tecnico non sia espressamente indicato dalla CBE fra le condizioni generali di brevettabilità, in alcune norme di essa, quali ad esempio gli artt. 27 e 29, ha stabilito che un'invenzione deve avere un carattere tecnico per dar luogo ad una valida brevettazione, ed ha poi confermato che un'invenzione deve mostrare di fornire un contributo tecnico allo stato dell'arte per essere brevettabile.

Questo principio è stato ribadito in una decisione successiva, nella quale è stato affermato che un metodo di gestione di un fondo pensione non è brevettabile per l'assenza di carattere tecnico, mentre lo è l'apparato che consente di attuarlo.

Di estremo interesse ai fini di una corretta lettura della proposta di direttiva della Commissione CE del 2002, sono i risultati uno studio

comparativo pubblicato nel 2000 e realizzato congiuntamente dagli Uffici brevetti europeo, statunitense e giapponese. In detto studio, il Presidente dell'UEB ha chiarito come i brevetti relativi a business methods sarebbero stati considerati dall'Ufficio classificando le rivendicazioni relative a business methods in tre categorie generali: (1) rivendicazioni "astratte" di un business method; (2) rivendicazioni che includono l'uso di un computer per realizzare almeno alcuni passaggi di un business method; (3) rivendicazioni che includono l'uso di qualche altro apparato diverso da un computer per realizzare almeno alcuni passaggi di un business method, come ad esempio telefoni cellulari.

Mentre le rivendicazioni del primo tipo avrebbero continuato ad essere escluse dalla brevettazione, quelle degli altri due tipi sarebbero state valutate sulla scorta dei principi elaborati dalla Commissione di ricorso, assumendo rilievo decisivo ai fini della brevettabilità il fatto che esse riguardino sistemi che apportano un contributo tecnico. Questo principio è stato ribadito con maggiore chiarezza nell'ultima versione delle Guidelines dell'UEB ove, dopo l'affermazione secondo cui uno schema per organizzare un'operazione commerciale non è brevettabile, si legge che tuttavia, se l'oggetto rivendicato individua un apparato o un procedimento tecnico per realizzare almeno alcune parti dello schema, quello schema e l'apparato o il procedimento devono essere esaminati come un tutt'uno. In particolare, se la rivendicazione indica computer, reti di computer o altri apparati programmabili, o un programma ad essi relativo, per realizzare almeno alcuni passaggi di uno schema, essa deve essere esaminata come un'invenzione attuata per mezzo di elaboratori elettronici.

Alle invenzioni attuate per mezzo di elaboratori elettronici è dedicata un'ampia parte delle Guidelines che dopo aver ricordato che l'esecuzione di un programma determina sempre degli effetti fisici, ad esempio di tipo elettrico, precisano, richiamando la decisione T 1173/97, che quei normali effetti fisici non sono, in sé considerati, sufficienti ad attribuire carattere tecnico a un programma per computer. Tuttavia, se un programma è in grado, quando viene fatto girare su un computer, di produrre un effetto tecnico ulteriore, che va oltre quei normali effetti fisici, non è escluso dalla brevettazione.

2.6 La funzione del brevetto: breve excursus teorico

In base al modello offerto da Teece (1986) esistono tre elementi fondamentali che consentono all'impresa di ottenere profitto dall'innovazione:

- il regime di appropriabilità (*appropriability regime*)
- gli *assets* complementari (*complementary assets*), che comprendono risorse che, unitamente all'innovazione, sono indispensabili per la sua commercializzazione (tecnologie complementari, marketing, assistenza post-vendita etc.)
- il paradigma dominante (*dominant design paradigm*) utilizzato nella fase detta appunto paradigmatica, che rappresenta lo schema consolidato di sfruttamento dell'innovazione e spesso corrisponde ad un processo produttivo.

Il regime di appropriabilità si riferisce ai fattori ambientali (escluse le imprese e la struttura del mercato) che governano l'abilità di un innovatore di catturare i profitti generati dall'innovazione; le

dimensioni più importanti di questo regime sono la natura della tecnologia e l'efficacia del sistema legale di protezione. La natura della tecnologia può contenere l'imitabilità di un prodotto a seconda che sia tacita piuttosto che codificata, mentre, per quanto riguarda i possibili strumenti di tutela dell'innovazione, uno di questi è il brevetto.

Secondo Mazzoleni & Nelson (1998) è possibile individuare nella letteratura economica quattro diverse teorie riguardo alla funzione economico-sociale che i brevetti possono rivestire:

- La previsione dell'ottenimento di un brevetto offre un incentivo alla realizzazione di invenzioni utili (*invention motivation theory*)
- L'ottenimento di un brevetto spinge il titolare a realizzare gli investimenti necessari a sviluppare e commercializzare l'invenzione (*induce commercialization theory*)
- I brevetti rappresentano un premio che la società offre agli individui che rivelano le loro invenzioni (*information disclosure theory*)
- Essi consentirebbero l'ordinata esplorazione di una nuova e ampia traiettoria tecnologica (*exploration control theory*)

Secondo la prima teoria, una classica funzione cui i brevetti assolvono consisterebbe nell'incentivare investimenti nella ricerca mediante il riconoscimento, attraverso i brevetti, dell'utilizzo esclusivo a favore del titolare del brevetto, che quindi avrebbe la possibilità di impedire ai propri competitors l'accesso all'innovazione introdotta. In questo modo, potrebbe far proprie delle rendite da monopolio derivanti dall'accesso esclusivo ad un'innovazione di

prodotto o di processo. A questa opportunità riconosciuta al titolare del brevetto si contrappone un costo sociale, rappresentato dal fatto che il monopolio (seppur temporaneo) impedisce, a meno che l'impresa titolare non conceda a tutti una licenza d'uso, che i competitors possano utilizzare la stessa innovazione, se non addirittura introdurre delle innovazioni incrementali, e ciò incide negativamente sul livello dei prezzi e, quindi, sul surplus del consumatore.

In realtà, diverse ricerche (Levin et al., 1987; Cohen et al., 2002) mettono in evidenza come, eccezion fatta per il settore farmaceutico, gli IPR non garantiscono assolutamente l'appropriabilità, da parte dell'impresa titolare della proprietà industriale, delle rendite derivanti dall'innovazione. Secondo Cohen et al. (2000), infatti, in base ai risultati raggiunti a seguito di un'indagine condotta nel 1994 su un campione di 1165 imprese statunitensi di medio-grande dimensione (fatturato superiore a 5 milioni di dollari) il brevetto è indicato, tra i diversi meccanismi di appropriazione dei profitti derivanti dall'innovazione (oltre al brevetto, il segreto industriale, il lead time, la presenza di investimenti complementari nella produzione e nel marketing, a cui si aggiungono altri strumenti legali) al penultimo posto in un'ideale graduatoria dell'efficacia di questi meccanismi, sia per le innovazioni di prodotto che per quelle di processo (Cohen et al., 2000 pp. 31-32). Questa ricerca risente del fatto che, basandosi su un campione di imprese di una certa dimensione, di fatto escluda le piccole imprese che, non avendo la possibilità di utilizzare gli investimenti in attività complementari di produzione e/o di marketing, devono giocoforza

puntare maggiormente sul brevetto come metodo di appropriazione, come del resto accade per diverse piccole imprese operanti nel settore biotech.

E' interessante rilevare come, nonostante la tutela mediante gli IPR sia considerata un mezzo di appropriazione assolutamente inefficace, negli USA, a partire dagli anni '80, si sia registrato un incremento del numero di brevetti esistenti di circa il 70% negli anni tra il 1983 ed il 1991. Kortum & Lerner (1998) hanno analizzato tre diverse ipotesi esplicative di questo fenomeno tutto americano:

– la *friendly court hypothesis*, in base alla quale l'incremento del numero di brevetti sarebbe imputabile ad un atteggiamento maggiormente favorevole ai brevetti da parte dell'ordinamento giurisdizionale americano, specie a seguito dell'introduzione della Court of Appeals of the Federal Circuit, nel 1982;

– la *fertile technology hypothesis*, che raggruppa diverse spiegazioni, che vanno dall'aumento del numero di imprese operanti in settori high tech, nel cui finanziamento un ruolo importantissimo è rivestito dalle società di venture capital, all'applicazione dell'IT al processo di ricerca, che determinerebbe l'aumento della sua produttività, al cambiamento delle modalità di gestione delle strutture che si occupano di R&S, soprattutto con l'indirizzamento verso attività più applicate;

– la *regulatory capture hypothesis*, che è un adattamento della teoria della cattura del policy maker, in base alla quale alcune lobbies avrebbero l'incentivo a spingere verso un rafforzamento della tutela degli IPR, per poterla utilizzare come barriera all'entrata.

Lo studio, testando le diverse ipotesi, giunge, per esclusione, ad affermare che alla base della crescita del numero di brevetti negli USA ci sarebbe il cambiamento nelle modalità di gestione delle attività di R&S, soprattutto con un riorientamento verso attività maggiormente applicate, che determina una maggiore produzione di innovazioni brevettabili.

L'incremento del numero di brevetti ottenuti è stato leggermente più marcato (circa 5 punti percentuali in più, secondo Lerner e Kortum, 1999) in alcuni settori high tech, come il settore delle biotecnologie e del software. Questi settori presentano alcune specificità che creano, in alcuni economisti, seri dubbi circa gli effetti positivi che la tutela degli IPR avrebbe come incentivo all'investimento in innovazione. Infatti questi settori, insieme ad altri, sono caratterizzati dalla presenza di tecnologie complesse. Secondo Kingston (2001), che cita Roycroft e Kash (1999), si definisce tecnologia complessa *a process or product that cannot be understood in full detail by an individual expert sufficiently to communicate all details of the process or product across time and distance to other experts. (A simple product or process is one that can be understood or communicated by one individual)*. Tali tecnologie complesse rivestono un'importanza economica crescente, tanto che, tra il 1970 ed il 1995 la percentuale dei 30 beni esportati di maggior valore, rappresentati da tecnologie complesse, è passata dal 53 al 78%. In settori basati su tecnologie complesse, l'esistenza di brevetti è intrinsecamente svantaggiosa per l'innovazione, in quanto ciascuna impresa possiede diversi brevetti sui diversi componenti di una siffatta tecnologia e, se esse non riescono a formare un pool o a rilasciarsi licenze d'uso

incrociate, l'innovazione nel settore può essere rallentata, se non resa addirittura impossibile, a causa della *tragedy of the anticommons* (Heller & Eisenberg, 1998).

Tra i possibili motivi che impediscono la formazione di pool vi è, soprattutto in passato, una certa diffidenza da parte delle autorità antitrust, compresa quella europea, che hanno visto in essi delle intese con finalità anticoncorrenziali. Solo nel corso del 2004 è stato approvato un regolamento (il Reg. CE 772/2004) che stabilisce i requisiti dei pool affinché rientri tra le tipologie di intese esenti dal divieto ex art. 81 Trattato di Roma, esenzione prevista per gli accordi di trasferimento della tecnologia tra imprese (per un'analisi delle differenze tra le Guidelines dell'Antitrust Usa e quelle sancite dal reg. CE cfr. Gilbert, 2004).

A differenza di quanto ritenuto dalle autorità antitrust per diversi decenni, la creazione di pool nei settori basati su tecnologie complesse ha effetti decisamente positivi sulla concorrenza. Infatti, in assenza di pool cui un'impresa può accedere mediante il pagamento di fees o conferendo un certo numero di innovazioni brevettate, essa si troverà costretta a brevettare il maggior numero possibile di innovazioni, anche incrementali, in modo da poterle usare come chips per poter accedere, mediante accordi di cross-licensing, alle innovazioni introdotte dai propri competitors. In una situazione simile, quindi, un'impresa si troverebbe sotto pressione, incentivata a brevettare tutto il brevettabile, per non vedersi precluso l'accesso a tecnologie indispensabili per la propria attività. In presenza di un pool, invece, verrebbe brevettato lo stretto indispensabile per poter continuare a fruire dell'accordo di trasferimento (Kingston, 2001).

Una seconda interpretazione del ruolo economico che un brevetto può rivestire viene offerta dalla teoria secondo la quale attraverso tale forma di tutela della proprietà industriale, riconosciuta nelle fasi iniziali dell'innovazione di prodotto o di processo, si incentiverebbe l'impresa ad investire nello sviluppo delle fasi successive necessarie per lo sfruttamento in-house dell'innovazione, oppure nella commercializzazione dell'innovazione. A parte i limiti del brevetto come strumento di appropriazione, messi in evidenza da diversi studi di cui sopra, può essere interessante rilevare come, al contrario, i diritti sulla proprietà industriale diventino, in alcuni contesti, degli strumenti ideali per il trasferimento delle tecnologie da parte delle imprese che non intendono attuare uno sfruttamento in-house dell'innovazione brevettata in precedenza. Infatti, in alcune situazioni contingenti si può rilevare l'esistenza di veri e propri mercati per la tecnologia, peraltro già esistenti, in alcuni settori, nel periodo a cavallo tra il XIX e il XX secolo (Lamoreaux & Sokoloff, 1997, 1999, 2002). Arora, Fosfuri, & Gambardella (2001) mettono in evidenza come la scelta di concedere licenze d'uso non sia, come spesso si crede, motivata dall'incapacità dell'impresa di sfruttare al proprio interno la tecnologia creata, o dalla scelta di imporla come uno standard de facto. Attualmente, infatti si sta rilevando una tendenza al trasferimento di tecnologie da parte delle imprese leader in alcuni settori (cfr. la tabella 7.2 pp.176-177, Arora et al., 2001). Alla base della decisione di concedere licenze d'uso sui propri brevetti ci sarebbe la valutazione di due effetti di segno opposto, derivanti da questa scelta, ovvero il *revenue effect* (cioè la possibilità di ottenere utili da tali accordi, grazie a fees o a royalties) e il *rent dissipation*

effect (cioè la possibilità, attraverso l'ottenimento via licenza della tecnologia dell'impresa leader, che i competitors erodano il suo premium price). E' ovvio che se l'impresa è l'unica proprietaria della tecnologia, non avrà nessun incentivo a concedere a nuovi competitors licenze d'uso, perché il rent dissipation effect dominerebbe completamente il revenue effect; la situazione cambia notevolmente se a possedere la tecnologia leader sono più incumbents, perché in questo caso il licenziante scarica una parte dell'effetto da dissipazione della rendita sulle altre imprese preesistenti. Per quanto riguarda il revenue effect, la sua portata dipende dai costi di transazione e dal potere contrattuale rispettivamente del licenziante e del licenziatario. In presenza di bassi costi di transazione e di un elevato potere contrattuale del licenziante, il revenue effect è elevato. Gli Autori elaborano poi un semplice modello (pp. 182-192, op. cit.) attraverso il quale essi mostrano la validità delle loro tesi.

Del resto, si può rilevare una crescente attenzione delle grandi imprese verso la gestione strategica del proprio portafoglio brevetti, che possono essere utilizzati, ad esempio, per decifrare le strategie di prodotto dei concorrenti, nonché il modo di bloccarli attraverso le patenti; conquistare un ingresso protetto da brevetti in mercati redditizi ma aspramente concorrenziali; acquisire diritti esclusivi su nuove tecnologie che permettono la leadership di mercato; aumentare l'efficacia di ricerca e sviluppo ed evitare i campi minati delle violazioni; scoprire possibili contraffattori e individuare nuove probabili fonti di reddito da licenze (Rivette & Kline, 2000, p. 27). Esistono diversi esempi di grandi imprese la cui crisi, in passato, è derivata da una cattiva gestione del proprio patrimonio in proprietà

industriale, e la cui rinascita si è fondata sull'adozione di un'accorta strategia brevettuale, come nel caso di Xerox, Texas Instrument etc. Emblematica, ad esempio, è stata la scelta di Xerox, nel 1979, di non registrare l'invenzione dell'interfaccia utente grafica (GUI, *graphical user interface*), che costituì più tardi la base dei sistemi operativi di Apple e Microsoft Windows. Se invece l'avesse fatto, Xerox, mediante la concessione di una licenza d'uso e fissando una royalty, ad esempio, dell'1%, avrebbe potuto ottenere, tra il 1984 ed il 1998, alla scadenza del brevetto, un ricavo da licenze di circa mezzo miliardo di dollari (Rivette & Kline, 2000, pp.88-89). Recentemente Xerox, attraverso la creazione di Xipo (Xerox Intellectual Property Operations), ha fissato l'obiettivo di aumentare notevolmente il ricavo delle licenze dagli 8,5 milioni di dollari nel 1997 – una cifra insufficiente a coprire il costo di mantenimento dei brevetti – ai 180 milioni nel 2002 (Rivette & Kline, 2000, p. 114)

Una delle distorsioni che la brevettazione, soprattutto qualora non vada a tutela dell'applicazione tecnica ma di un'idea è, secondo R. Nelson (2004), dovuto al problema che, mentre in teoria, esiste una netta distinzione tra scienza e tecnologia, nella realtà i confini tra esse sono confusi, e ciò fa sì che spesso si finisca col brevettare non tecnologie, bensì vere e proprie scoperte scientifiche. Secondo Nelson, questo problema di fondo non può essere risolto da una appropriata normativa sugli IPR; ciò nonostante, questa può essere utile nel limitare alcune situazioni decisamente negative, come la concessione di brevetti su scoperte che sono in realtà dei veri e propri fenomeni naturali: si dovrebbe in questo caso cercare di limitare il riconoscimento di patenti a quei risultati della ricerca che consistono

in trasformazioni sostanziali rispetto, ad esempio, ad un composto presente in natura. Un altro possibile obiettivo di una normativa sulla proprietà industriale che tuteli i *science commons* dovrebbe essere una più stringente definizione dell'utilità (nell'ordinamento giuridico italiano, il requisito dell'industrialità) e quindi una più adeguata dimostrazione del progresso derivante dall'innovazione verso la soluzione di un problema concreto. Il terzo obiettivo dovrebbe essere una limitazione dell'ampiezza di ciò che viene rivendicato nel brevetto (cd. *patent claim*), problema su cui, per altro, sono stati scritti diversi contributi (ad es. Gilbert & Shapiro, 1990).

Una terza possibile funzione del brevetto consiste nella diffusione delle informazioni relative all'esistenza di una nuova tecnologia e facilitarne quindi il trasferimento a terzi. Ad esempio, nel caso della pubblicazione di brevetti universitari, questa può fungere come strumento di pubblicità dei risultati dell'attività di ricerca; si stima che, prima dell'introduzione del Bayh-Dole Act, le università spendessero ingenti quantità di denaro per attrarre investimenti di privati nella ricerca, mentre oggi questa funzione sarebbe in parte assolta dalla pubblicazione dei brevetti. Non esistono però, allo stato attuale, degli studi che dimostrino l'efficacia di questa forma di pubblicità nell'attività di technology transfer dalle università verso le imprese (Nelson e Mazzoleni, 1998, p.278). E' poi interessante mettere in evidenza come le diverse modalità di disclosure dei brevetti possano agevolare R&D spillovers tra le diverse imprese. Cohen et al. (2002) confrontano le normative a tutela degli IPR degli Stati Uniti e del Giappone, mettendo in evidenza le diverse finalità che i policy makers perseguono nei due paesi. In Giappone prevarrebbe l'interesse,

perseguito dal MITI fin dal secondo dopoguerra, di facilitare la diffusione delle innovazioni tecnologiche tra le imprese, e non è un caso che si preveda l'obbligo della disclosure, ovvero della rivelazione del contenuto del brevetto, 18 mesi dopo la presentazione della domanda, e non una volta che il brevetto viene riconosciuto, come avviene nella totalità dei paesi industrializzati.

Questo fatto incrementa l'utilizzo, da parte delle imprese giapponesi, dei brevetti come fonte o canale d'informazione sulle attività di R&D dei concorrenti. Non è quindi sorprendente rilevare come l'85,4% delle imprese giapponesi del campione analizzato consideri i brevetti uno dei canali più importanti per acquisire questo tipo di informazioni, a fronte del 49,1% delle imprese statunitensi (Cohen et al., 2002, p. 1363).

Secondo la quarta ed ultima teoria citata da Mazzoleni e Nelson (1998), l'esistenza di un ampio brevetto su una invenzione che dà luogo ad una nuova traiettoria tecnologica renderebbe possibile lo sviluppo e lo sfruttamento delle innovazioni successive in una maniera ordinata. In assenza di un brevetto che, in un certo senso "controlli" le innovazioni successive, infatti, lo sviluppo della traiettoria tecnologica avverrebbe in maniera dispersiva, in quanto diverse imprese indirizzerebbero i propri sforzi verso lo stesso obiettivo, rischiando di ottenere delle duplicazioni di innovazioni già ottenute dai propri competitors, con uno spreco di risorse che avrebbero potuto essere indirizzate altrove. In realtà, questa interpretazione del ruolo dei brevetti solleva diversi dubbi, in quanto la concessione di un claim ampio ad un brevetto che apre una nuova prospettiva tecnologica può rivelarsi estremamente dannosa per il progresso in quelle che Merges

e Nelson (1990, cit. in Mazzoleni e Nelson, 1998, p. 280) definiscono *cumulative systems technologies*, in cui il progresso tecnologico dipende dalla possibilità di utilizzare un ampio numero di componenti già sviluppati, che corrispondono alle tecnologie complesse, così come definite da Kingston (2001, cfr. supra). Un altro caso in cui il riconoscimento di brevetti dalle ampie rivendicazioni possono condurre ad effetti negativi, nel lungo periodo, sul progresso tecnologico, è quello in cui l'innovazione iniziale sia ben lungi dall'avere un'applicazione pratica, che invece sarebbe raggiunta dalle innovazioni successive che ne derivano.

2.7 I brevetti nel software – L'esperienza degli U.S.A.

Il tema della brevettazione del software si è posto con particolare rivelanza negli Stati Uniti all'indomani della celeberrime sentenze della cause Diamond v. Diehr e Diamond v. Bradley, entrambe del 1981, nelle quali la Corte Suprema aveva ammesso la brevettabilità degli algoritmi, sottesi i software per pc.

Queste sentenze furono subito recepite sia dai tribunali che dall'Ufficio Brevetti USA (U.S. Trademark and Patent Office), e l'esempio vivido di questo atteggiamento è dato dalla causa che, a partire dal 1993, vide la Stac Electronics citare in giudizio la Microsoft per aver utilizzato, nel sistema operativo MS-Dos 6.0, un software di compressione dei dati, chiamato Double Space, causa che si concluse con la vittoria della Stac Electronics e il ritiro del prodotto da parte della Microsoft (Graham & Mowery, 2006).

In generale, nel periodo compreso tra il 1987 ed il 2003 si è assistito ad un aumento, negli U.S.A., del numero di brevetti sul software. Nel 1995 è poi intervenuta una modifica, grazie alla quale

la loro durata è passata dai 17 (decorrenti dalla data di richiesta) ai 20 anni (decorrenti dalla data di ottenimento). A questa si è aggiunta una modifica più recente in virtù della quale è obbligatorio pubblicare i brevetti decorsi 18 mesi dalla domanda di riconoscimento. Questa modifica normativa può aver disincentivato il ricorso al brevetto (Graham & Mowery, 2006).

In virtù anche delle considerazioni, riportate nel precedente paragrafo, circa l'efficacia del brevetto, questo non è l'unica forma di tutela del software, visto che si accompagna alle altre forme di tutela della proprietà intellettuale, come il diritto d'autore o il segreto industriale, molto più adatti a tutelare i programmi informatici, dati i tempi abbastanza lunghi legati all'iter di ottenimento della patente.

L'incremento del numero di brevetti nel periodo 1987-2003, nonostante i problemi sopra riportati, è stato notevole, e, in maniera inaspettata, ha riguardato in misura maggiore le imprese operanti nell'elettronica piuttosto che quelle operanti nel settore del software, il cui prodotto finale è il pacchetto applicativo.

Per le prime, l'analisi di Graham e Mowery (2006) ha evidenziato come, nel caso del campione di 12 imprese[17] operanti nei sistemi elettronici ed informatici (tra queste, la IBM) esse abbiano ottenuto circa il 21% dei brevetti sul software nel 1990, diventato il 28% nel 1994, per poi calare al 21-23% nel periodo 1998-2003.

In netto contrasto rispetto alle altre imprese del campione, il numero di brevetti sul software ottenuto dalla IBM si presenta in costante crescita in tutto il periodo 1987-2003.

[17] Le 12 imprese, oltre alla IBM, sono Intel, Hewlett-Packard, Motorola, National Semiconductor, NEC, Digital Equipment, Compaq Computers, Hitachi, Fujitsu, Texas Instruments e Toshiba.

Per quanto riguarda invece le imprese che operano nel settore del software, il trend è stato ugualmente di crescita, anche se la percentuale di brevetti sul software ad esse ascrivibili risulta essere minore rispetto alle imprese di cui sopra. Nel 1987 la percentuale di brevetti sul software ad esse ascrivibili era lo 0,06%, divenuto il 4,75% nel 2002, per poi ridursi al 4,13% nel 2003. Se si elimina la Microsoft dal campione, tali percentuali diventano lo 0,06% nel 1987, l'1,35% nel 2000, diventato l'1% nel 2003.

La spiegazione della crescita dei brevetti sul software nel periodo compreso tra il 1987 ed il 2003 è senza dubbio legata al trend positivo che ha riguardato, specie negli Stati Uniti, tutti i brevetti, con le spiegazioni offerte ad esempio da Kortum & Lerner (1998, cfr. il paragrafo precedente), ma si registra anche un aumento del peso dei brevetti sul software sul totale dei brevetti richiesti ed ottenuti, in quanto essi rappresentavano l'1,7% del totale nel 1987 per poi più che raddoppiarsi nel 1997, diventando il 3,9% del totale.

E' probabile il maggior ricorso, da parte delle imprese che producono programmi per elaboratori, ai brevetti come "merce di scambio" e quindi l'utilizzo della strategia di "defensive patenting" (Graham & Mowery, 2006) per raggiungere accordi di concessioni di licenze reciproche con i propri concorrenti.

Un trend interessante, inaugurato dalla IBM, che evidenzia il perseguimento di una strategia ancora non chiarissima all'esterno, riguarda la concessione di brevetti alle comunità open source. La prima impresa a farlo è stata, appunto, la IBM nel 2005 con la concessione di circa 500 brevetti, accompagnata dagli investimenti diretti in progetti open source. Apparentemente questa scelta

sembrerebbe andare nella direzione dell'adozione, da parte della IBM, di una strategia di open innovation.

2.8 IPR e Open Source - Le licenze d'uso

2.8.1 Caratteristiche comuni delle licenze os: l'Open Source Definition

Il modello di produzione proposto dal movimento Open Source si basa sui diritti di proprietà. Come Mc Gowan (2001) mette in evidenza, gli IPR sono una forza nascosta dietro il software open source.

Il movimento Open Source utilizza in modo molto particolare il diritto d'autore. Il diritto d'autore è un insieme di cinque diversi diritti, ciascuno dei quali separabile a seconda della scelta fatta dall'autore. Il titolare del diritto d'autore può riprodurre, modificare, distribuire, eseguire pubblicamente, e mostrare pubblicamente il materiale protetto. Le licenze Open Source scindono questi diritti e stabiliscono quale di questi è riconosciuto al licenziatario.

Ad un livello pratico, le licenze open source servono ad indebolire gli IPR (Lee, 1999); è quindi interessante comprendere perché un sistema si basi sui diritti sulla proprietà intellettuale al fine di indebolirli.

Come mette in evidenza Mc John (2000) le persone coinvolte nel movimento Open Source pensano al diritto d'autore come ad un male necessario che consente al movimento di vivere, tenendo il codice sorgente aperto. Grazie al diritto d'autore, il programmatore ha il potere di impedire un uso improprio del suo software e proibire che

venga fatto proprio da terzi in un software proprietario. Rendere il software di dominio pubblico non consente di avere questo potere.

Il software privo di qualsiasi tutela, di dominio pubblico, infatti, può essere utilizzato da chiunque in qualsiasi modo e, soprattutto, può essere sottratto al pool di risorse comuni da parte di chi lo ingloba in un software proprietario (Unix docet).

Il software open source si caratterizza per essere un bene collettivo prodotto in maniera privata (Von Hippel & Von Krogh, 2003): esso è infatti il risultato del lavoro di migliaia di sviluppatori e, allo stesso tempo, ha le caratteristiche tipiche dei beni comuni (o *commons*), ovvero non escludibilità e assenza di rivalità nel consumo.

I commons si caratterizzano per essere soggetti alla tragedy of the commons (Hardin, 1968), cioè ad un eccessivo sfruttamento da parte di attori che si comportano da free riders, anche se tale tragedia non riguarda i beni intangibili, ed in più, anche per i beni pubblici tangibili, è stato dimostrato (Ostrom, 1990) l'esistenza di meccanismi di governo dell'utilizzo delle risorse tali da garantire la sostenibilità nel tempo dei commons.

Il problema maggiore che affligge il software open source come commons non è la possibile presenza di free riders che utilizzano i programmi senza contribuirvi, ma piuttosto la sua possibile sottrazione, in caso di ridotta tutela, dal pool delle risorse comuni, ad opera di terzi che lo rendono parte di un software proprietario.

E' chiaro come invece, la particolare forma di tutela derivante dalle licenze open source, a loro modo una sorta di copyright, possa mettere al riparo autore e intera comunità di utilizzatori del software da questa evenienza.

L'Open Source Definition chiarisce in che modo il fascio di diritti del diritto d'autore viene modificato al fine di evitare la sottrazione dal dominio pubblico del software.

In prima approssimazione, coerentemente con Valimaki (2005) si può parlare di software open source in presenza di un programma distribuito sotto una licenza d'uso che consente:

- Un uso libero, cioè il divieto di imporre restrizioni ad esempio sull'uso commerciale, sugli utenti o sull'hardware su cui il software potrà essere impiegato;
- La sua distribuzione e la copia senza dover pagare royalties, con la conseguenza che, per le software companies, è precluso nello sfruttamento commerciale l'utilizzo di un business model basato sulla vendita delle licenze d'uso;
- La possibilità di modificarlo, senza pagamento di royalities, anche se spesso vengono imposte dalla licenza delle condizioni per la modifica, come ad esempio la divulgazione dei cambiamenti apportati;
- Un codice sorgente aperto e facilmente ottenibile (non necessariamente gratis), che rappresenta lo strumento fondamentale per poter apportare modifiche.

Nel dettaglio, la Open Source Definition richiede che le licenze d'uso del software, affinché possano definirsi come open source, devono soddisfare le seguenti condizioni.

1. ***Libera ridistribuzione:*** *la licenza non può limitare alcuno dal vendere o donare il software che ne è oggetto, come componente di una distribuzione aggregata, contenente programmi di varia origine. La licenza non può richiedere*

diritti o altri pagamenti a fronte di tali vendite. Il diritto d'autore, e le licenze d'uso standard del software, impediscono la redistribuzione. Al contrario, le licenze open source la consentono, anche a pagamento, che riguarda però il supporto su cui il programma viene venduto, oppure l'offerta di servizi aggiuntivi.

2. ***Codice sorgente:*** *il programma deve includere il codice sorgente e ne deve essere permessa la distribuzione sia come codice sorgente che in forma compilata*. Questa prescrizione racchiude l'essenza dell'Open Source: non solo il software deve poter essere distribuito liberamente, ma deve esserlo con il suo codice sorgente, al fine di consentire agli sviluppatori la creazione di lavori derivati.
3. ***Prodotti derivati:*** *La licenza deve permettere modifiche e prodotti derivati, e deve permetterne la distribuzione sotto le stesse condizioni della licenza del software originale*. Le opere derivate incrementano il volume dei programmi open source in circolazione, anche se l'ampiezza con cui questa prescrizione dell'OSD viene applicata può essere utilizzata come elemento per una prima tassonomia delle licenze open source, ed in particolare tra licenze copyleft e licenze non copyleft, ovvero se rispettivamente impongono o no degli obblighi reciproci a carico del licenziatario. Lo schema del copyleft, infatti, fa sì che una volta che il programma è ottenuto in licenza d'uso da uno sviluppatore, il programma che ne deriva debba essere distribuito con la stessa licenza d'uso.

4. ***Integrità del codice sorgente originario:*** *la licenza può impedire la distribuzione del codice sorgente in forma modificata, a patto che venga consentita la distribuzione dell'originale accompagnato da "patch", ovvero file che permettono di applicare modifiche automatiche al codice sorgente in fase di compilazione. La licenza deve esplicitamente permettere la distribuzione del software prodotto con un codice sorgente modificato. La licenza può richiedere che i prodotti derivati portino un nome o una versione diversa dal software originale.* Lo scopo di questa prescrizione è che, garantendo comunque la creazione di prodotti derivati, si dia comunque agli utenti la possibilità di sapere chi è responsabile del software e delle sue modifiche. La licenza può quindi comportare la distribuzione del codice sorgente mediante la combinazione del codice originale più le modifiche sotto forma di patch. In questo modo le modifiche "non ufficiali" possono essere distribuite pur mantenendole distinte da quello "originale".
5. ***Discriminazione contro persone o gruppi:*** *la licenza non deve discriminare alcuna persona o gruppo di persone*. Lo scopo di questa previsione è rendere possibile il coinvolgimento del maggior numero possibile di persone.
6. ***Discriminazione per campo di applicazione:*** *la licenza non deve impedire di far uso del programma in un ambito specifico. Ad esempio non si può impedire l'uso del programma in ambito commerciale o nell'ambito della ricerca genetica.* Lo scopo di questa clausola è impedire la creazione di "trappole" che limitino l'uso commerciale del software open source. Si evita

quindi la preclusione dell'uso e la partecipazione allo sviluppo del software open source da parte delle imprese

7. ***Distribuzione della licenza:*** *i diritti allegati a un programma devono essere applicabili a tutti coloro a cui il programma è redistribuito, senza che sia necessaria l'emissione di ulteriori licenze.* La finalità di questa prescrizione è che, mediante accordi di non diffusione, la licenza venga di fatto aggirata, consentendo la "chiusura" del software.
8. ***Specificità ad un prodotto:*** *i diritti allegati al programma non devono dipendere dall'essere il programma parte di una particolare distribuzione di software. Se il programma è estratto da quella distribuzione e usato o redistribuito secondo i termini della licenza del programma, tutti coloro che ricevono il programma dovranno avere gli stessi diritti che sono garantiti nel caso della distribuzione originale.* Con questa clausola si vuole vietare altre licenze trappola.
9. ***Vincoli su altro software:*** *La licenza non deve porre restrizioni su altro software distribuito insieme al software licenziato. Per esempio, la licenza non deve richiedere che tutti gli altri programmi distribuiti sugli stessi supporti siano software open source***.** Con questa previsione si vuole far sì che chi utilizza e distribuisce software open source insieme ad altro software proprietario possa liberamente farlo. E' importante mettere in evidenza che l'altro software distribuito insieme a quello open source non è un'opera derivata, altrimenti si ricadrebbe invece nella prescrizione definita dal n. 3, che, a seconda del posizionamento della licenza os lungo il continuum tra copyleft

e non copyleft, comporterebbe, rispettivamente, l'obbligo o la facoltà di distribuzione dell'opera derivata con la stessa licenza d'uso del programma originario. L'idea dell'esistenza di vincolo sull'altro software imposto dalle licenze open source, soprattutto la GPL, è un pregiudizio che, inizialmente, ha portato addirittura a parlare di una supposta "viralità" della GNU General Public License, di cui si tratterà più diffusamente nei prossimi paragrafi.

10. ***Neutralità rispetto alle tecnologie:*** *la licenza non deve contenere clausole che dipendano o si basino su particolari tecnologie o tipi di interfacce.* Con questa clausola si vuole far sì che l'utilizzo di un programma open source sia possibile a prescindere dal supporto utilizzato per la sua distribuzione; in particolare, si vuole evitare che, mediante strumenti come il "click" su un tasto per l'approvazione della licenza da parte dell'utente, si precluda l'utilizzo di tecnologie diverse dai siti FTP come i cd-rom o altri per l'installazione del programma.

2.8.2 La tassonomia delle licenze open source

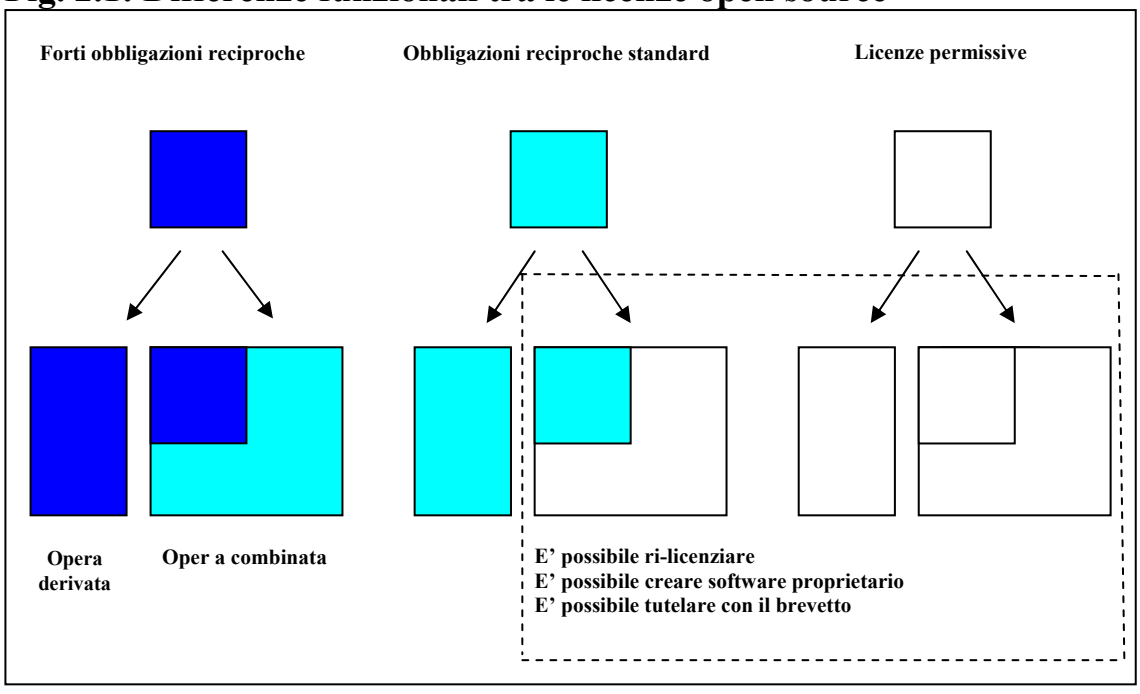
E' possibile classificare le diverse licenze esistenti seguendo diverse tassonomie (Valimaki, 2005). In prima approssimazione, si possono individuare licenze copyleft e licenze non copyleft, in cui la distinzione deriva dalla assimilabilità o meno della licenza alla GNU General Public License, ovvero se rispettivamente impongono o no degli obblighi reciproci a carico del licenziatario. Lo schema del copyleft, infatti, fa sì che una volta che il programma è ottenuto in licenza d'uso da uno sviluppatore, il programma che ne deriva debba essere distribuito con la stessa licenza d'uso.

Una prima classificazione è quindi quella "funzionale", in base alla quale è possibile distinguere licenze che impongono obbligazioni reciproche 1) forti 2) standard e 3) licenze permissive.

Da un punto di vista funzionale, le licenze open source possono essere distinte a seconda di come ciascuna di esse definisce il diritto di modificare il codice sorgente (ovvero la possibilità di creare opere derivate).

Si parla di licenze che impongono obbligazioni reciproche "standard" quando esse richiedono al licenziatario di distribuire le opere derivate sotto le stesse condizioni. In presenza però di un programma creato a partire dalla combinazione di uno open source con un altro, anche chiuso, tale obbligo non si estende ad esso. Ci si riferisce a queste licenze come *copyleft*.

Fig. 2.1: Differenze funzionali tra le licenze open source



Fonte: Valimaki (2005)

Nel caso, invece, delle licenze che impongono forti obbligazioni reciproche, i loro termini si estendono anche al software ottenuto dalla combinazione dello stesso con un altro chiuso, con il risultato che il prodotto ottenuto dovrà essere licenziato sotto queste licenze open source, come accade, ad esempio, per i software *embedded*.

Infine, nel caso delle licenze "permissive", è consentita la libera distribuzione, copia, e modifica del codice sorgente, così come la modifica della licenza stessa. Non esistono, nel loro caso, obbligazioni reciproche poste a carico del licenziatario.

Un'altra possibile classificazione delle licenze open source è di tipo storico, cioè legato al modo con cui le varie licenze hanno via via tenuto conto di ulteriori diritti o obbligazioni rispetto al copyright.

In base a questa classificazione è possibile distinguere tra licenze GPL, licenze accademiche, licenze "community" e licenze "corporate".

Le licenze GPL sono state inizialmente introdotte da Richard Stallman della Free Software Foundation e si caratterizzano per avere una forte impronta ideologica, che determina spesso la loro incompatibilità con altre licenze open source.

Le licenze accademiche fanno riferimento a tutte quelle licenze d'uso inizialmente utilizzate in ambito universitario, come ad esempio Berkeley, per lo sviluppo della infrastruttura di Internet e sono infatti state usate per i protocolli TCP/IP, per il sistema di naming dei domini (BIND) e per i principali software per i client di posta (Sendmail). Grazie a queste licenze molto permissive sono diventati degli standard di fatto.

La terza categoria è quella delle *community licenses*, che in genere hanno avuto origine dai maggiori progetti free source e hanno acquistato popolarità grazie alle implementazioni Unix e ad Internet. La più popolare è la Licenza Artistica originalmente distribuita con il linguaggio di programmazione Perl. Da un punto di vista giuridico si caratterizza per la frequente imprecisione nella definizione delle sue clausole che la rendono decisamente ambigua, anche se perfettamente in linea con la filosofia hacker di cui è permeata. Una licenza sicuramente più rigida è invece quella Apache, elaborata dalla Apache Software Foundation.

Fig. 2.2: Le licenze open source più utilizzate (2005, 2008)

Fonte: Valimaki (2005) (^) e Blackduck Software (*)

L'ultima categoria, quella delle licenze *corporate* si è sviluppata a decorrere dagli anni '90, quando la maggiore rilevanza assunta dal movimento open source nel panorama ICT ha spinto alcune software companies, già orientate verso il mondo FLOSS, a creare delle proprie licenze specifiche, come nel caso di Netscape per il suo browser (Mozilla), IBM (Common Public License), Apple (Apple Public Source License) e Sun (Sun Public License). Si differenziano dalle altre licenze os per la particolare attenzione riservata agli aspetti legati

ai brevetti, al copyright, ai marchi che sono assenti nelle altre tipologie.

Può essere interessante procedere all'analisi delle singole licenze utilizzando tre diversi elementi di analisi:

1. Opere derivate: verificare se e quali obbligazioni reciproche sono imposte dalla licenza;
2. Brevetti: come si pone la licenza rispetto ad essi;
3. Compatibilità: se e con quali altre licenze è compatibile.

Per quanto riguarda il concetto di opera derivata, nella normativa (e giurisprudenza) statunitense, esso fa riferimento, per quanto riguarda al software, all'esistenza di similarità sostanziali che comportano anche l'inclusione di porzioni del software originale (Valimaki, 2005). In ambito europeo non esiste un concetto esattamente corrispondente; nell'art. 4 della Direttiva 91/250/CEE del 14.05.1991 (Ubertazzi, Galli, & Sanna, 2003) viene, ad esempio, riservato in via assoluta al titolare del diritto d'autore sul programma originario "la traduzione, l'adattamento, l'adeguamento e ogni altra modifica di un programma per elaboratore e la riproduzione del programma che ne risulti, fatti salvi i diritti della persona che modifica il programma".

Si potrebbe, da ciò, ricavare che, in ambito comunitario, il concetto di opera derivata sia decisamente più ampio rispetto a quanto riscontrabile negli Stati Uniti, comprendendo tutte le tipologie di modifica al programma originario precedentemente elencate; ne consegue che, ad esempio, inserire alcune righe di codice sorgente del programma di un'altra persone potrebbe connotare come opera

derivata il software che si ottiene perché avrebbe alterato il software altrui, mentre potrebbe non essere definita come tale negli Stati Uniti perché non implicherebbe necessariamente il fatto che sia basato su di esso.

2.8.2.1 La GNU General Public License

La prima licenza copyleft, la GPL, definisce un'opera derivata come contenente in tutto o in parte o derivante da un software, stabilendo che ogni opera derivata distribuita o pubblicata debba essere concessa in licenza nei termini della stessa GPL[18]. Il fine di questa prescrizione della GPL è, come messo in evidenza in precedenza, evitare l'appropriazione in un software proprietario, e quindi la sottrazione dal pool di risorse comuni della comunità, di un software floss.

Da un altro punto di vista, questa previsione della GPL determina quella che l'Economist (1999) definisce come caratteristica virale[19] di questa licenza d'uso, facendo allontanare chi teme che la semplice interazione del proprio software con un altro licenziato con la GPL determini automaticamente l'obbligo di distribuirne il codice sorgente.

In realtà parlare di natura virale della licenza GPL non è corretto, in quanto il termine legale più corretto è quello che si riferisce a questa caratteristica della licenza GPL come persistenza o

[18] Dalla GNU GPL versione 2, punto 2: *"E' lecito modificare la propria copia o copie del Programma, o parte di esso, creando perciò un'opera basata sul programma, e copiare o distribuire tali modifiche secondo i termini del precedente comma 1, a patto che siano soddisfatte tutte le condizioni che seguono: (omissis) bisogna fare in modo che ogni opera distribuita o pubblicata, che in parte o nella sua totalità derivi dal Programma o da parti di esso, sia concessa nella sua interezza in licenza gratuita ad ogni terza parte, secondo i termini di questa Licenza"*

[19] Anche l'a.d. di Microsoft Steve Ballmer ha affermato nel 2001 "The way the license is written, if you use any open-source software, you have to make the rest of your software open source… [GPL'd software] is *a cancer* that attaches itself in an intellectual property sense to everything it touches" (cit. in Valimaki, 2005)

inheritance property. Ci sono però situazioni in cui, a seconda dell'interpretazione restrittiva o meno delle condizioni dettate dalla GPL, potrebbe ravvisarsi una sua ipotetica viralità.

Uno di questi esempi è rappresentato dai compilatori coperti da GPL. Si pensi al software come ad una scatola nera che, mediante una computazione (o, meglio, compilazione), elabora un input (le linee di codice) in un output (il programma eseguibile o codice oggetto). Si potrebbe concepire il software come l'opera derivata dal codice sorgente più il lavoro di compilazione che, solitamente, viene fatto da un compilatore[20]. Ci si potrebbe legittimamente domandare se anche l'autore del compilatore possa accampare diritti su tale output: se il compilatore si limita a trasformare del codice da un linguaggio ad un altro, allora certamente no, ma spesso succede che parti comuni del codice vengano copiate automaticamente dal compilatore. Il problema è reale, tant'è che GCC (GNU Compiler Collection) ha nella sua licenza GPL un'eccezione, che esclude dal concetto di opera derivata il software prodotto con un compilatore GPL.

Un altro problema si crea con le *librerie*[21] esterne al programma e magari contenute nel sistema operativo coperto da GPL con esso interagisce, che talvolta, con collegamenti statici o dinamici, vengono

[20] Un **compilatore** è un programma che traduce una serie di istruzioni scritte in un determinato linguaggio di programmazione (codice sorgente) in istruzioni di un altro linguaggio (codice oggetto) (fonte:Wikipedia, 2008).

[21] Una **libreria software** è un insieme di funzioni di uso comune, predisposte per essere collegate ad un programma software. Il collegamento può essere statico o dinamico, nel qual caso si parla di dynamic-link library. Lo scopo delle librerie software è quello di fornire una vasta collezione di funzioni di base pronte per l'uso, evitando al programmatore di dover scrivere ogni volta le stesse funzioni e facilitando le operazioni di manutenzione. Ad esempio molti linguaggi di programmazione hanno una libreria matematica, che offrono numerose funzioni come l'elevamento a potenza, il calcolo dei logaritmi e così via, e almeno altrettanti hanno funzioni di I/O (fonte: Wikipedia, 2008).

richiamate da un altro software. Ci si può legittimamente domandare se quest'ultimo debba essere assoggettato alla stessa licenza che copre le librerie del programma GPL da esso richiamate oppure no, e questo interrogativo si è posto concretamente nella causa che ha visto opporsi MySQL AB contro la Progress Software Corporation il cui software Gemini richiamava, con un collegamento statico, una parte del programma GPL MySQL (una "table handler"), senza che poi essa rispettasse i termini della licenza os, soprattutto senza la divulgazione del codice sorgente di Gemini (Majerus, 2003).

Un altro problema di possibile *viralità* della licenza GPL riguarda i *plug-in* che, secondo l'interpretazione fornita dalla Free Software Foundation andrebbero distribuiti, se richiamano funzioni di programmi GPL, sotto la stessa licenza di questi ultimi (cit. in Valimaki, 2005).

Un ultimo caso in cui si potrebbe riscontrare l'esistenza di un software derivato è in presenza di programmi non open source che utilizzano altri licenziati sotto la GPL in un rapporto tipo client-server oppure che fungono da interfacce grafiche per essi (ad esempio un interfaccia grafica per MySQL). In questo caso si utilizzerebbe il database di MySQL per far funzionare il programma, a fronte dell'utilizzo del software GPL che non viene modificato nel suo codice sorgente o oggetto. Dato che i due programmi interagirebbero insieme, secondo la definizione data dalla licenza GPL, anche la GUI dovrebbe essere distribuita sotto i suoi stessi termini; si hanno però dubbi sull'esistenza di un'opera derivata secondo l'accezione giuridica.

La *persistenza* riguarda unicamente le opere derivate ed in più, se parti identificabili di esse non sono derivate dal programma GPL, e possono ragionevolmente essere considerate indipendenti e separate, allora la licenza GPL e i suoi termini non si applicano se sono distribuite separatamente. Di conseguenza, la distribuzione sullo stesso supporto di programmi coperti da licenza GPL (es Linux) insieme a programmi che interagiscono con lo stesso (es. applicativi) non comporta l'obbligo di distribuirli tutti alle condizioni del copyleft.

Anche se, alla luce degli esempi sopra riportati, esistono situazioni in cui oggettivamente si potrebbe riscontrare questa supposta *viralità* della licenza pubblica generica, sono gli stessi sviluppatori di programmi GPL a spiegare come è possibile evitare di dover sottostare alla forte reciprocità imposta alle opere derivate, fatto questo che denota l'assenza dell'intenzione di ricondurre software proprietari sotto il dominio open source. La *inheritance property* si crea solo in riferimento alle opere derivate (con tutti i problemi di definizione giuridica sopra riportati) e non ai programmi, ad esempio, distribuiti insieme ad altri GPL sullo stesso supporto.

Quest'ultima caratteristica è specificamente definita nella OSD (punto 9 della OSD, cfr *supra)*, laddove si stabilisce che la licenza non deve porre restrizioni su altro software distribuito insieme al software licenziato. Per esempio, la licenza non deve richiedere che tutti gli altri programmi distribuiti sugli stessi supporti siano software open source, e questa è una previsione finalizzata a non limitare il diritto del distributore di scegliere quali software inserire in una data distribuzione.

2.8.2.2 La GNU Lesser General Public License

Questa licenza (in italiano tradotta come licenza pubblica generica attenuata) impone una reciprocità standard a carico dell'utente che impiega il software per creare un'opera derivata; ciò significa che solo le modifiche dirette al programma coperto da questa licenza comporta l'obbligo di distribuire quanto ottenuto con le stesse condizioni, mentre questo obbligo non sussiste nel caso di mero utilizzo del software in combinazione con altri, magari di tipo proprietario.

E' un tipo di licenza specificamente elaborato per le librerie informatiche, che possono essere utilizzate anche con un software proprietario, ad esso collegate. E' importante, a tal fine, che ci sia un link che consenta di mantenere le librerie distinte e che il loro codice sorgente sia reso comunque disponibile insieme alle istruzioni su come collegarle al programma originario.

2.8.2.3 La Mozilla Public License

Questa licenza ha avuto origine nel 1998 per consentire la diffusione del codice sorgente del browser Navigator di Netscape e può essere considerata anche come la prima licenza "corporate" mai creata. Impone delle condizioni di reciprocità "standard", in quanto l'obbligo di utilizzarla è a carico di chi apporta modifiche ad un programma coperto dalla licenza pubblica di Mozilla e vuole poi distribuirle, ma non a carico di chi impiega lo stesso in un'opera derivata, per cui quest'ultima può essere licenziata secondo i termini preferiti dall'autore, mentre rimane l'obbligo di distribuire il

programma coperto dalla MPL secondo i termini di quest'ultima[22]. Quest'ultima previsione determina l'incompatibilità di questa licenza con la licenza pubblica generica.

La licenza pubblica Mozilla ha poi una clausola specificamente destinata ai brevetti, per cui coloro che contribuiscono al software mediante porzioni di codice da loro elaborati e protetti da brevetti, garantiscono agli altri utenti una licenza d'uso illimitata sugli stessi. Se invece gli autori del programma venissero citati in giudizio da terzi per violazione di brevetti, allora anche le licenze d'uso concesse dagli autori sui loro contributi cesseranno nei confronti di chi ha agito in giudizio (cd. *termination clause*, n. 8 della MPL versione 1.1). A questo punto, l'attore in giudizio ha di fronte a sé due alternative, tra cui scegliere entro 60 giorni dalla presentazione della citazione: ritirarla, oppure pagare le licenze per l'uso eventualmente fatto del programma coperto dalla MPL.

2.8.2.4 La Berkeley Software Distribution

Questa licenza deve il suo nome al fatto che fu utilizzata per la prima volta per distribuire la versione della Berkeley Software Distribution di Unix, una versione libera di questo sistema operativo creata all'università di Berkeley. E' una licenza permissiva, che permette ma non obbliga chi apporta modifiche al suo codice sorgente o lo utilizza in opere derivate a mantenere tale codice aperto; non si impone neanche la distribuzione del codice sorgente, in quanto le modifiche o le opere derivate possono essere distribuite anche sotto forma di codice binario. Ciò però non significa che il codice distribuito con

[22] La clausola 3.7 della MPL v. 1.1 letteralmente stabilisce che "You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code" (Mozilla Foundation)

questa licenza sia di dominio pubblico, e questo sia perché riconosce agli autori i contributi effettuati, e sia perché tutela la reputazione degli stessi impedendo l'utilizzo dei loro nomi nel caso di opere derivate create da terzi, una tutela tipica del copyright (cd *endorsement clause*) . Queste condizioni rendono la BSD compatibile con qualsiasi altra licenza.

2.8.2.5 La licenza MIT

Questa licenza è molto simile alla BSD, con la differenza che non offre agli autori la stessa tutela offerta dall'altra nell'esatta attribuzione dei contributi nel caso di opere derivate (la cd. *endorsement clause*).

2.8.2.6 La licenza Apache

E' molto simile alla licenza BSD, con l'unica differenza, nella versione 2.0, che nel caso di opere derivate, essa consente al programmatore di scegliere qualsiasi licenza sotto cui distribuire il software, libertà questa stabilita nell'ultimo capoverso della clausola n. 4 della licenza:

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License (Apache Software Foundation, 2004). Per tale motivo è incompatibile con la licenza pubblica generica.

La licenza Apache prevede prescrizioni specifiche in relazione ai brevetti, stabilendo che chi contribuisce al software concede licenze d'uso illimitate a tutti gli altri utenti e, analogamente a quanto succede

con la Mozilla Public License, prevede una termination clause, cioè che tali licenze d'uso illimitate cessino immediatamente nei confronti di chi cita in giudizio gli autori di software coperto dalla APL per violazioni di propri brevetti.

2.8.2.7 La licenza artistica

Questa licenza è stata ideata nel 1991 per la distribuzione del linguaggio di programmazione Perl 4.0. E' una licenza permissiva che, in quanto tale, è incompatibile con la licenza GPL, e proprio per consentire la compatibilità con quest'ultima Perl è stato distribuito sia con licenza artistica che con la GPL. Da un punto di vista legale è molto vaga e presenta diverse lacune che, unite alla sua intrinseca permissività, non consentono un'adeguata tutela.

Capitolo 3

L'Open Source e le forme di resistenza creativa del consumatore

3.1 Premessa: De Certeau e la questione dell'agency del consumatore

Il fenomeno open source mette in evidenza come gli individui, coinvolti nella collaborazione online finalizzata alla produzione di artefatto, grazie alla Rete possono assumere un ruolo attivo e creativo contribuendo alla propria autorealizzazione (Hemetsberger, 2005).

In tal senso, questo fenomeno può essere letto sotto la lente della letteratura ricadente nel filone della *consumer culture theory* (Arnould & Thompson, 2005), che, tra i diversi temi, si occupa degli aspetti produttivi insiti nel consumo, ed in particolare di come i consumatori rielaborano attivamente e trasformano i significati simbolici insiti nella pubblicità, nei marchi o nei beni materiali – nonché i beni materiali stessi – grazie alle risorse che il mercato mette a disposizione del consumatore per la creazione della propria identità individuale e collettiva.

In questo senso può essere utile ed interessante inquadrare il movimento open source come una forma collettiva di resistenza creativa del consumatore rispetto al mercato, in quanto attraverso la partecipazione ad una comunità di programmatori open source l'individuo cerca di emanciparsi dal "lato oscuro del mercato", di dare significato alle proprie esperienze, e di costruire un diverso aspetto di sé e della propria identità (Hemetsberger, 2005).

L'idea che il consumatore non sia un mero soggetto passivo non è nuova; già Michel De Certeau definisce il consumo come il regno dell'uso dell'oggetto da parte di chi non l'ha prodotto; ma anche nel momento dell'uso esiste un momento di produzione, di realizzazione, una poietica, secondo l'Autore nascosta, perché si dissemina negli spazi definiti e occupati dai sistemi della "produzione" (televisiva, urbanistica, commerciale, etc) e perché l'estensione sempre più totalitaria di tali sistemi non lascia più ai "consumatori" un luogo in cui rivelare l'uso che fanno dei prodotti. A una produzione razionalizzata, espansionista e al tempo stesso centralizzata, chiassosa e spettacolare, ne corrisponde un'altra, definita consumo: un'attività astuta, dispersa, che però si insinua ovunque, silenziosa e quasi invisibile, poiché non si segnala con prodotti propri, ma attraverso il modo di usare quelli imposti da un ordine economico dominante (De Certeau, 1990, p. 7).

Le principali categorie elaborate da De Certeau nel teorizzare il consumo sono le strategie e le pratiche (Poster, 1992, pag. 102). Per «strategia» De Certeau intende il calcolo dei rapporti di forza che diviene possibile a partire dal momento in cui un soggetto di volontà e di potere è isolabile in un «ambiente». Essa presuppone un luogo che può essere circoscritto come *proprio* e fungere dunque da base a una gestione dei suoi rapporti con un'esteriorità distinta. La razionalità politica, economica o scientifica è stata costruita su questo modello strategico.

Intende al contrario per «tattica» un calcolo che non può contare su una base propria, né dunque su una frontiera che distingue l'altro come una totalità vivibile. La tattica ha come luogo solo quello

dell'altro. Si insinua, in modo frammentario, senza coglierlo nella sua interezza, senza poterlo tenere a distanza. Non dispone di una base su cui capitalizzare i suoi vantaggi, prepararsi a espandersi e garantire un'indipendenza in rapporto alle circostanze. (De Certeau, 1990, p. 15).

"Sono le mille pratiche il cui uso serve a riappropriarsi dello spazio organizzato mediante le tecniche della produzione socio-culturale, che pongono questioni analoghe e contrarie a quelle affrontate nel libro di Foucault: analoghe, poiché si tratta di distinguere la proliferazione di operazioni quasi microscopiche all'interno delle strutture tecnocratiche e di trasformare il funzionamento attraverso una molteplicità di «tattiche» basate su «dettagli» quotidiani: contrarie, poiché non si tratta più di precisare in che modo la violenza dell'ordine si tramuti in tecnica disciplinare, bensì di riesumare le forme surrettizie che assume la creatività dispersa, tattica e minuta dei gruppi o degli individui intrappolati ormai nelle reti della «sorveglianza». Queste procedure e astuzie dei consumatori finiscono col costituire la trama di un'antidisciplina che è precisamente l'oggetto della nostra ricerca." (De Certeau, 1990, pp. 8-9).

Attraverso le pratiche quotidiane di consumo, quindi, il consumatore cerca di rimpiazzare gli schemi interpretativi imposti dai marketer con i propri schemi socio-culturali, alterando in questo modo il significato dei beni di consumo e dei servizi (Moisio & Askegaard, 2002).

Il ruolo attivo assunto dal consumatore nella rielaborazione, attraverso le pratiche quotidiane di consumo, del significato attributo

ai beni dall'industria secondo la visione di De Certeau si discosta dalla precedente visione pessimistica elaborata da Adorno e Horkheimer (Horkheimer e Adorno, 1996, cit. in Holt, 2002) secondo i quali invece il consumatore ha un ruolo meramente passivo, in quanto il sistema della produzione culturale di massa, con cui la cultura diventa merce, rappresenta il collante ideologico che mantiene ampio il consenso delle masse verso il potere dominante.

In questo senso, l'unico modo rimasto al consumatore per esprimere la propria identità consiste nella possibilità di scegliere tra prodotti differenziati, creati dall'industria, grazie alla segmentazione della domanda, un sistema di dominio grazie al quale il consumatore viene classificato, organizzato ed etichettato. Le nuove tecnologie distributive comportano la cancellazione delle idiosincrasie (Horkheimer e Adorno, 1996, cit. in Holt, 2002).

L'attenzione verso la produzione simbolica attuata dal consumatore è rilevante nella visione postmoderna del consumo elaborata da Firat e Venkatesh (1995), che rifiutano l'impostazione moderna del momento del consumo come nettamente distinto dal momento della produzione, dicotomia che si ricollega ad altre di derivazione modernista, come la separazione tra il momento del lavoro e quello del divertimento, tra la casa e il luogo di lavoro, e la separazione tra il dominio pubblico e il dominio privato, in cui la produzione è un'attività svolta unicamente nell'ambito "pubblico" della società: nella fabbrica, nell'ufficio etc. Il consumo, nella visione modernista, serve all'individuo unicamente per essere in grado di svolgere l'unica attività realmente creativa, la produzione.

Il postmodernismo ha spostato l'analisi dal luogo della produzione a quello del consumo, laddove quest'ultimo è il momento del processo in cui si verificano gli scambi simbolici che determinano e riproducono il codice sociale, dove si verifica un'attiva appropriazione dei segni e non la semplice distruzione dell'oggetto. Questo momento di creazione non è individuale, ma è un vero atto sociale in cui significati simbolici, codici sociali, ideologie politiche e relazioni sono prodotte e riprodotte.

Nella visione postmoderna elaborata dai due autori, il processo di consumo è "liberatorio", e combina sia il "reale" (l'oggetto) che l'immaginario (significati, cultura e legami derivanti dal bene di consumo ed elaborati dal consumatore): con questo processo, le persone consumano oggetti, simboli ed immagini sempre più considerati come un unicum, mentre con l'approccio modernista considera il consumo solo nel suo essere parte del mercato. La domanda che Firat e Venkatesh (1995, p. 258) si pongono è se il consumo possa aver luogo al di fuori del mercato e, a tal proposito, elaborano la metafora dei "teatri del consumo", spazi di emancipazione in cui essi rifiutano i significati e le identità codificate dalla logica totalizzante del mercato, e contribuiscono attivamente alla creazione di ciò che "va in scena".

Holt (2002) contesta il fatto che la celebrazione postmoderna della sovranità del consumatore nell'uso creativo e talvolta sovversivo del mercato corrisponda alla sua emancipazione. Attraverso il suo case-study, avente per oggetto le scelte di consumo di tre persone, soprattutto dalle informazioni ottenute da due dei suoi intervistati, entrambi estremamente abili nello sviluppare delle scelte di consumo

antimarketing, egli rileva come tali scelte abbiano come "teatro" il mercato, e non la famiglia o la comunità o il lavoro. Anche Kozinets (2002) nella sua etnografia del "Burning man" mette in evidenza come in realtà la partecipazione a questa manifestazione rappresenti una forma di temporanea emancipazione del consumatore rispetto al mercato: "perhaps it is not possible to completely evade the market. . . The urge to differentiate drives from other consumers participation at Burning Man and does not release them from the grip of the market's sign game and its social logics".

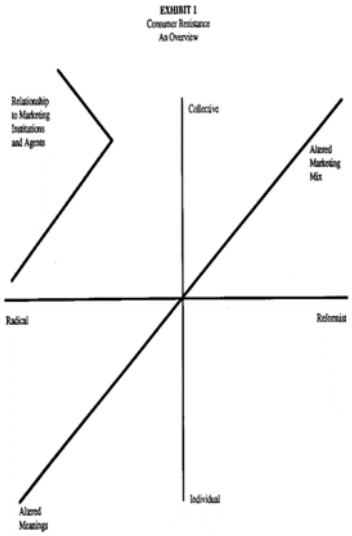
Come Thompson (2004) rileva, sia Firat & Venkatesh (1995), che Holt (2002) che Kozinets (2002), pur inserendo i propri contributi nel filone postmoderno dell'approccio alla teoria del consumo, nel voler o cercare di posizionare il consumatore fuori o dentro il mercato adottano paradossalmente una visione modernista del mondo, se è vero che il postmodernismo si caratterizza per il rifiuto della netta divisione tra categorie. L'esempio che Thompson (2004) prima e Thompson & Coskuner-Balli (2007) poi propongono per descrivere la difficoltà – se non l'impossibilità – per il consumatore di emanciparsi dal mercato riguarda il consumo di cibo biologico, che negli anni '60 era un totem della controcultura hippie e che oggi viene venduto anche da Wal-Mart, attraverso un processo che ha avuto luogo nel corso di circa 30 anni e che Thompson e Coskuner-Balli definiscono come co-optazione.

3.2 La tassonomia della resistenza

Seguendo Poster (1992), Peñaloza e Price (1993) definiscono la resistenza del consumatore come il modo con cui gli individui ed i gruppi praticano una strategia di appropriazione in risposta alle

strutture di dominazione. Secondo le due autrici, è possibile costruire una tassonomia delle numerosissime forme di resistenza, individuando quattro diverse dimensioni (fig. 3.1).

Fig. 3.1: Le quattro dimensioni della resistenza del consumatore



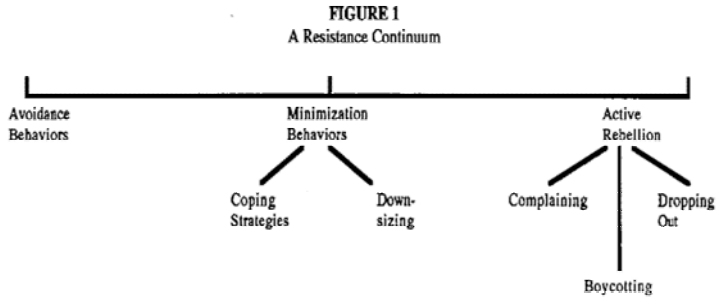
Fonte: Peñaloza e Price (1993)

Un asse è rappresentato dalla dimensione organizzativa, che può variare dalle forme individuali a quelle collettive; un secondo asse è costituito dall'ampiezza dell'obiettivo perseguito, che può essere riformista o radicale. Una terza dimensione rappresenta le tattiche di resistenza che possono andare dalle azioni dirette ad alterare il marketing mix ad azioni dirette a modificare il significato dei prodotti. Infine, l'ultima dimensione rappresenta l'importanza delle legame del consumatore con le istituzioni e gli agenti del marketing, riconoscendo che la resistenza del consumatore può far propri tali istituzioni come strumento della propria resistenza oppure il consumatore può rimanere al di fuori di tali istituzioni, utilizzandone altre come strumenti di cambiamento.

La tassonomia di Peñaloza e Price fa riferimento al modo di resistere, ma non alle sue motivazioni (Roux, 2004), ed è chiaramente ispirata al concetto di tattica elaborata da de Certeau (cfr. *supra*).

Secondo un secondo approccio, quello di Fournier (1998), la resistenza può essere concepita come un *continuum* di risposte del consumatore al mercato, in cui la forma più attenuata è rappresentata dai comportamenti di evitamento, la posizione mediana è invece rappresentata dai comportamenti di aggiustamento e riduzione dell'acquisto, e la forma più estrema, invece, dagli atti di ribellione attiva come i boicottaggi (fig. 3.2)

Fig. 3.2: Il continuum della resistenza



Fonte: Fournier (1998)

Simile a questa è la concezione di resistenza del consumatore offerta da Ritson e Dobscha (1999), ritenuta una forma variabile di partecipazione e coinvolgimento del consumatore nel mercato.

Secondo Moisio ed Askegaard (2002) le classificazioni precedentemente esposte non pongono sufficiente attenzione sul fatto che le diverse forme di *consumer resistance* differiscono su un piano ontologico, che rappresenta diverse concezioni del consumatore, del consumo così come del legame tra i consumatori e le loro pratiche di resistenza. Secondo la loro visione, il concetto di consumer resistance può essere declinato in tre diverse accezioni.

Una prima forma di tale fenomeno è rappresentata dalla resistenza all'adozione o all'acquisto di prodotti e servizi o dalla propensione dei consumatori a venir coinvolti in forme di azione collettiva, che essi considerano come una forma negativa di resistenza al mercato. La seconda classe di significati si basa sulle dimensioni simboliche legate al consumo, quelle che provocano l'acquisto o il rifiuto di determinati prodotti o marche, in cui l'elemento dominante è l'avversione del consumatore verso una particolare classe di prodotti e di servizi, e in questa tipologia di consumer resistance si creerebbero dei legami simbolici tra gruppi di consumatori, aventi a loro volta dei confini che li contraddistinguono. Infine, un terzo tipo di significati conferisce agli atti di ribellione la dimensione di una scelta politica specifica di fronte a ciò che è percepito come una egemonia culturale del consumo, ed in questo senso essi sono dei micro-atti politici di ritorsione rispetto a forme e pratiche culturali dominanti.

L'aspetto più interessante dell'analisi di Moisio e Askegaard è l'evidenziazione della possibilità che i consumatori abbiano una consapevolezza più o meno marcata della loro "ribellione": accanto ad atti di opposizione volontaria e meditata, infatti, vi sono delle pratiche di resistenza meno marcata, che si traducono in gesti banali e quotidiani come le diverse modalità d'uso, nell'esempio portato dai due Autori, del telefono cellulare.

3.3 Dalla resistenza individuale alla resistenza collettiva

Il consumatore, inizialmente attivo nella produzione meramente simbolica attraverso la rielaborazione di nuove pratiche di consumo diverse da quelle inizialmente ideate dal mercato, mediante la quale egli crea nuovi significati del prodotto, realizza nuove forme di

resistenza creativa, dando luogo, in determinati contesti sociali a vere proprie "sottoculture"[23] (Hebdige, 1983), in cui questi simboli vengono elaborati in modo totalmente diverso, incoerente rispetto a quanto proposto dalla maggioranza. A ciò si aggiunge il fatto che la resistenza del consumatore si muove dalla dimensione individuale a quella collettiva, arrivando ad alterare in misura sempre più rilevante il marketing mix.

In misura sempre più notevole nel corso degli anni '90 si è rilevata una notevole attenzione della letteratura verso gli aspetti sociali del consumo, ispirata anche dalla crescente diffusione della visione postmoderna del consumo dato che, come Firat (2006) mette in evidenza "the postmodern is often the forceful return to/of community (…)". La ricerca del consumatore di significati e sostanza, e l'immersione in esperienze ricche di ciò, può avvenire attraverso la partecipazione e la costruzione di comunità. L'epoca moderna, invece, dall'Illuminismo in poi, si era caratterizzata per il tentativo di "liberare" l'individuo dalla schiavitù indotta dalle comunità tradizionali (famiglia, scuola, religione) sostituendo ai legami imposti da questi gruppi sociali un legame liberamente scelto, reversibile, come il contratto (sociale).

L'individuo postmoderno, secondo Cova (1997), appare oggi come liberato dai limiti imposti dagli ideali collettivi riguardanti l'istruzione, la famiglia etc, si presenta con un numero limitato di legami sociali e la sua azione è caratterizzata da un'estrema mobilità

[23] Schouten & McAlexander (1995) offrono questa definizione di sottocultura: "We define a subculture of consumption as a distinctive subgroup of society that self-selects on the basis of a shared commitment to a particular product class, brand or consumption activity. Other characteristics of a subculture of consumption include an identifiable, hierarchical social structure; a unique ethos, or set of shared beliefs and values; and unique jargons, rituals, and modes of symbolic expression."

sia spaziale che sociale. Egli è un nomade del presente. In questo senso, e in apparente contraddizione con quanto postulato da Firat (2006), l'epoca postmoderna può essere vista come un periodo di notevole dissoluzione sociale e di estremo individualismo. Al contempo, però, l'individuo, liberato dai vincoli imposti dalle comunità tradizionali, può ricomporre il proprio universo sociale attraverso la propria libera scelta, un tentativo disperato di ricostruire un legame sociale attraverso la partecipazione a comunità quasi arcaiche, che Cova (1997) definisce come tribù.

Il progresso tecnologico dell'epoca moderna, pur avendo liberato l'individuo da molteplici incombenze domestiche lo ha di fatto ulteriormente isolato, in quanto diverse attività venivano svolte in modo comunitario, con i vicini etc (si pensi ad esempio al lavaggio del bucato, o alla cottura del pane nel forno comunale). Pur consentendo il soddisfacimento del bisogno primario, i prodotti realizzati non sono riusciti a rimpiazzare l'aspetto sociale di tali attività, ora svolte, ad esempio, dagli elettrodomestici.

L'insoddisfazione del consumatore rispetto a questa ricerca dell'aspetto comunitario del consumo può essere legato alla deconsumption e può assumere la forma del rifiuto della soddisfazione virtuale legata all'acquisto, nonché attraverso il continuo acquisto del "nuovo" ma, soprattutto, attraverso la ricerca della soddisfazione attraverso la condivisione delle proprie emozioni con gli altri. Per il consumatore postmoderno, quindi, i beni ed i servizi acquistati hanno valore in quanto consentono di creare legami sociali con gli altri, e si assiste quindi alla riduzione del consumo dei beni che isolano

l'individuo, a favore invece di quelli che creano un legame con gli altri, e che quindi possiedono un *linking value*.

L'attribuzione al prodotto e quindi al brand di un significato legato alla sua capacità di rendere possibile l'interazione sociale rafforza il *consumer empowerment* (Wathieu, 2002), che si realizza a seguito della capacità del consumatore di assumere il controllo di variabili tradizionalmente sotto il controllo dei marketers, come, appunto, il significato del brand (Cova & Pace, 2006).

Uno degli attori determinanti nell'appropriazione e ridefinizione del brand da parte del consumatore è la *brand community* o comunità di marca.

Le comunità di marca sono entità sociali in cui si riflette il radicamento della marca nella vita quotidiana dei consumatori, e i modi con cui questa lega il consumatore a sé ed il consumatore con gli altri consumatori.

Secondo Muniz & O'Guinn (2001) affinchè si possa parlare di una brand community è necessario che siano presenti i seguenti elementi:

1. "consciousness of a kind"[24]„ cioè la consapevolezza dell'appartenenza ad un gruppo (o "consapevolezza di razza" – Cucco & Dalli, 2008), grazie ad una marca che viene sostenuta da tutti i membri della comunità;
2. Esistenza di rituali e tradizioni che circondano il marchio;

[24] Il termine è stato inizialmente definito dal sociologo statunitense F.H. Giddings, che usò questa definizione: *"[Consciousness of kind is] that pleasurable state of mind which included organic sympathy, the perception of resemblance, conscious or reflective sympathy, affection, and the desire for recognition."* In altri termini, la consciousness of kind è un sentiment innato di appartenenza e di similarità (American Sociological Association)

3. Assunzione di impegni e doveri nei confronti della comunità e dei membri che la compongono.

La "consciousness of kind" può essere definita come la connessione intrinseca che i membri di una comunità avvertono rispetto agli altri, e la percezione collettiva della differenza esistente rispetto a chi non fa parte di essa, ovvero un senso condiviso di appartenenza.

Il secondo elemento distintivo di una comunità di marca è la presenza di rituali condivisi e di tradizioni, che perpetuano la storia, la cultura e la consapevolezza condivisa della comunità. I rituali servono a contenere il flusso di significati, mentre le tradizioni sono un insiemi di pratiche sociali per mezzo delle quali vengono celebrati ed inculcati determinati valori e norme comportamentali.

Il terzo elemento distintivo è la percezione, da parte dei membri della comunità, di obbligazioni e doveri rispetto alla comunità.

Lo spazio ideale per lo sviluppo delle brand communities e, in generale, per le comunità di consumo e di innovazione è senza dubbio rappresentato da Internet: assumono quindi una crescente rilevanza le comunità virtuali di consumo, intese come reti di individui che partecipano ad un processo congiunto di creazione di conoscenza, su temi in qualche modo legati ad attività di consumo, in ambienti mediati da un'interfaccia elettronica (Verona & Prandelli, 2006). Nella definizione di Kozinets (1999), le comunità virtuali di consumo rappresentano uno specifico sottogruppo di comunità virtuali che basano esplicitamente il proprio focus su attività legate al consumo; sono gruppi le cui interazioni online si basano sulla passione condivisa e la conoscenza di una specifica attività di consumo.

Fig. 3.3: Evoluzione della partecipazione alle CVC

Fonte: Kozinets (1999)

Lo schema dello sviluppo delle relazioni nelle comunità virtuali di consumo (CVC) si caratterizza per il fatto che la consumption knowledge va di pari passo con le relazioni sociali. La conoscenza nel consumo viene diffusa attraverso le norme culturali del gruppo online, i concetti e i linguaggi specialistici, e l'identità di esperti e altri membri del gruppo. La coesione culturale si alimenta attraverso le storie condivise e l'empatia; si ha la creazione di una struttura di potere (Kozinets, 1999). Ciò che era inizialmente una fonte di informazioni diventa una comunità (fig. 3.3) Può essere utile, a questo punto, condurre una piccola digressione sul concetto di "comunità virtuale".

3.4 Le comunità virtuali

Secondo Rheingold (1993), il primo ad aver creato una comunità virtuale, the WELL, l'avvento delle comunità virtuali è stato predetto nel 1968 da J.C.R. Licklider e R. Taylor, direttori della ricerca per la Advanced Research Project Agency (ARPA) del dipartimento della Difesa degli USA, promotori delle ricerche che successivamente (vds capitolo 1) diedero vita alla rete ARPANet, che

alla domanda "Come saranno le future comunità interattive on-line?" diedero questa risposta: "Saranno composte da membri geograficamente dispersi, qualche volta raggruppati in piccoli clusters, altre da individui che lavoreranno singolarmente, e saranno comunità che condividono uguali interessi.

La prima comunità virtuale, the WELL fu creata proprio da Rheingold nel 1985, che la definì (Rheingold, 1993) "un'aggregazione sociale che emerge dalla Rete quando un numero sufficiente di persone si impegnano abbastanza a lungo in discussioni pubbliche, con un discreto feeling umano, creando ragnatele di relazioni personali nel cyberspazio".

In questo senso, affinchè si possa parlare di comunità virtuale, è necessario che esista:

- Una massa critica, cioè un numero di utenti abbastanza elevato da rendere attraente la comunità;
- L'interazione nel tempo, in quanto le relazioni create non possono essere spot;
- La partecipazione consapevole, necessaria affinchè si crei un senso di appartenenza da parte dei suoi membri.

La definizione offerta da Rheingold è sicuramente generica, ma coglie sicuramente una pluralità di fenomeni anche se, successivamente, si è animato un vivace dibattito accademico sulle condizioni necessarie affinchè si possa parlare di comunità virtuali (Verona & Prandelli, 2006).

Wellman & Gulia (1999), analizzano le caratteristiche delle comunità virtuali rispondendo ad alcune domande chiave.

Il primo interrogativo che i due Autori si pongono è in merito al tipo di supporto ci si può aspettare dalle relazioni nate nelle comunità virtuali. Nelle comunità reali, da ciascuno dei rapporti, anche stretti, in genere si ottiene un aiuto specializzato: chi offre un sostegno emotivo è in genere diverso da chi offre compagnia o aiuto finanziario. Le persone ottengono tutte le tipologie di sostegno dall'appartenenza ad una comunità, ma hanno bisogno di rivolgersi a diversi membri della comunità per i diversi tipi di aiuto. Ne consegue che le persone devono avere un portafoglio differenziato di legami per ottenere un'ampia varietà di risorse.

Per quanto riguarda le relazioni formatesi nella Rete, la tecnologia per un certo aspetto alimenta le relazioni specializzate, basate sullo scambio di informazioni reso rapido ed efficiente da questo mezzo di comunicazione. Un esempio tipico di relazioni specializzate è dato dai canali IRC delle chat, apparentemente organizzate su uno specifico topic, attorno al quale si crea un gruppo di persone in cui l'elemento aggregante è la condivisione di un interesse comune, piuttosto che di un luogo condiviso.

Ma la Rete non è solamente un mezzo per lo scambio di informazioni, e ciò è dimostrato dal fatto che, anche se nati per non essere *supportive*, molti gruppi della Rete finiscono per esserlo, in virtù del fatto che, come esseri sociali, chi vi partecipa spesso ricerca amici e legami sociali che vanno aldilà del mero scambio di informazioni.

Il secondo interrogativo che Wellman e Gulia si pongono è in che modo i "legami deboli" creati sulla Rete possono essere utili. Le comunità virtuali condividono con quelle reali la capacità di offrire

aiuto ai propri membri, spesso sotto forma di relazioni specializzate. Un loro elemento distintivo è invece rappresentato dalla capacità di offrire informazioni, sostegno, amicizia, senso di appartenenza a persone quasi o totalmente sconosciute offline. Questa situazione è in palese contrasto con quanto si crea a livello di interazioni face-to-face, in cui è estremamente difficile che si intervenga per aiutare uno sconosciuto, mentre è possibile osservare come, ad esempio in un newsgroup, una richiesta di supporto, rivolta non ad una specifica persona ma al gruppo intero, venga facilmente raccolta da qualcuno, il cui sforzo sarà rilevato dal newsgroup e positivamente ricompensato.

Un altro elemento che determina l'esistenza di legami deboli nelle comunità virtuali è la maggiore facilità con cui nelle interazioni virtuali è possibile uscire da situazioni problematiche, rispetto a quanto sia possibile nei rapporti face-to-face. A questo si aggiunge l'ignoranza sullo status sociale del proprio interlocutore, di cui spesso si conoscono pochissimi dati personali, spesso solo l'indirizzo e-mail. In ogni caso, sia on che off line, i legami deboli sono i più adatti a creare interrelazioni tra persone aventi caratteristiche sociali differenti, e ciò comporta il fatto che, in determinati contesti, è più importante avere contatti con persone diverse da un punto di vista sociale e culturale, piuttosto che con un ampio numero di persone simili.

Le caratteristiche tipiche delle comunità virtuali, ovvero assenza di un'interazione personale, prevalenza di legami deboli e assenza di una struttura comunitaria che imponga il rispetto di una reciprocità fanno emergere il problema della motivazione delle singole persone nell'offerta di sostegno e aiuto agli altri membri della comunità. Wellman e Gulia (1999) rilevano che, in genere, maggiore è la

distanza fisica e sociale tra chi cerca (the seeker) aiuto e chi lo offre (provider), minore sarà la probabilità con cui questo aiuto sarà restituito.

In realtà nelle comunità virtuali emergono altre possibili determinanti della motivazione a partecipare dei singoli membri; una di queste si ha quando offrire informazioni e assistenza in rete è un mezzo per esprimere la propria identità, in particolare se le capacità e conoscenze tecniche sottese a tale aiuto sono parte integrante della concezione di sé. In questo caso, aiutare gli altri aumenta la propria autostima, il rispetto degli altri e facilita il raggiungimento di un particolare status all'interno del gruppo.

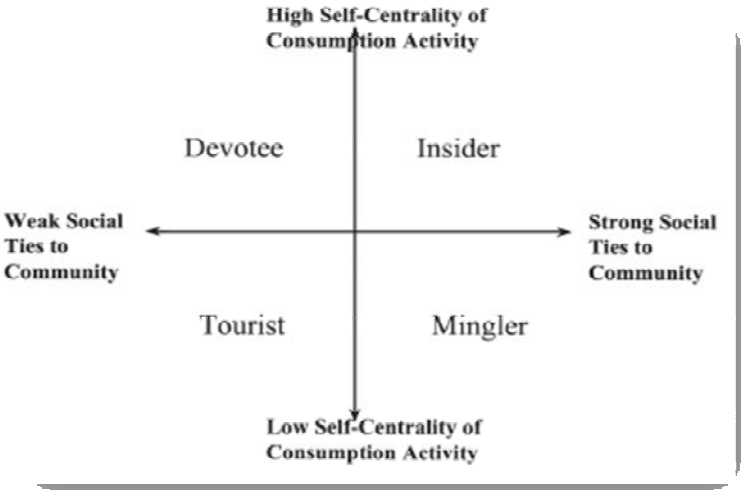
Nella cultura hacker, ad esempio, quando vengono poste in essere attività illegali, che renderebbero necessario il ricorso all'anonimato, gli hackers ricorrono a pseudonimi, che però vengono utilizzati ripetutamente, nonostante questo possa portare alla loro identificazione, perché cambiando nickname perderebbero lo status associato ad esso.

L'altro elemento che determina la motivazione a partecipare alla comunità è la condivisione dei suoi obiettivi, e l'esistenza di un forte attaccamento alla stessa da parte dei propri membri, dipendente a sua volta anche dall'intensità dei legami che le persone che la compongono riescono a creare.

L'intensità dei legami esistenti tra la comunità virtuale e i partecipanti dipende da diversi fattori, diversi a seconda del tipo di comunità analizzato; nel caso delle comunità virtuali di consumo, Kozinets (1999) individua due fattori non indipendenti. Il primo è la relazione esistente tra l'individuo e l'attività di consumo: maggiore è

la centralità dell'attività di consumo rispetto al concetto di sé o, meglio, maggiore è l'importanza assunta dai simboli di questa particolare forma di consumo per l'immagine che di sé ha il consumatore, maggiore sarà la probabilità che il consumatore divenga parte della comunità (virtuale o reale) che è focalizzata su tale attività di consumo. Il secondo fattore è l'intensità dei rapporti sociali che il singolo crea con gli altri appartenenti alla comunità. Grazie a queste due variabili, Kozinets costruisce una tassonomia dei partecipanti alle comunità virtuali di consumo, rappresentata dalla figura sottostante (fig. 3.4).

Fig. 3.4: I partecipanti alle comunità virtuali



*Fonte:*Kozinets, 1999 p. 255

La prima tipologia di utenti è quella dei *tourists* (turisti) che non hanno forti legami con il gruppo e hanno un interesse superficiale o temporaneo nei confronti dell'attività di consumo. La seconda tipologia di utenti è quella dei *minglers* (combattuti), che coltivano forti legami con la comunità pur avendo un interesse ridotto nell'attività di consumo; segue poi la categoria dei *devotee* (dedicati)

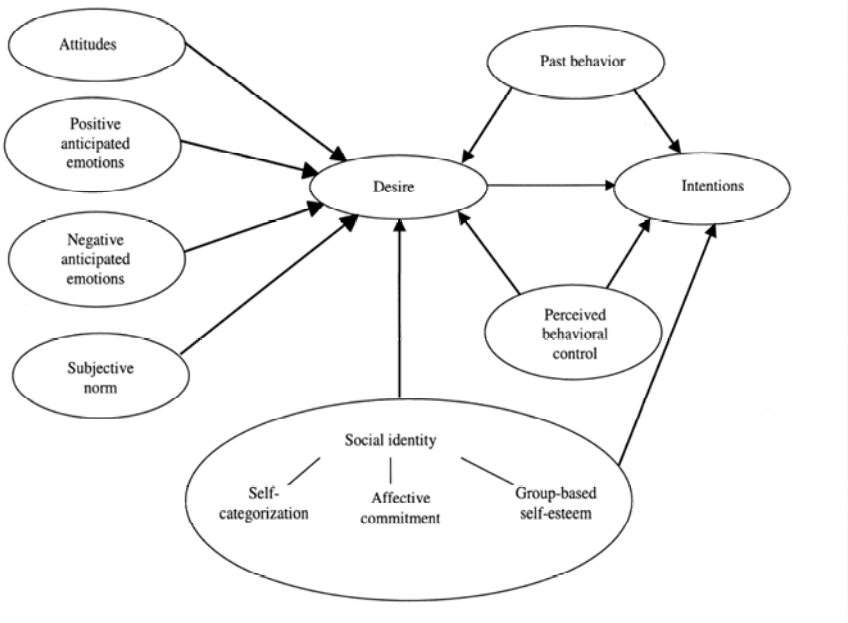
che si caratterizzano per avere un comportamento esattamente opposto, in quanto hanno un forte interesse e passione nei confronti dell'attività di consumo ma un ridotto attaccamento sociale al gruppo. Infine gli *insiders* (integrati) che hanno un forte interesse nell'attività di consumo e un forte legame al gruppo.

Come messo in evidenza da Verona e Prandelli (2006), le comunità virtuali di consumo tendono ad agire quali meccanismi volti alla continua rigenerazione della conoscenza accumulata, fornendo stimoli utili ad alimentare sistematicamente il processo innovativo dell'impresa. L'apporto di informazione e e la socializzazione di esperienze da parte di soggetti sempre nuovi fa sì che il patrimonio di conoscenza sedimentato all'interno di una comunità virtuale sia perennemente in divenire e l'attrattività della stessa sia funzione proprio di quanto più intenso è questo meccanismo di rinnovamento della conoscenza collettiva. A differenza delle comunità fisiche, che nascono da una comune matrice territoriale, storica e valoriale, e mirano alla produzione e al rafforzamento della conoscenza costruita al loro interno, sancendo in maniera netta chi appartiene al gruppo e chi no, le comunità virtuali si caratterizzano per avere confini dinamici, a rispecchiare la contingente ricerca di nuove forme di conoscenza. Anche il livello di coinvolgimento tende ad essere diverso, in quanto l'adesione ad una comunità virtuale è sempre frutto di una scelta personale, anche se nella maggioranza dei casi dà vita a relazioni basate su "legami deboli".

In merito alla possibilità che l'adesione ad una comunità di consumo, fisica o virtuale, sia il risultato di una scelta consapevole, Bagozzi (2000) ha elaborato il concetto di *"intentional social action"*.

Il suo approccio si differenzia dall'analisi delle collettività sociali elaborate da Holt (1997), laddove per esse Holt intende gruppi di persone che hanno creato legami sociali a partire da similarità esistenti tra di loro (istruzione, lavoro etc), sono coinvolti in relazioni sociali simili, e quindi tendono ad avere lo stesso background culturale. Le collettività non sono organizzate formalmente e possono essere geograficamente disperse; sono socialmente costruite però l'appartenenza ad esse non è necessariamente frutto di una scelta consapevole, secondo l'interpretazione di Holt.

Fig. 3.5: Il modello dell'intentional social action



Fonte: Bagozzi, 2000

Secondo il modello dell'azione sociale intenzionale di Bagozzi, invece, l'azione collettiva è necessariamente consapevole, in quanto i membri di una collettività acquisiscono un'identità sociale che si manifesta in (1) una componente cognitiva, composta dalla consapevolezza dell'essere membro di un gruppo, (2) una componente affettiva che consiste in un sentimento di appartenenza al gruppo, (3)

una componente valutativa che è collegata alla autostima collettiva. L'identità sociale ha effetti sul modo in cui i membri pensano e sulla percezione di sé, come le persone che fanno o non fanno parte del gruppo sono considerate, e sull'azione nei confronti di persone che fanno o non fanno parte della collettività. L'elemento centrale dell'analisi di Bagozzi è quindi ciò che egli chiama *we-intention*, riprendendo la citazione di Tuomela (1995, cit. in Bagozzi, 2000), che la definisce come "l'impegno di un individuo a partecipare ad un'azione congiunta che comporta un accordo, implicito o esplicito, a realizzare tale azione comune. Le we-intention comportano necessariamente l'esistenza di forme di cooperazione tra i membri di un gruppo e il coordinamento di piani ed azioni". Le we-intentions vengono formulate quando gli individui pensano a sé stessi come "noi" o "noi stessi" e sono congiuntamente pronti ad agire per realizzare gli obiettivi collettivi del gruppo (Bagozzi & Dholakia, 2006).

Il nucleo del modello di comportamento (fig. 3.5) di queste collettività sono i desideri e l'identità sociale, in quanto al cuore del consumo collettivo c'è il desiderio di acquistare, usare, mostrare o disporre di beni materiali o di servizi. Mentre una prospettiva individualistica potrebbe vedere i desideri del consumatore come la volontà di acquisire una distinzione o differenziazione individuale, la prospettiva dell'intentional social action vede il desiderio del gruppo di differenziarsi, attraverso le proprie scelte di consumo, dagli altri gruppi, di acquisire un proprio miglioramento attraverso le azioni ed il raggiungimento degli obiettivi del gruppo. L'identità sociale consiste

di processi di auto-categorizzazione[25], emozioni, positive o negative anticipate dai partecipanti alla comunità, autostima basata sul gruppo che determina il desiderio degli utenti di partecipare al gruppo.

Secondo Melucci (1996) l'identità collettiva è una definizione interattiva e condivisa prodotta da un certo numero di individui riguardo l'orientamento della propria azione e l'ambito delle opportunità e dei limiti entro i quali tale azione avrà luogo. Per definizione interattiva e condivisa, Melucci intende che gli elementi che la compongono vengono costruiti e negoziati attraverso un processo di attivazione ricorrente delle relazioni che lega gli attori insieme. Ne consegue che:

1. L'identità collettiva, come processo, comporta l'esistenza di definizioni cognitive in merito ai fini, ai mezzi e all'ambito dell'azione del gruppo. Questi diversi elementi, o assi, dell'azione collettiva sono definiti per mezzo di un linguaggio ed incorporati in un insieme di rituali, pratiche ed artefatti culturali;
2. L'identità collettiva come processo fa riferimento quindi ad un network di relazioni attive tra attori che interagiscono, comunicano, si influenzano a vicenda, negoziano e prendono decisioni. Forme di organizzazione e modelli di leadership, canali e tecnologie di comunicazione sono elementi costitutivi di questo network di relazioni;
3. Infine, nella definizione di un'identità collettiva si richiede un certo livello di investimento emozionale, che rende possibile

[25] E' il processo attraverso il quale una persona vede se stessa come appartenente ad una categoria sociale, con implicazioni sulla percezione di sé e conduce alla spersonalizzazione al fine di accentuare le similarità rispetto agli altri membri del gruppo.

agli individui sentirsi parte di una unità comune. L'identità collettiva non è mai interamente negoziabile perché la partecipazione all'azione collettiva è dotata di un significato che non può essere ridotto ad un mero calcolo costi-benefici. Esiste quindi una parte irrazionale dell'identità collettiva.

In una comunità (virtuale o fisica che sia) assume quindi una grande rilevanza l'interazione discorsiva e libera tra i partecipanti ed è, soprattutto, la produzione di materiali originali prodotti dai suoi membri che rende possibile consolidare la comunità (Verona & Prandelli, 2006)., materiali che vengono liberamente distribuiti e condivisi grazie all'esistenza su Internet grazie ad una vera e propria Internet o Hi-tech gift economy (Barbrook, 2005)

3.5 La gift economy

Le interazioni sociali che caratterizzano le comunità virtuali, non solo di consumo, sono basate, come già Rheingold (1993) aveva messo in evidenza, su una *gift economy* o economia del dono, in cui vengono offerti aiuto e informazione senza che ci sia l'aspettativa di un'immediata contropartita.

Veale (2003) mette in evidenza come Internet, probabilmente anche a causa delle sue origini accademiche, abbia per oltre 25 anni vissuto in un ambiente pre-commerciale, in cui la produzione di contenuti si è basata sul dono, sulla produzione e condivisione gratuita di informazioni.

Ma che cos'è un dono? Kollock (1999), riprendendo la definizione di Carrier (1991, cit. in Kollock, 1999), definisce il dono come (1) il trasferimento obbligatorio (2) di oggetti e servizi inalienabili (3) tra due agenti economici collegati e mutualmente

obbligati. Mauss (1924, trad. it. 2002) vede nello scambio di doni un prototipo di contratto, in cui l'individuo è obbligato a dare, ricevere e ricambiare.

Sherry (1983) rileva come le dimensioni del dono (il prezzo, la qualità) siano usate per creare, mantenere, modulare, o rompere relazioni sociali. Il valore del dono riflette parzialmente l'importanza di una relazione, e la natura mutevole della relazione è parzialmente riflessa nel valore del dono, che risente anche del mutato ruolo sociale assunto dall'individuo. La reciprocità insita nello scambio del dono deve infatti riflettere le eventuali differenze sociali esistenti tra chi dona e chi riceve, non può essere più bilanciata di queste ultime, a meno che uno dei due agenti non voglia correre il rischio di mostrare ostentazione o avidità: donare troppo o troppo poco o troppo tardi può alterare la relazione sociale fino a farla dissolvere.

Secondo il modello di Sherry (1983) lo scambio del dono si articola in tre fasi:

- Gestazione
- Prestazione
- Riformulazione

La fase della gestazione incorpora tutti i comportamenti antecedenti lo scambio del dono, è il periodo durante il quale il dono diventa reale, ed è il preludio alla creazione o al rafforzamento di un legame sociale. Esiste una condizione che determina la consegna del regalo (che può essere strutturale, come nel caso di una ricorrenza festiva) e che viene percepita, determinando l'espressione di uno stato emotivo attraverso una strategia del donare, che può essere altruistica o agonistica.

La seconda fase è quella della prestazione, in cui il dono viene effettivamente consegnato al destinatario, durante la quale sia chi dà che chi riceve il dono presta particolare attenzione al tempo, luogo e modo con cui questo viene consegnato; in altri termini, esiste un vero e proprio rituale del dono.

La terza fase è quella della riformulazione, in cui l'attenzione è focalizzata sull'uso del regalo, che può essere consumato, conservato, riciclato o rifiutato. In questa fase, il dono diventa il veicolo attraverso il quale la relazione tra il donatore ed il ricevente viene riallineata. Il legame sociale tra i due attori dello scambio può essere rafforzato, attenuato o rotto a seguito della verifica del reciproco bilanciamento. Se la relazione risulta bilanciata, allora spesso si assiste all'inversione dei ruoli in un successivo scambio di doni, con il donatore che diviene ricevente e viceversa.

Lo scambio del dono nella dimensione diadica analizzata da Sherry (1983) non è sufficiente a spiegare la gift economy tipica di Internet e delle relazioni esistenti nelle comunità virtuali. Essa rappresenta infatti un approccio atomistico che riduce le relazioni sociali sottostanti lo scambio del dono ad un rituale di interazione diadica, in cui si focalizza l'attenzione sul dono come un processo di scambio reciproco bilanciato. In realtà il dono è un "fatto sociale totale" (Mauss, 2002) , che permea la sfera sociale, politica, legale e religiosa della società, e soddisfa importanti funzioni nel loro sviluppo e nella loro continuità (Giesler, 2006)

Malinowski (1922, cit. in Giesler, 2006) rileva l'esistenza di una dicotomia tra il dono e il bene (gift vs. commodity), in cui lo scambio del dono deve essere concepita come un'economia opposta

rispetto a quella basata sullo scambio di mercato. La differenza tra il dono ed un bene economico si rileva anche nel modo in cui l'individuo può accrescerne l'utilità: nel caso del dono, i benefici ottenuti dallo scambio di doni possono essere incrementati aumentando l'ampiezza e la diversità del proprio network sociale, migliorando quindi la "tecnologia delle relazioni sociali", mentre nel caso dei beni economici o commodities la loro utilità viene accresciuta intervenendo sulle tecnologie di produzione. Quindi le gift economies si basano sulle relazioni sociali, mentre le commodities economies si basano sul prezzo.

Il modello diadico proposto da Sherry mal si adatta alla gift economy tipica delle comunità virtuali esistenti su Internet, innanzitutto perché nel dono in Rete il ricevente è spesso sconosciuto ed è altamente probabile che il donatore non lo incontri mai più: viene a mancare la reciprocità diadica ipotizzata da Mauss e dagli altri studiosi succedutiglisi. Inoltre, le informazioni e gli altri dati vengono offerti non ad uno specifico individuo, ma ad un gruppo. Mentre quindi è impossibile pensare che si possa creare una reciprocità bilanciata con un singolo individuo, è decisamente più probabile che questo bilanciamento possa avvenire con un intero gruppo. Questa situazione, percui un vantaggio prodotto a beneficio di un singolo viene restituito da un altro membro del gruppo viene definito come *scambio generalizzato* (Ezeh, in Kollock, 1999).

Questo sistema di condivisione è molto generoso e rischioso, rispetto al tradizionale dono, in quanto un individuo produce un beneficio per un terzo senza aspettarsi un'immediato contraccambio, creando al contempo il pericolo che il terzo possa raccogliere

informazioni senza offrire nessun contributo in cambio. Se tutti cadono nella tentazione di agire da free rider, allora nessuno produrrà ed otterrà più alcuna informazione utile su Internet che qualcun altro possiede, e tale problema si ha in presenza dei beni pubblici.

3.6: I beni pubblici

La quasi totalità di ciò che viene prodotto su Internet ha le caratteristiche tipiche di un bene pubblico, che sono beni di cui chiunque può fruire, senza distinzioni tra chi ha contribuito alla sua produzione e chi invece no. Esistono quindi due elementi che definiscono un bene pubblico:

- l'assenza di rivalità nel consumo, cioè il consumo da parte di una persona non riduce la quantità disponibile per gli altri;
- l'assenza di escludibilità dal consumo del bene da parte di chi non ha contribuito alla sua produzione.

Nella maggior parte dei casi, un bene pubblico presenta le suddette caratteristiche con diversi gradi di intensità, mentre nei beni pubblici puri sono entrambe presenti. Tutti i membri di un gruppo ricavano vantaggi dall'offerta di un bene pubblico, ma non vi è certezza che questo venga prodotto in quanto l'impossibilità di escludere dal suo utilizzo chi non vi ha contribuito determina una forte tentazione a comportarsi da free rider.

Offrire beni pubblici comporta due sfide: la prima è la motivazione, in quanto bisogna riuscire a far contribuire gli individui all'offerta di un bene pubblico nonostante la tentazione di comportarsi da free-rider. Questa tentazione deriva almeno da due fattori: il desiderio di trarre vantaggio dallo sforzo altrui, e la sensazione, pur

volendo collaborare, di perdere il proprio tempo a contribuire all'offerta di un bene che non verrà mai realizzata. La seconda sfida riguarda la coordinazione: anche se un gruppo di individui è motivato a creare un bene pubblico, è necessario coordinare i loro sforzi e spesso questo conduce a costi (Kollock, 1999).

Come è stato messo in evidenza in alcuni recenti contributi (Arvidsson, 2005; Zwick, Bonsu, & Darmody, 2008) il consumo si caratterizza sempre più come un vero e proprio lavoro immateriale. Per "lavoro immateriale" Lazzarato (1997, cit. in Arvidsson, 2005) intende due diversi aspetti del lavoro: da un lato per quanto riguarda il "contenuto informativo" dei beni, si riferisce ai cambiamenti che stanno avendo luogo nei processi produttivi, dove al lavoratore viene richiesta la capacità di saper gestire le tecnologie informatiche, che consentono la comunicazione orizzontale e verticale. Dall'altro lato, per quanto riguarda l'attività di produzione del "contenuto culturale" dei beni, il lavoro immateriale comporta una serie di attività, come la definizione di standard culturali ed artistici, mode etc. che non sono normalmente riconosciute come "lavoro" . Questo concetto deriva dalla nozione marxiana di "lavoro vivente" e si riferisce all'idea che gli individui siano in primo luogo dei lavoratori, non solo per qualcun altro. Le persone lavorano nel senso che esse creano la sostanza ed il significato della propria vita quotidiana (intelletto generale) senza riguardo per il loro ruolo come lavoratori dipendenti, autonomi etc. Quindi, secondo questa prospettiva, la dimensione più intima ed essenziale degli esseri umani è quella lavorativa, nel senso della produzione di qualcosa di valore per sé stessi e per la comunità in cui vivono (Cova & Dalli, 2007).

Il marxismo classico non riconosceva inizialmente alla circolazione dei beni, in cui il consumo si inserisce, la capacità di produrre valore. Marx mantiene infatti distinto il valore d'uso da quello di scambio. In realtà, grazie all'importanza dell'informazione nella *information economy* il bene oggetto della transazione, l'informazione, non viene consumato ma viene trasferito, creando ulteriore valore (Arvidsson, 2005). Nella circolazione dell'informazione, che viene rielaborata nei vari passaggi, quindi si crea ulteriore valore, e si ha la coincidenza tra la sua produzione e la sua circolazione.

Uno dei più importanti esempi di lavoro immateriale è quello insito nel consumo, inteso come capacità di creare una relazione sociale all'interno della quale i beni assumono un significato; i consumatori producono un contesto di consumo, un sito critico in cui le identità, i confini ed i significati condivisi vengono forgiati. I beni di consumo diventano dei "linking devices" che rendono possibile la cristallizzazione di forme di comunità transitorie o neo-tribali. Essi producono un *commons* sotto forma di comunità, un'identità condivisa o anche un'esperienza di breve durata che accresce il valore d'uso del bene, creando un surplus etico (Arvidsson, 2005).

In questo senso, la comunità virtuale di consumo produce contenuti e svolge funzioni utili anche per l'impresa, come Verona & Prandelli (2006) rilevano:

- Aggregazione della domanda su scala globale: le comunità virtuali di consumo consentono di ricreare segmenti globali o sovranazionali per consumi iperspecializzati, in quanto permettono l'incontro virtuale di consumatori dispersi

geograficamente attorno ad ambiti di interesse spesso estremamente focalizzati, innescando quindi fenomeni di autosegnalazione e di autosegmentazione[26];

- Analisi del mercato: l'analisi (spesso netnografica) delle comunità virtuali di consumo consente di accedere in tempo reale ad una dimensione autenticamente sociali ed esperenziale della conoscenza di consumo, soprattutto per quei beni il cui significato (e quindi valore) dipende dai valori che ad essi vengono associati, attraverso meccanismi di attribuzione di senso collettivo[27].
- Intensificazione dei flussi di comunicazione con e tra i consumatori: le comunità virtuali di consumo possono diventare strumento di democrazia informativa, in virtù della quale gli utenti finali non vengono più marginalizzati dalle decisioni che portano alla definizione dei prodotti e si attendono di poter esprimere le proprie opinioni ed eventualmente le propie lamentele attraverso un word of mouse[28].
- Co-definizione dei valori associati al brand aziendale: i consumatori diventano protagonisti attivi nel processo di

[26] Questa funzione è svolta tipicamente dalle comunità virtuali di consumo a supporto dei processi di transazione, che nascono sulla Rete con la funzione di agevolare operazioni di commercio elettronico oppure di facilitare le attività di acquisto e di vendita di prodotti o servizi, così come il trasferimento delle informazioni agli stessi associate. In questo caso, le comunità virtuali possono consentire ai propri membri di consultarsi prima di effettuare l'acquisto di un determinato prodotto, oppure di organizzarsi collettivamente per l'acquisto a condizioni più vantaggiose, come nel caso dei gruppi di acquisto (Verona & Prandelli, 2006).

[27] Questa attività è in genere svolta dalle comunità di interesse, ovvero aggregati di individui determinati ad approfondire aree tematiche comuni, a scambiarsi informazioni su argomenti specifici, ad aggiornarsi e arricchire costantemente le rispettive conoscenze (Verona & Prandelli, 2006).

[28] I fenomeni di passaparola sono alimentati in genere dalle cosiddette comunità di relazione, formate da partecipanti determinati a creare un tessuto di interazioni robusto con gli altri utenti della Rete, ovvero a vivere l'ambiente virtuale come uno spazio per scambi sociali ricchi di significati (Verona & Prandelli, 2006).

creazione del significato del brand e all'interno della comunità si innesca una sorta di negoziazione tra questi significati[29].

- Fidelizzazione degli utenti: le cvc possono contribuire ad accrescere la fedeltà degli utenti attraverso la continua alimentazione e il progressivo consolidamento del valore percepito nella partecipazione alle interazioni promosse. In questo senso, per il singolo consumatore si vengono a creare delle barriere all'uscita dalla relazione con una specifica impresa direttamente proporzionali all'attività che egli associa non solo con al rapporto con l'impresa stessa, ma anche al complessivo sistema di relazioni sviluppato con gli altri utenti.

Nel caso della co-definizione del valore del brand, come modalità di impiego delle brand communities da parte delle imprese, è chiaro il tentativo delle stesse di far proprio il surplus etico prodotto dalle comunità di consumo. Il brand, per la sua particolare caratteristica di essere un "open-ended object", il cui significato viene continuamente rimodulato, si caratterizza per possedere una notevole mobilità, che il brand management cerca di canalizzare, rendendo possibile alla libertà dei consumatori di evolversi entro determinate direzioni.

Non ci si può infatti permettere che il brand venga dirottato ("brand hijack") dai consumatori, evento che si verifica quando essi fanno assumere al marchio un'evoluzione differente rispetto a quella

[29]Oltre alle comunità di relazione, contribuiscono alla co-definizione del brand aziendale anche le comunità di fantasia, che privilegiano la dimensione dell'interazione libera e la disponibilità al dialogo dei partecipanti rispetto alle loro specifiche competenze: i partecipanti mirano ad interagire per estendere l'ambiente virtuale, piuttosto che a competere per acquisire potere al suo interno. Ciò che conta è quindi la disponibilità dell'individuo a creare nuovi ambienti e storie, ad assumere nuove identità virtuali e ad alimentarla nel tempo, mantenendo in piedi un mondo parallelo rispetto a quello reale (Verona & Prandelli, 2006)

ideata dai marketers e che può assumere due diverse forme (Wipperfürt, 2005 in Cova & Pace, 2006):

- Dirottamento serendipico (serendipitous hijack) in cui viene assunto il controllo del brand ed il suo significato, solitamente da parte di fanatici del brand appartenenti ad una sottocultura, ed è un atto in buona parte imprevisto ed imprevedibile per i marketers dell'impresa;
- Dirottamento co-creato (co-created hijack) in cui c'è l'invito alle sottoculture a contribuire alla co-creazione del brand, rendendo possibile l'adozione del nuovo significato co-creato anche da parte della massa.

Nella sua forma attuale, il brand management, quindi, riconosce l'autonomia dei consumatori, mirando ad offrire un ambiente, in cui programma ed anticipa l'agency dei consumatori.

Gli esempi di questa strategia sono notevoli, specie su Internet dove gli spazi virtuali creati dall'impresa per catturare il surplus etico prodotto dalle comunità di consumatori sono spazi web lasciati a disposizione degli utenti; tra questi si può citare il caso della "my Nutella the community" (Cova & Pace, 2006), della comunità creata per il restyling della Fiat 500 (Cucco & Dalli, 2008) o della Ducati per lo sviluppo di nuovi prodotti (Verona & Prandelli, 2006, pp. 120-124).

L'impresa riesce a convertire il surplus etico in valore reale attraverso due meccanismi: il primo è il maggior premium price riconosciuto al brand di successo, mentre il secondo è il maggior valore azionario riconosciuto all'impresa titolare del marchio, valore che viene ottenuto attraverso la capitalizzazione di assets quali il "brand awareness" (quanti consumatori conoscono il marchio), il

"brand associations" (se il brand origina associazioni mentali positive nel consumatore) e la "brand loyalty" o fedeltà alla marca (Arvidsson, 2005).

A questo punto si realizza un doppio sfruttamento dei consumatori: da un lato, dopo aver assunto un ruolo fondamentale nell'innovazione, e nello sviluppo del prodotto, essi come unica ricompensa hanno un prodotto ritagliato sulle proprie preferenze, per il quale però non hanno ottenuto nessun compenso, che invece viene riconosciuto ai lavoratori. Dall'altro lato, in virtù del maggior valore assunto dal brand, essi si ritroveranno a pagare un maggior premium price (Cova & Dalli, 2007).

Questa situazione può essere analizzata da un duplice punto di vista, attribuendole significati contrapposti: si realizza, infatti, la previsione pessimistica di Holt (2002) e Kozinets (2002), laddove essi ipotizzano l'impossibilità per il consumatore di evadere dal mercato, visto che i beni pubblici da esso creati vengono fatti propri dall'impresa, ma dall'altra si realizzano, in misura maggiore nelle situazioni in cui la resistenza creativa del consumatore dà luogo a vere e proprie innovazioni di prodotto, nuove fonti esterne all'impresa per la propria innovazione.

In questo senso diventa interessante comprendere innanzitutto perché i consumatori (o utenti che dir si voglia) decidono di contribuire alla realizzazione di un bene pubblico senza ricevere, apparentemente, una remunerazione; comprendere come tali contributi possono essere organizzati al fine di realizzare un'attività produttiva, e quali soluzioni possono essere adottate per consentire lo sfruttamento economico del bene prodotto, senza che questo comporti il suo

"hijack"o free riding, con specifico riferimento al caso del free/libre e open source software.

Capitolo 4

Aspetti istituzionali del FLOSS

4.1 Il ruolo dei consumatori nello sviluppo di innovazioni commerciali

Nel capitolo precedente è stato evidenziato il ruolo del consumatore nella produzione, prevalentemente simbolica, legata, ad esempio, alla rielaborazione del significato del brand.

In realtà la creatività del consumatore va ben oltre, in quanto esistono diversi casi documentati di innovazioni sviluppate da consumatori. In particolare, nel campo delle attività legate al tempo libero, diversi nuovi prodotti commercializzati con successo sono stati sviluppati dagli stessi utilizzatori di questi beni. Gli esempi sono numerosi, e vanno dalle barrette di muesli, agli sport drinks, alle mountain bikes, video games, apparecchiature fotografiche, per arrivare ai prodotti da forno.

Oltre a questi casi aneddotici, ci sono diversi studi empirici che si occupano dell'importanza delle attività innovative poste in essere dai consumatori/utilizzatori finali. I risultati a cui tali studi sono giunti sono diversi. Lawton e Parasuraman (1980, cit. in Lüthje, 2000), in uno studio comparativo tra diversi mercati, mettono in evidenza come il 12,7% dei processi innovativi venga stimolato dagli utilizzatori; nell'ambito dell'elettronica di consumo, Utterback et al. (1976, *ibidem*) sostengono che il 32,1% delle innovazioni nel settore sia derivato da dettagliate richieste in tal senso da parte dei consumatori.

Shah (2000), invece si è occupata delle fonti di innovazione nelle attrezzature relative a tre sport di recente introduzione:

skateboard, windsurf e snowboard. Nello studio vengono analizzate circa 57 innovazioni che hanno caratterizzato le attrezzature di questi sport nel corso degli ultimi 40 anni, e si sottolinea come l'invenzione di questi sport sia sempre dovuto ai primi appassionati che li hanno praticati, e come, successivamente, il 58% delle innovazioni più importanti nelle attrezzature necessarie per praticarli sia attribuibile ai cd. users e agli users-manufactures.

I consumatori quindi innovano, anche se nella maggior parte dei casi si tratta di miglioramenti di prodotti già esistenti, o meglio, di innovazioni incrementali.

Tali innovazioni nella maggior parte dei casi sono state indotte da consumatori che presentavano le caratteristiche di *lead users*, definiti da Von Hippel (1988) come quei consumatori che, rispetto ad un prodotto nuovo o rinnovato:

- Rilevano bisogni che saranno avvertiti dal mercato solo dopo mesi o anni;
- Si caratterizzano per ottenere benefici sostanziali dal soddisfacimento di tali bisogni.

Le caratteristiche che un lead user possiede lo rendono molto interessante agli occhi dei marketers, in quanto egli detiene quella conoscenza della realtà e del prodotto tale da consentirgli di anticipare quali saranno le tecnologie, le tendenze etc che muteranno la domanda del prodotto. In secondo luogo, il lead user che si attende elevate rendite dalla risoluzione di un problema legato all'uso di un prodotto può essere altamente motivato a cercare da sé una soluzione al problema.

Il lead user è quindi un consumatore che è alla ricerca di un prodotto adattato alle sue esigenze (o *customizzato*) e che, teoricamente, può far fronte a tale bisogno attraverso una scelta tra make e buy (Von Hippel, 2005). Infatti egli potrebbe rivolgersi ad imprese specializzate nella realizzazione di prodotti su misura, incorrendo però in costi di transazione. Se un consumatore "assume" un produttore per la realizzazione di un nuovo prodotto, egli diventa un principale che ha assunto un agente: se l'interesse del principale non coincide con quello dell'agente, si creeranno dei costi di agenzia, consistenti nel

1. Costo per monitorare l'agente, al fine di assicurarsi che egli agisca nell'interesse esclusivo del principale;
2. Costo sostenuto dall'agente per impegnarsi a non agire contro l'interesse del suo principale;
3. Costo associato ad un risultato che non va interamente incontro alle esigenze del principale.

Di fondo, esiste sempre una fondamentale divergenza di interessi tra il principale e l'agente nello sviluppo di beni e servizi: l'utilizzatore vuole ottenere esattamente ciò che desidera, mentre il produttore vuole ottenere un prodotto utilizzando soluzioni già note o che potrebbero servire una vasta platea di consumatori, in modo da minimizzare il più possibile il costo dello sviluppo.

Oltre ai costi di transazione, esiste un altro fattore, non meno importante, nel determinare la scelta del lead user verso il "make", ovvero il piacere, l'apprendimento ed il divertimento legati allo sviluppo di un'innovazione.

Un altro aspetto interessante è dato dal fatto che gli user-innovator spesso diffondono liberamente le proprie invenzioni (Von Hippel, 2005). Esistono, in questo senso, secondo Harhoff, Henkel, & Von Hippel (2003) che hanno testato empiricamente in cinque casi di studio[30] le loro ipotesi, diverse motivazioni alla libera diffusione delle innovazioni introdotte dagli utilizzatori:

1. *Indurre miglioramenti da parte dei produttori*: rivelando un'innovazione di prodotto o di processo, l'utilizzatore finale consente al produttore di adottarla e migliorarla. Quando la versione migliorata del prodotto sarà offerta sul mercato, egli avrà la possibilità eventualmente di ottenere ulteriori benefici se tale versione migliorata conterrà anche le innovazioni prodotte da altri. In genere il produttore, poi, offre anche una serie di servizi, come l'assistenza post-vendita che, nel caso in cui l'innovatore non avesse divulgato la propria invenzione, avrebbe dovuto assicurarsi da sé.
2. *Stabilire uno standard favorevole per l'utilizzatore*: divulgando liberamente la propria invenzione, l'utilizzatore consente ad altri la sua adozione, rendendolo uno standard. I vantaggi per l'innovatore possono essere legati a due fattori: la possibilità che l'innovazione si basi su fattori produttivi o vantaggi competitivi distintivi rispetto agli altri e che, al contempo, l'avvenuta adozione della propria innovazione come standard

[30] I casi di studio erano relativi ad innovazioni diffuse liberamente da IBM nel settore dei semiconduttori di rame, alla Technicon nel settore delle attrezzature per le analisi cliniche, a quelle liberamente distribuite dalle biblioteche australiane relative ai cataloghi elettronici delle risorse (OPAC), ai sistemi operativi Linux e Apache nel settore del software, ed in ciascuno di essi gli Autori hanno riscontrato la presenza di una o più delle motivazioni sopra riportate a sostegno della scelta della disclosure dell'innovazione.

impedisca la diffusione di altre soluzioni, sulle quali non avrebbe alcun vantaggio competitivo[31].

3. *Reciprocità ed effetti sulla reputazione*: rivelando un'innovazione, l'innovatore può creare o adempiere ad obbligazioni di reciprocità generalizzata, e può guadagnarci in reputazione: i maggiori benefici, in questo caso, vanno a chi per primo diffonde l'innovazione.
4. *Condizioni di ridotta rivalità*: quando la competizione tra gli utilizzatori è ridotta, ad esempio a causa della separazione geografica tra i rispettivi mercati di riferimento, chi rivela l'innovazione non sopporta alcun conseguenza negativa dai vantaggi prodotti a favore degli altri. Vi è quindi l'assenza di disincentivi alla diffusione dell'innovazione.

4.2 Le comunità di innovazione

Come evidenziato precedentemente, gli utilizzatori rivelano facilmente le proprie innovazioni ed informazioni in generale. Essi possono essere dispersi geograficamente, o avere solo poche innovazioni o idee da condividere: in questo senso, l'utilità di queste innovazioni liberamente distribuite, che costituiscono a tutti gli effetti un bene pubblico, può essere accresciuta se le informazioni vengono rese facilmente accessibili a terzi.

In questo senso, una funzione determinante è assunta dalle comunità di innovazione, definite da Von Hippel (2005) come "nodi di significato, composti da individui o imprese interconnesse tramite legami per il trasferimento di informazioni, comportanti il ricorso ad

[31] Questo risultato può essere ottenuto, ovviamente, anche con la concessione in licenza d'uso delle tecnologie, ed è stato per esempio il caso della Nintendo e della Matsushita nel settore dei videogames e dei videoregistratori (Arora, Fosfuri, & Gambardella, 2001, p. 177-178)

una comunicazione face-to-face, elettronica etc. Possono, ma non necessariamente devono, esistere entro i confini dell'appartenenza ad un gruppo. Spesso possono, ma non necessariamente devono, incorporare le qualità di una comunità di partecipanti, in cui per comunità si intende un network di legami interpersonali che offre socialità, sostegno, senso di appartenenza ed identità sociale".

Le comunità di innovazione possono avere tra i propri membri sia utilizzatori che produttori, e possono prosperare quando almeno un membro innova e rivela volontariamente la propria innovazione, e quando gli altri appartenenti alla comunità trovano tale informazione di proprio interesse.

Esse sono spesso specializzate, in quanto possono servire come punti di raccolta o depositi di informazioni relative a ristrette categorie di innovazioni, e possono offrire altre interessanti funzionalità, utili alla diffusione delle informazioni, come chat rooms o mailing lists attraverso le quali i membri della comunità possono partecipare ai vari progetti offerti.

Le comunità di utenti sorte attorno ai progetti free/libre e open source sono tipici esempi di comunità di innovazione e rappresentano una novità assoluta rispetto ai precedenti modelli di sviluppo della conoscenza, essenzialmente firm-based (Lee & Cole, 2003).

La letteratura economica, infatti, per molto tempo ha considerato come unità di analisi, nell'ambito dei processi di creazione della conoscenza e dell'innovazione, l'impresa o gruppi di imprese, in riferimento ai processi di creazione della conoscenza e ha ritenuto fondamentale, per il trasferimento della conoscenza, soprattutto quella implicita e non codificata, l'interazione face-to-face

al fine di favorire la costruzione fiducia reciproca e la condivisione di uno spazio fisico comune (Nonaka & Takeuchi, 1997). Inoltre, considerando l'impresa come unità di analisi nei processi di creazione dell'innovazione, si è implicitamente ipotizzato che la creazione dell'innovazione potesse avvenire unicamente nel contesto dell'organizzazione gerarchica dell'impresa (Lee & Cole, 2003).

Tabella 4.1: I modelli firm-based e community-based a confronto

Principi organizzativi	Modello firm-based	Modello community-based
Proprietà dei diritti industriali	La conoscenza è privata e di proprietà dell'impresa	La conoscenza è pubblica e può essere di proprietà dei membri che la condividono e vi contribuiscono
Restrizioni alla membership	L'appartenenza al gruppo è basata sulla selezione, e la dimensione dell'impresa è condizionata dal numero di dipendenti	L'adesione al gruppo è libera
Autorità ed incentivi	I membri dell'impresa sono dipendenti che ricevono un salario in cambio del proprio lavoro	I membri della comunità sono volontari che non ricevono paga in cambio del loro lavoro
Distribuzione della conoscenza lungo i confini geografici ed organizzativi	La distribuzione è limitata dai confini dell'impresa	La distribuzione si estende oltre i confini dell'impresa
Modalità dominanti di comunicazione	Interazione face-to-face	Interazione mediata dalla tecnologia

Fonte: Lee & Cole, 2003

Nella tabella 4.1 vengono evidenziate le principali differenze esistenti tra il modello *firm-based* e quello *community-based* di creazione della conoscenza, da cui si può ricavare come in quest'ultima le assunzioni che sono alla base del modello di creazione della conoscenza basato sull'impresa vengano violate a causa di:

1. un sistema di assegnazione dei diritti sulla proprietà industriale finalizzato alla creazione della fiducia e alla condivisione della conoscenza;
2. libero ingresso nella comunità, avente quindi dimensioni molto più ampie rispetto all'impresa;
3. motivazioni ed incentivi che non sono più quelli di un lavoratore dipendente, ma di volontari che contribuiscono al progetto in assenza di un'autorità che regola il loro comportamento;
4. individui che sono dispersi sia da un punto di vista geografico che da un punto di vista organizzativo;
5. una piattaforma di creazione della conoscenza basata su sistemi di comunicazione *many-to-many*.

4.3 Il modello privato di creazione dei beni pubblici

I due diversi modelli proposti da Lee & Cole (2003) sono, in altri termini, i due modelli di innovazione tra loro antitetici proposti da Von Hippel & Von Krogh (2003): il modello di investimento privato (*private investment model*), e il modello dell'azione collettiva (*collective action model*).

Nel primo modello, l'innovazione sarà sostenuta dagli investimenti privati e i profitti ottenuti saranno appropriati da chi ha effettuato tale investimento. Affinchè ciò avvenga, è necessario che la società assicuri agli innovatori attraverso i meccanismi di tutela della proprietà intellettuale ed industriale dei diritti, di durata limitata, per lo sfruttamento esclusivo dei benefici di tali innovazioni, come con i

brevetti[32], il diritto d'autore, etc. In questo modello, qualsiasi rivelazione gratuita o spillover della conoscenza proprietaria riduce il profitto dell'innovatore.

Il secondo modello è quello dell'azione collettiva, che si applica all'offerta dei beni pubblici, modello che richiede il rilascio incondizionato della conoscenza prodotta da parte dei contributori, che diventerà un bene pubblico, andando a costituire un *common pool*. Con questo modello si evitano i problemi connessi al ridotto accesso all'innovazione prodotto dal modello privato, ma al contempo sorgono problemi in merito alla motivazione dei contributori.

Un classico problema dei beni pubblici è quello del free rider (Olson, 1983), ovvero della presenza di soggetti che vogliono avvantaggiarsi del bene collettivo senza sostenerne il costo, da cui deriva il problema di motivare adeguatamente i singoli a contribuire anziché fruire gratuitamente del bene.

Nel caso dei gruppi eccessivamente numerosi, poi, esisterebbero tre fattori che impediscono loro di promuovere i loro stessi interessi (Olson, 1983):

1. Tanto più numeroso è il gruppo, tanto più piccola è la frazione del beneficio complessivo che ogni individuo, il quale agisca nell'interesse del gruppo, riceve, e quanto meno la ricompensa per ogni azione di gruppo è adeguata, tanto più il gruppo è incapace di ottenere una fetta vicina a quella ottimale del bene collettivo, anche nel caso in cui ne ottenga una qualche quantità;

[32] Cfr. par. 2.6 per un rapido excursus sulla letteratura in tema di efficacia della tutela brevettuale.

2. Quanto più è numeroso il gruppo, tanto meno probabile è l'interazione oligopolistica che potrebbe aiutare a conseguire il bene stesso;
3. Maggiore è il numero di componenti del gruppo, maggiori saranno i costi dell'organizzazione.

A causa dei motivi sopra elencati, Olson (1983) conclude che l'eccessiva numerosità di un gruppo determina la sua incapacità di procurarsi un bene collettivo in quantità ottimali. E' solo nei gruppi intermedi che la presenza di incentivi distinti e selettivi consente la produzione di beni pubblici.

Il gruppo intermedio si caratterizza per il fatto che nessun singolo membro riceve una fetta di beneficio sufficiente ad incentivarlo a procurarsi da sé il bene stesso, ma che non ha tuttavia un numero di membri talmente elevato da far sì che nessuno dei membri si accorga se un altro membro concorre o meno a procurare il bene collettivo. Un simile gruppo può ottenere o non ottenere il bene collettivo, ma non è mai in grado di conseguire alcun bene collettivo senza una qualche misura di coordinamento o di organizzazione. Soltanto un incentivo distinto e selettivo sarà in grado di stimolare un individuo razionale ad un'azione collettiva. L'incentivo deve essere selettivo nel senso che esso deve consentire un trattamento differenziato nei confronti di chi non contribuisce al bene collettivo rispetto a chi invece lo fa.

Inoltre, in un gruppo più piccolo, è più facile monitorare il comportamento dei singoli membri ed eventualmente sanzionare le loro defezioni.

Tutte queste caratteristiche tipiche dell'azione collettiva non si applicano, secondo Von Hippel & Von Krogh (2002) al software open source; innanzitutto è possibile vedere come, in linea di massima, i project leader non danno luogo ad alcuna attività di reclutamento dei contributori, in quanto la richiesta di aiuto nella realizzazione di un progetto viene semplicemente postata su siti (come FreshMeat e SourceForge) o mailing lists. Poi la dimensione del gruppo, specie in alcuni progetti di successo, sembrerebbe non avere alcuna rilevanza sulla capacità di realizzare il progetto: non si realizza la cosiddetta "legge di Brooks", per cui aggiungere altri programmatori ad un progetto in ritardo, lo si fa ritardare ancora di più a causa dell'aumento dei costi legati alla maggiore complessità e alle maggiori difficoltà nella comunicazione che un maggior numero di sviluppatori comporta: l'aumento del numero degli sviluppatori determina l'aumento esponenziale dei costi di comunicazione e di coordinamento, mentre la quantità di lavoro realizzata aumenta in modo lineare (Raymond, 1999).

Infine, il free-riding non è assolutamente sanzionato, neanche moralmente: chiunque può liberamente scaricare il software, utilizzandolo nell'ambito della specifica licenza open source che lo protegge (cfr. capitolo 2).

Al contempo, il software open source si differenzia anche dal modello privato di produzione dell'innovazione innanzitutto perché gli innovatori sono utilizzatori e non produttori industriali di programmi elettronici, e poi perché essi rivelano liberamente il software che essi hanno creato.

La spiegazione che Von Hippel & Von Krogh (2003) danno, in merito a questi scostamenti del modello di produzione della conoscenza offerto dal software open source, è che in realtà esso è un modello privato di produzione di un bene pubblico, in quanto esistono delle ricompense private, riconosciute a chi contribuisce attivamente alla realizzazione di un progetto open source, che non sono riconosciute a chi opera da free rider.

Il modello open source, quindi, secondo Von Hippel e Von Krogh (2003), si colloca a metà strada tra il modello privato e quello collettivo di produzione della conoscenza, attraverso:

- L'eliminazione dell'assunto del modello dell'investimento privato per cui la rivelazione gratuita dell'innovazione sviluppata con l'investimento di risorse private determina una perdita del profitto (privato) di chi vi ha contribuito, in quanto, sotto determinate condizioni, la rivelazione dell'innovazione produce benefici privati;
- L'eliminazione dell'assunzione tipica del modello dell'azione collettiva per cui il free rider è in grado di appropriarsi degli stessi benefici di chi contribuisce attivamente al progetto: in realtà esistono degli incentivi selettivi a favore degli sviluppatori più attivi nel contribuire al software.

I benefici privati attesi dagli sviluppatori che contribuiscono ad un software open source possono essere diversi; si pensi ad esempio al problem solving che un programmatore deve affrontare nella scrittura di codice: questo può essere fonte di divertimento ed apprendimento, e di un senso di appartenenza del software creato decisamente maggiore rispetto a quanto avvertito nell'elaborazione di software proprietario

per conto di imprese (Lakhani & Wolf, 2005). Un'altra possibile determinante dei benefici privati è legata alla possibilità per i programmatori di offrire codice al progetto che, al contempo, viene utilizzato dallo stesso sviluppatore per le proprie necessità e talvolta con finalità diverse, avendo però al contempo la possibilità di vedere il programma testato da terzi e da questi eventualmente migliorato.

Un altro beneficio privato è legato alla partecipazione alla comunità e alla cooperazione che si crea con gli altri sviluppatori, cooperazione i cui vantaggi, se essa è intensa e continua, possono superare i vantaggi privati conseguiti dai singoli sviluppatori (Von Hippel & Von Krogh, 2003)

Il modello sopra esposto solleva tre interrogativi, a cui la letteratura economica ha cercato di rispondere attraverso l'analisi di casi empirici e l'elaborazione di modelli, e precisamente:

1. Perché e come gli sviluppatori decidono di partecipare o abbandonare un progetto open source, e l'emersione di eventuali categorie sociali in tali progetti.
2. Se e come i partecipanti ai progetti vengono integrati nelle comunità.
3. Quali forme assume l'organizzazione del lavoro offerto volontariamente dai programmatori.

4.4 Le motivazioni degli sviluppatori

Uno dei temi legati all'open source più affrontati nella letteratura riguarda senza dubbio il perché alcuni programmatori,

spesso di notevole capacità, contribuiscano gratuitamente all'offerta di un bene pubblico come il software open source[33].

Hars & Ou (2001) sono stati tra i primi ad affrontare il tema e facendo riferimento a Deci (1975, cit in Hars & Ou, 2001) creano una prima tassonomia delle possibili motivazioni sottostanti la partecipazione ad un progetto open source distinguendole in:

- Motivazioni intrinseche, che derivano dall'innato desiderio delle persone di sentirsi competenti ed in grado di essere determinanti nel proprio ambiente, nonché dal desiderio di porre in essere qualcosa che sia interessante o divertente o sfidante (Bitzer, Shrettl, & Schroder, 2007). Esse, nella definizione di Ryan & Deci (2000), consistono nella tendenza innata a ricercare novità e sfide, ad estendere ed esercitare le proprie capacità, esplorare ed imparare.
- Motivazioni estrinseche, che fanno riferimento alla possibilità di ottenere un compenso monetario, diretto o indiretto, così come al riconoscimento delle proprie capacità da parte dei propri pari.

Secondo Bitzer, Shrettl, & Schroder (2007), nel caso del software open source, sarebbe possibile identificare tre diverse motivazioni intrinseche sottostanti la partecipazione ad un progetto open source, soprattutto nel caso degli iniziatori del progetto:

1. La necessità di un software come soluzione ad un problema;
2. Il divertimento insito nella programmazione, ovvero lo sviluppatore come *homo ludens*;
3. Il desiderio di fare un dono alla comunità.

[33] E' l'interrogativo che si pongono Lerner & Tirole (2002):"Why should thousands of top-notch programmers contribute freely to the provision of a public good?"

L’altruismo, quindi, visto come il desiderio di fare qualcosa di utile per gli altri sostenendone i costi, è quindi visto come una componente motivazionale, e la cultura hacker sarebbe una vera e propria *gift culture* (Zeitlyn, 2003): il dono e l’obbligo di ricambiare creano delle vere e proprie relazioni sociali, e inoltre, nel caso specifico dei programmatori, l’aver contribuito più volte ad un software accresce la loro reputazione o, detto in altri termini, crea un capitale simbolico (Bourdieu, cit. in Zeitlyn, 2003).

Un'altra motivazione intrinseca è l’identificazione con la comunità: i programmatori si identificano con la comunità e allineano i propri desideri con quelli del gruppo.

Hertel, Niedner, & Herrmann (2003) analizzano le motivazioni sottostanti la partecipazione degli utenti allo sviluppo di Linux, partendo dall’assunto che, per le proprie caratteristiche, il movimento open source sia, a tutti gli effetti, un movimento sociale.

Questa similarità consente, nell’analizzare in dettaglio le motivazioni dei partecipanti alla comunità di Linux, di mutuare un tipico modello esplicativo, utilizzato nel contesto, appunto, dei movimenti sociali, quale il modello di Klandermans. In base a questo modello, la partecipazione a movimenti sociali dipende da diversi costi e benefici attesi, classificabili in diverse classi.

Una prima classe è definita come *collective motives* (motivazioni collettive): essa si basa sulla valutazione degli obiettivi del movimento, che vengono ponderati con la probabilità con cui tali obiettivi verranno raggiunti. In base a questo approccio, la partecipazione sarà tanto più elevata quanto maggiore sarà la

valutazione attribuita agli scopi della comunità e quanto più la persona percepisce come probabile il loro raggiungimento.

La seconda classe è quella dei *social motives* (motivazioni sociali) e contiene le reazioni attese di altri, importanti per il partecipante, come gli amici o i familiari. La motivazione a partecipare ad un movimento sociale sarà tanto maggiore quanto più positiva sarà la reazione di tali *significant others*, ponderata con l'importanza che il partecipante attribuisce ad essi.

La terza classe di motivi include i *reward motives*, che derivano dai costi e benefici attesi come gli investimenti in tempo e denaro, la possibilità di crearsi nuove amicizie etc.

Alle tre classi di motivazioni alla partecipazione ai movimenti sociali del modello di Klandermans, Hertel et al. (2003) ne aggiungono una quarta, legata all'*identificazione collettiva*, in base alla quale i partecipanti ad un movimento sociale si sentono e si definiscono come membri di un sottogruppo che appartiene alla comunità, e si comportano secondo le norme e gli standard di tale gruppo. Il processo di identificazione con un determinato sottogruppo sarebbe una determinante della partecipazione molto più forte rispetto all'identificazione con l'intera comunità. Per esempio, l'identificazione come attivista femminista rappresenta una motivazione alla partecipazione all'azione collettiva nei movimenti femminili rispetto all'identificazione come donna. Sembra quindi che quanto più ci si identifica con sottogruppi attivi di un movimento, tanto maggiore sarà il desiderio di contribuire personalmente. Si può ragionevolmente ipotizzare che questi processi possano essere applicati ai vari progetti open source.

Il secondo modello utile per la comprensione delle motivazioni alla partecipazione è il modello VIST. Viene usato per la comprensione dei processi motivazionali nei team di lavoro delocalizzati o "team virtuali", i cui membri lavorano in luoghi differenti e coordinano i loro sforzi per mezzo di media elettronici.

Lo sviluppo del software open source avviene grazie ad Internet, senza un'interazione face-to-face tra i contributori, ed è in questo senso virtuale. Però, a causa dell'alto numero di partecipanti e della facilità di accesso, i progetti OSS sono in genere concepiti come basati sulle comunità o su network collaborativi, piuttosto che come team. Le comunità solitamente includono un gran numero di persone, e sono aperte a chiunque voglia unirsi ad esse, purchè vengano rispettate delle regole generali di comportamento. I network collaborativi sono più restrittivi nella loro politica di accesso, che è basata sulla reputazione, e sviluppano un codice di comportamento comunitario molto più specifico, che prevede forme di sanzione per chi lo viola. I loro confini sono tuttavia abbastanza permeabili, in quanto permettono un frequente scambio di collaboratori. Per team si intende un gruppo di collaboratori relativamente piccolo (da 2 a 20), avente confini, norme, funzioni e ruoli chiari e relativamente stabili. Nonostante, come detto prima, Linux possa essere considerata una comunità, è altamente probabile che al suo interno possano essere individuati dei team, corrispondenti ai team del progetto che si formano spontaneamente per lavorare a ciascuna della specifiche parti di Linux, i cosiddetti moduli. I moduli sono le diverse parti in cui Linux può essere scomposto, e rappresentano parti per molti versi autonome del kernel; esempi di moduli sono quelli relativi alle

interfacce usb, alle interfacce fire-wire, nonché alle interfacce grafiche come GNOME e KDE. Ognuno di questi moduli viene testato ed inglobato in Linux dopo l'approvazione del project maintainer, in base all'architettura complessiva del programma e in modo da garantirne la stabilità. Attorno a ciascuno di questi moduli lavorano dei sottogruppi o team di programmatori; in questo senso, l'utilizzo del modello VIST può essere utile.

In base al modello VIST, le motivazioni individuali che spingono a lavorare in un team virtuale sono quattro, precisamente *valence*, *instrumentality*, *self-efficacy*, e *trust*. La *valence* è definita come la valutazione soggettiva degli obiettivi del team, come nel caso dei collective motives del modello di Klandermans. Come in quest'ultimo, la motivazione a partecipare è correlata positivamente ad una maggiore valutazione degli obiettivi. La differenza rispetto al modello di Klandermans è nel fatto che nella valenza gli obiettivi a cui si fa riferimento non sono quelli della comunità, ma quelli più specifici di un team, corrispondente ad un eventuale sottogruppo.

L'*instrumentality* è definita come l'importanza o l'indispensabilità percepita del contributo dato dal singolo ai risultati del gruppo. Maggiore sarà l'importanza percepita dal soggetto del contributo da lui dato, maggiore sarà la motivazione ad offrire il proprio lavoro al gruppo.

La *self-efficacy* fa riferimento alla capacità percepita di saper porre in essere le attività richieste per la realizzazione dei compiti. In pratica, la self-efficacy fa riferimento alla possibilità percepita che il proprio contributo conduca il team ad elevate performances. Se un membro del gruppo ritiene di non sapere compiere i compiti assegnati,

la sua motivazione sarà minore, anche se attribuisce un elevato valore ed il suo contributo percepito come necessario al raggiungimento del successo del gruppo.

Infine la *trust* o fiducia indica le aspettative dei membri del gruppo circa il fatto che il proprio contributo sarà contraccambiato e non sfruttato dagli altri (*interpersonal trust* o fiducia interpersonale) e che il sistema elettronico di supporto funzioni in modo affidabile (*trust in the system* o fiducia nel sistema).

Si ipotizza che ciascuna delle quattro componenti influenzi in modo positivo la motivazione di ciascun membro a partecipare.

Hertel et al. (2003) hanno applicato i due modelli esplicativi della motivazione ad un campione di 141 partecipanti (6 donne e 135 uomini, età media 30 anni, con un range di 15-64), provenienti da 28 paesi diversi, di cui il 48% dal Nord America, il 37% dall'Europa e il 7% dall'Australia. Meno del 3% proveniva rispettivamente dall'Africa, dall'Asia e dal Sud America. Il 67% dei partecipanti era impiegato a tempo pieno, il 5% part-time e un altro 5% era disoccupato; il restante 23% era composto da studenti. I partecipanti hanno in media lavorato 17 mesi al kernel, con un range che variava da 1 a 98 mesi) Metà del campione (69 persone, il 49%) era attivamente impegnato nel processo di sviluppo del kernel e si autodefiniva "sviluppatore attivo del kernel Linux" o "maintainer" di uno specifico modulo.

Nell'ambito di questo campione, il modello di Klandermans è stato utilizzato per esplorare le motivazioni generali che hanno spinto le persone a contribuire alla comunità di Linux e, in questo senso, la partecipazione poteva consistere sia nell'offerta di parti del software

(in termini di numero di linee del codice o di risoluzione dei bugs, per esempio), sia contribuendo alla discussione e offrendo idee nelle mailing lists.

L'applicazione del modello ha comportato la somministrazione di questionari alle persone incluse nel campione in cui si chiedeva loro di indicare l'importanza, con un numero da 1 a 5, (1= nessun importanza, 5=massima importanza) da essi attribuita ad una serie di possibili motivi della loro partecipazione alla comunità di Linux.

In particolare, i *collective motives* ipotizzati sono stati: migliorare la qualità del kernel, migliorare la qualità del sottosistema (o modulo), ed un obiettivo più ampio, cioè creare un software libero. Nel caso dei *social* o *norm-oriented motives* si è chiesto al campione l'importanza che le persone percepite come *significant others* attribuivano alla loro attività nella comunità di Linux. E' stato poi chiesto di valutare i benefici e i costi derivanti dai *reward motives*, come: facilitare il lavoro quotidiano attraverso un software migliore, avere interscambi con altri programmatori, acquisire una reputazione come esperto sviluppatore di Linux, divertimento insito nella programmazione, migliorare le proprie abilità come programmatori, possibili vantaggi di carriera, perdite di tempo, assenza di pagamento per il loro contributo. Infine, per poter definire la rilevanza dell'identificazione (definibile anche come "we-ness"), è stato chiesto agli intervistati, quanto, in una scala da 1 a 5, si identificassero, rispettivamente, come utilizzatore di Linux, come sviluppatore di Linux, o quanto si identificassero con un certo sottosistema.

Le varie possibili motivazioni sono state poi riclassificate in sette possibili fattori esplicativi della partecipazione allo sviluppo di

Linux, comprendenti ciascuno alcune delle ipotetiche motivazioni indicate precedentemente:

1) Identificazione come sviluppatore di Linux o come sviluppatore di uno dei sottosistemi;

2) Identificazione come utilizzatore di Linux;

3) Motivazioni pragmatiche, comprendenti il miglioramento di Linux o di uno dei suoi sottosistemi, nonché la facilitazione del lavoro quotidiano con il programma e la possibilità di miglioramento della propria carriera lavorativa;

4) Motivazioni sociali o norm-oriented, comprendenti i social o norm-oriented motives;

5) Motivazioni edonistiche, consistenti nel divertimento legato alla programmazione;

6) Motivazioni socio-politiche, ovvero scambi personali con gli altri sviluppatori, guadagno in reputazione come esperto programmatore di Linux, miglioramento delle abilità nella programmazione, assenza di pagamento (correlata negativamente alla partecipazione), a cui si aggiunge la difesa della libertà del software;

7) Valutazione della perdita di tempo legata alla programmazione, correlata negativamente con la partecipazione.

I sette fattori motivazionali, a cui sono stati aggiunti due criteri di misurazione dello sforzo, come ore settimanali spese nello sviluppo di Linux e desiderio di essere coinvolti anche nel futuro, sono stati analizzati in due sottogruppi del campione, il gruppo degli sviluppatori (n=69) e il gruppo dei "lettori interessati" (n=72), distinti

in base alla tipologia di contributo da essi apportato al programma, rispettivamente linee di codice, risoluzione di bugs o patches per il primo, idee, pareri e discussione per il secondo gruppo. I risultati sono visibili nella tabella 4.2.

Come si può vedere dalla tabella, l'analisi ha messo in evidenza tre elementi interessanti: una più specifica identificazione come sviluppatore di Linux o di uno dei suoi sottosistemi, piuttosto che come utilizzatore; una notevole tolleranza rispetto alle perdite di tempo legate allo sviluppo del software; un interesse piuttosto pragmatico nei vantaggi personali derivanti dal miglioramento della qualità del kernel (pragmatic + hedonistic motives).

Tabella 4.2: Sviluppatori e modello di Klandermans, Hertel et al. 2002

Variabile	Sviluppatori (n=69)		Lettori interessati (n=72)		Differenza
	Media	Dev. St.	*Media*	Dev. St.	
Identificazione come utente	3,9	0,9	4,2	0,7	$p < 0,09$
Identificazione come sviluppatore	4,0	0,7	3,3	1,0	$p < 0,01$
Motivazioni sociali	3,9	0,8	3,6	0,9	$p < 0,06$
Motivazioni pragmatiche	4,3	0,5	4,1	0,8	$p < 0,07$
Motivazioni socio-politiche	4,1	0,4	4,1	0,5	n.s.
Motivazioni edonistiche	4,7	0,6	4,5	1,0	n.s.
Perdite di tempo	3,6	1,0	3,3	1,1	$p < 0,11$
Ore/settimane dedicate	18,4	18,0	6,6	9,2	$p < 0,001$
Volontà coinvolgimento futuro	4,5	0,8	3,9	1,1	$p < 0,001$

Note: La scala per tutti i valori – eccetto il tempo dedicato – varia da 1 a 5. Elevati valori della media indicano un'elevata identificazione, un notevole peso delle motivazioni sociali, di quelle pragmatiche, etc. I valori indicati nella differenza indicano il risultato del test "T" a due code applicato alle due serie di medie aritmetiche.
Fonte: Hertel, Niedner, & Herrmann (2003)

Un dato interessante è poi espresso dall'elevato valore dello scarto quadratico medio, indice di valori molto diversi nel campione, per quanto riguarda le ore settimanali dedicate allo sviluppo del kernel, che esprime una caratteristica peculiare di Linux e, in generale, dei software open source di successo, cioè la *granularità*. La granularità può essere definita, d'accordo con Benkler (2002), come la dimensione del modulo o, in generale, del contributo dato al programma, misurata in termini di tempo e sforzo investiti dal programmatore per realizzarlo. L'elevato sqm è indice di una granularità eterogenea, che permette alle persone aventi diversi livelli di motivazione di partecipare apportando contributi di diversa dimensione, coerenti con i loro livelli di motivazione.

Per quanto riguarda, invece, i livelli motivazionali all'interno di ciascun sottogruppo di lavoro, o team, il modello utilizzato è stato il VIST, in base al quale si ipotizza che la partecipazione ad un team di lavoro (virtuale e legato ad un modulo o sottogruppo di lavoro all'interno di Linux, in questo caso) dipenda da quattro diversi fattori, quali valence, instrumentality, self-efficacy e trust. A questi fattori ne sono stati aggiunti altri; due di questi misurano la motivazione dei partecipanti a lavorare per il progetto del sottosistema (o modulo), precisamente le ore settimanali spese nello sviluppo di Linux e la volontà di essere coinvolti anche nel futuro (misure dello sforzo); altri due, invece, misurano in modo oggettivo il risultato della partecipazione, e sono il numero di patches e il numero di linee di codice scritte (misure della performance). Per ciascuno degli otto fattori esplicativi sono state calcolate le medie (M) e lo sqm (S.D.), e sono state correlate fra loro (tab. 4.3)

Tabella 4.3: Sviluppatori e modello VIST (Hertel et al., 2003)

Variabile	1	2	3	4	5	6	7	8
Valence	-							
Instrumentality	0,19	-						
Self-efficacy	0,07	0,46	-					
Trust	0,21	0,03	-0,03	-				
Ore/settimane dedicate	0,19	0,34	0,21	0,19	-			
Aumento partecipazione	0,40	0,34	0,09	0,06	-0,02	-		
Numero di patches	0,05	0,35	0,36	0,06	0,32	0,09	-	
Linee codice (scala log.)	-0,13	0,08	0,32	-0,06	0,05	-0,33	0,31	-
Media	4,7	3,4	4,0	3,8	20,5	2,4	2,4	2,6
Dev.st.	0,5	1,0	0,6	0,8	19,4	1,0	1,6	0,9

Note: La scala per tutti i valorivaria da 1 a 5, eccetto che per le ore/settimane dedicate a Linux, per il numero di patches e per le linee di codice.
Fonte: Hertel, Niedner, & Herrmann (2003)

I primi tre fattori del modello VIST sembrano essere positivamente correlati fra di loro, mentre la componente trust ha una minore importanza. La regressione del tempo dedicato allo sviluppo sembra essere correlato in maniera significativa con il fattore instrumentality (beta=0,34): il tempo dedicato diventa maggiore se il partecipante ritiene che il suo contributo sia molto importante per lo sviluppo del sottosistema.

Il desidero di incrementare la propria partecipazione nel futuro, invece, risulta essere significativamente correlato con due fattori del modello VIST, ovvero valence (beta=0,40) e instrumentality (beta=0,34): gli sviluppatori desidereranno dedicare più tempo al sistema se maggiore sarà l'importanza attribuita agli obiettivi del sottosistema e se maggiore sarà l'importanza percepita del contributo che essi sapranno dare al progetto.

Il numero di patch offerte ed accettate è un valore che dipende dall'esperienza e dall'abilità del programmatore, e non a caso è correlato positivamente con le ore dedicate al progetto (beta=0,32). Questa misura della performance del programmatore è correlata in modo significativo con l'instrumentality (beta=0,35) e con la self-

efficacy (beta=0,36): un maggior numero di patch accettate è dovuto ad un'elevata self-efficacy e alla forte convinzione della crucialità del proprio contributo per il successo del sottosistema.

Il numero di linee di codice offerte (in scala logaritmica) è un altro indicatore della performance del programmatore che sembra essere significativamente correlato con la self-efficacy percepita, ovvero con la percezione della propria capacità di compiere efficacemente il compito assunto, mentre gli altri fattori del VIST sembrano non avere alcun legame rilevante con questa variabile.

L'analisi delle motivazioni, in base ai due modelli precedentemente esposti, ha delle implicazioni pratiche interessanti non solo per lo sviluppo di Linux e altri software open source, ma utili anche per le imprese informatiche che usano un approccio "bazaar-style". Un aspetto è l'importanza assunta dai processi di identificazione, che diventa sempre più rilevante man mano che si passa dall'analisi della comunità di Linux in generale, ai vari sottogruppi in cui essa si articola.

Interessante è poi l'influenza negativa sulla partecipazione derivante da eventuali perdite di tempo: in questo senso, l'articolazione del software in moduli a se stanti da coordinare in un'architettura semplice, nonché strumenti di coordinamento basati su sistemi di comunicazione in tempo reale possono aiutare a ridurre gli sprechi di tempo.

Le motivazioni pragmatiche, consistenti nel desiderio di migliorare un software utile per il proprio lavoro, nonché nel miglioramento delle proprie opportunità di carriera, sembrano essere delle importanti motivazioni per la partecipazione a progetti simili nel

futuro, anche se non hanno influito più di tanto nella partecipazione offerta in passato. Perciò aspettative irrealistiche sull'utilità personale ricavabile dalla partecipazione a progetti oss potrebbero incidere negativamente sul coinvolgimento in futuro e, quindi, potrebbe essere utile informare i partecipanti sui reali costi e benefici derivanti dalla propria attività nel progetto.

Infine, lo studio ha messo in evidenza come una buona parte del lavoro di sviluppo di Linux sia stato realizzato da team spontanei, che lavorano ciascuno ad uno o più sottosistemi del software. Il modello VIST ha consentito di mettere in evidenza come, analogamente a quanto avviene nei team virtuali all'interno delle imprese, l'indispensabilità percepita dei propri contributi al gruppo sia un importante fattore di motivazione, insieme ad un'elevata valutazione degli obiettivi del team e ad un'elevata self-efficacy.

Tab. 4.2: I Lugs in Italia

Regione	N. LUGs	%	Iscritti	%	Simpatizzanti	%
Italia	4	1,89%	293	5,13%	604	3,39%
Abruzzo	11	5,19%	161	2,82%	582	3,27%
Basilicata	2	0,94%	150	2,63%	200	1,12%
Calabria	10	4,72%	179	3,13%	825	4,64%
Campania	20	9,43%	516	9,04%	1168	6,56%
Emilia Romagna	16	7,55%	967	16,94%	2023	11,37%
Friuli Venezia Giulia	5	2,36%	123	2,15%	366	2,06%
Lazio	13	6,13%	200	3,50%	783	4,40%
Liguria	5	2,36%	81	1,42%	391	2,20%
Lombardia	25	11,79%	509	8,91%	2221	12,48%
Marche	11	5,19%	279	4,89%	646	3,63%
Molise	4	1,89%	29	0,51%	55	0,31%
Piemonte	14	6,60%	293	5,13%	1362	7,65%
Puglia	6	2,83%	195	3,42%	550	3,09%
Sardegna	6	2,83%	268	4,69%	366	2,06%
Sicilia	18	8,49%	432	7,57%	2229	12,52%
Toscana	17	8,02%	476	8,34%	1393	7,83%
Trentino Alto	3	1,42%	135	2,36%	362	2,03%

Adige						
Umbria	4	1,89%	90	1,58%	265	1,49%
Valle d'Aosta	1	0,47%	15	0,26%	60	0,34%
Veneto	17	8,02%	319	5,59%	1347	7,57%
	212		5710		17798	

*Fonte:*Ns. elaborazioni su dati del sito della Italian Linux Society (www.linux.it)

L'identità sociale riveste un ruolo cruciale anche nella definizione delle motivazioni a partecipare ai Linux User Groups nel modello dell'intentional social action sviluppato da Bagozzi & Dholakia, (2006). I LUG sono comunità di utenti del sistema operativo Linux, organizzati a livello locale e presenti sia fisicamente che virtualmente, con propri siti, forum, e mailing lists, che periodicamente si incontrano online e face-to-face per svolgere diverse attività, sia individuali che collettive, che possono essere gli Installfest (o "installation festivals", in cui, in genere durante i cd. "Linux day", chiunque può farsi installare il sistema operativo sul proprio pc) e in generale, la diffusione presso il pubblico dei sistemi operativi open source, la formazione avanzata in tema di programmazione, la diagnosi di problemi e la segnalazione dei "bugs" agli sviluppatori; il sostegno agli utenti e agli sviluppatori per la risoluzione di problemi; la diffusione, anche tra le imprese, di questo sistema operativo e, in generale, delle soluzioni open source; la raccolta di fondi a sostegno delle varie fondazioni oss; la diffusione, attraverso un positivo passaparola, dei benefici dall'uso di questo tipo di software.

Nel modello della intentional social action applicato da Bagozzi & Dholakia (2006) ai LUGs da loro analizzati, il nucleo del modello è la *we-intention* di partecipare alla comunità. Grazie alle we-intentions, i membri del gruppo credono che da soli non potranno mai realizzare

gli obiettivi del gruppo e che il gruppo deve agire in modo da raggiungere i propri obiettivi.

I fattori che influenzano le we-intentions a partecipare sono diversi, e vanno dall'atteggiamento verso la partecipazione al LUG, alla percezione del membro del LUG in merito alla facilità o meno di partecipare alla comunità, alle emozioni anticipate, cioè come l'utente del LUG anticipa le conseguenze della sua partecipazione o meno al LUG.

Altri fattori sono appunto il ruolo dell'identità sociale, che portano l'utente del LUG ad enfatizzare le similarità che lo legano agli altri membri del gruppo (ad esempio, il rifiuto del modello di creazione offerto dal software proprietario), nonché l'identificazione con il movimento e l'ideologia open source.

Il modello elaborato da Bagozzi e Dholakia (2006) presenta diverse analogie ma anche sostanziali differenze con il modello VIST proposto da Hertel et al. (2003, cfr. *supra*).

In particolare, la *valence* intesa come la valutazione da parte del singolo degli obiettivi del gruppo viene da Bagozzi e Dholakia rielaborata con l'aggiunta dell'aspettativa sulla probabilità che il gruppo raggiungerà i propri obiettivi (ponderazione che è invece presente nei *collective motives* del modello di Klandermans).

La self-efficacy del modello VIST viene riproposto dai due autori sotto forma della capacità dell'utente di intervenire nello sviluppo del software (*perceived behavioral control*).

Infine la fiducia, o *trust*, che viene da Hertel et al. (2003) concepita come l'aspettativa del singolo circa il fatto che il proprio contributo sarà contraccambiato dagli altri, mentre nel modello della

intentional social action esso rientra nel più ampio concetto dell'identità sociale, che non è basata su un calcolo utilitaristico, ma piuttosto nei processi di identificazione con la comunità, con la consapevolezza di esserne parte, con i legami affettivi rispetto agli altri membri del gruppo, nonché con la autostima legata al gruppo.

I risultati dello studio portano gli autori ad affermare che la partecipazione degli utenti ai LUG può essere spiegata da una combinazione di fattori sociali e psicologici. Le *we-intentions* sono dei fattori determinanti nel decidere la partecipazione al gruppo, e sono a loro volta influenzate dall'identità sociale. Inoltre essi colgono una differenza sostanziale tra i novizi e gli utenti esperti, per cui questi ultimi, grazie alla maggiore esperienza, alla maggiore efficacia dell'identità sociale sulle we-intentions, risultano essere più impegnati nelle attività sociali rispetto ai *gnubies*,

Questa visione si distanzia notevolmente dai risultati di Hertel, Nieder, & Herrmann, (2003), in quali individuano come una delle principali determinanti della partecipazione degli sviluppatori l'interesse di tipo pragmatico legato alla necessità di migliorare la qualità del kernel.

Un'altra classe di motivazioni sottostanti la partecipazione ai progetti open source è legata alle motivazioni estrinseche, definite da Lerner & Tirole (2002) come la somma tra il payoff immediato e quello futuro derivante dalla partecipazione ad un progetto open source. Questi payoff derivano dalla differenza tra i benefici, immediati o attesi, ed i costi sostenuti per la partecipazione.

I costi sono principalmente dei costi opportunità, derivanti dal fatto che, mentre sono impegnati nella scrittura di linee di codice per

un software open source, gli sviluppatori, se sono lavoratori autonomi, rinunciano al guadagno che avrebbero potuto ottenere scrivendo lo stesso codice per un software proprietario, mentre se lavorano alle dipendenze di un'impresa, un ente di ricerca etc, il costo è rappresentato dalla distrazione offerta da tale attività.

I benefici attesi possono essere diversi, e legati a diversi fattori, come la possibilità di ottenere profitti dalla vendita futura di servizi collegati al software open source come la formazione, il supporto, l'implementazioni di versioni customizzate, la distribuzione etc, o ancora essere legati alle opportunità di crescita professionale legate alla maggiore esperienza o all'accrescimento delle proprie capacità, che contribuiscono alla creazione di un capitale umano. A questi si aggiungono il *self marketing*, cioè la capacità del software open source, vista la trasparenza che caratterizza la sua produzione[34], nonché, per lo stesso motivo, ottenere il riconoscimento dei propri pari. (Lee & Cole, 2003)

Lerner & Tirole (2002) definiscono gli incentivi derivanti da questi benefici attesi come *career concern incentives*, che si riferiscono alla possibilità di ottenere futuri impieghi, dar vita a future open source software companies o attirare venture capital, a cui si aggiungono gli incentivi derivanti dai benefici legati alla gratificazione del proprio ego, definiti come *ego gratification incentives*, derivanti dal riconoscimento dei propri pari. Insieme, i due tipi di incentivi danno luogo ai cd. *signaling incentives*, che è maggiore quando:

[34] Grazie all'accesso al Concurrent Versions System – che consente di visualizzare tutte le modifiche apportate al software – e al *credit file*, è possibile conoscere in qualsiasi momento chi ha contribuito a fare che cosa.

- La performance è più visibile per l'audience rilevante, e ciò determina delle complementarità strategiche, in quanto i programmatori vorranno lavorare ai progetti che hanno il maggior numero di sviluppatori;
- E' maggiore l'impatto dello sforzo sulla performance;
- E' elevato il contenuto informativo della performance circa il talento sottostante del programmatore.

Rispetto ad un software proprietario, la partecipazione ad un software open source determina un maggiore *signaling incentive*, in quanto esso consente:

1. Una migliore misurazione della performance, grazie alla maggiore trasparenza del processo di creazione del software open source, di cui è possibile in ogni istante seguire l'evoluzione dalla semplice visione del *concurrent versioning system* ospitato sul sito del progetto, nonché dal *credit file* che accoglie il nome dei contributori;
2. Piena iniziativa del programmatore, che agisce da principale di se stesso;
3. Maggiore fluidità nel mondo del lavoro per chi opera nel mondo open source: i programmatori possiedono un capitale umano meno idiosincratico, meno firm-specific che consente loro di cambiare più rapidamente lavoro.

Buona parte degli studi sulle motivazioni degli sviluppatori, quindi, dicotomizza le motivazioni degli sviluppatori distinguendole in intrinseche ed estrinseche, in quanto il modello di riferimento è quello di Ryan & Deci (2000) fondato sulla self-determination, modello che presenta alcuni limiti (Freeman, 2007):

- Esso assume che la motivazione sia misurabile, mentre in realtà la motivazione è un fenomeno complesso, che comporta aspetti qualitativi;
- Le motivazioni sono concepite come innate, ed i bisogni sono ordinati gerarchicamente, per cui il soddisfacimento di quelli collocati negli strati superiori di tale gerarchia dipende dalla soddisfazione di quelli collocati ai livelli immediatamente inferiori.
- Il modello di Ryan e Deci è stato testato in contesti sperimentali, in cui era possibile tenere sotto controllo tutte (o quasi) le variabili analizzate: la sua trasferibilità nel contesto delle comunità di innovazione sottostanti i progetti open source è tutta da valutare;
- La distinzione tra motivazioni intrinseche ed estrinseche è artificiale, in quanto implica che lo sviluppatore che partecipa ad un progetto open source per divertimento, come homo ludens, non possa al contempo gradire l'ottenimento di un beneficio economico per questo impegno.

Per questo motivo, alcuni autori hanno usato un approccio più olistico allo studio delle motivazioni dei partecipanti ai progetti open source. Uno di questi è stata Sonali Shah (2006), che attraverso un'analisi di tipo qualitativo ha individuato due diverse tipologie di partecipanti: la prima è quella dei partecipanti *need-driven*, la cui motivazione era essenzialmente legata alla necessità di ottenere un software utile nel proprio contesto lavorativo, che contavano sulla reciprocità da parte degli altri membri della comunità, dalla necessità di apportare modifiche ad un programma e da aseptti legati alla

propria carriera. La seconda categoria è invece quella degli hobbisti, la cui motivazione è legata al desiderio di divertirsi e di ottenere dei feed-back dagli altri membri della comunità. In sostanza, anche se con termini differenti, Shah riproduce una dicotomia simile a quella tra motivazioni intrinseche-motivazioni estrinseche offerte dal resto della letteratura.

4.5 L'integrazione degli utenti: comunità di pratica?

Un altro elemento comune ai diversi modelli presentati riguarda l'esistenza di diversi livelli di partecipazione e, quindi, diverse tipologie di utenti: ad esempio, Hertel et al. (2003) distinguono tra sviluppatori e "lettori interessati" nell'analizzare il caso di Linux, mentre Bagozzi e Dholoakia (2006) rilevano diverse motivazioni sottostanti la partecipazione dei gnubies e degli utenti di maggiore esperienza, mentre Shah (2003) mette in evidenza come i "novizi" ricerchino le comunità open source affinchè possano esser loro d'aiuto nella risoluzione di un loro problema ma, con l'accumulazione dell'esperienza, le loro motivazioni cambino e diventino il divertimento nel contribuire all'architettura del software.

Capire se e come gli utenti vengano progressivamente integrati nello sviluppo vero e proprio del software assume una notevole rilevanza anche perché, al contrario di quanto si crede comunemente, dietro i software open source spesso e volentieri non c'è una comunità, ma piuttosto un solo sviluppatore. Questa situazione è sintomatica di un fallimento del progetto open source.

Krishnamurthy (2002), in uno dei più ampi studi empirici finora condotti, rileva infatti che in 100 progetti open source "maturi", circa il 51% di essi aveva un solo project administrator, mentre solo il 29%

aveva più di cinque programmatori attivi, il 19% più di dieci mentre il 22% aveva un solo sviluppatore che contribuiva al software.

E' anche vero che lo studio di Krishnamurthy concentra l'attenzione sui prodotti maturi, anche se l'analisi condotta sui 20 principali (in termini di numero di utenti) progetti, collocati in diverse fasi del loro ciclo di vita (progettazione, pre-alfa, alfa, beta, produzione/stabile, maturo) evidenzia comunque un esiguo numero di sviluppatori rispetto all'ampiezza delle rispettive comunità (cfr. tab. 4.3).

Tabella 4.3: Programmatori nei top 20 progetti open source

Variabile	1	2	3	4	5	6	7	8
Valence	-							
Instrumentality	0,19	-						
Self-efficacy	0,07	0,46	-					
Trust	0,21	0,03	-0,03	-				
Ore/settimane dedicate	0,19	0,34	0,21	0,19	-			
Aumento partecipazione	0,40	0,34	0,09	0,06	-0,02	-		
Numero di patches	0,05	0,35	0,36	0,06	0,32	0,09	-	
Linee codice (scala log.)	-0,13	0,08	0,32	-0,06	0,05	-0,33	0,31	-
Media	4,7	3,4	4,0	3,8	20,5	2,4	2,4	2,6
Dev.st.	0,5	1,0	0,6	0,8	19,4	1,0	1,6	0,9

Fonte: Krishnamurthy, 2002

Esistono alcune condizioni che i progetti open source, affinchè siano di successo, devono assolvere, condizioni che cambiano proprio a seconda del ciclo di vita che attraversano.

Secondo Schweik & Semenov (2003) esisterebbero tre fasi del ciclo di vita dei progetti open source:

1. Fase 1 – iniziazione del progetto: in questa fase si ha l'ideazione del progetto ad opera di una o più persone essenzialmente per motivi riconducibili alla risoluzione di un problema concreto, al desiderio di intraprendere un progetto collocato alla frontiera della programmazione, uniti all'assenza di costi opportunità legati all'intrapresa del progetto e all'opportunità di accrescere le proprie capacità. In questa fase è poi fondamentale l'esistenza

di un nocciolo del programma (come era stato il kernel per Linux) che deve esistere già prima che il progetto diventi pubblico. Fondamentale poi è l'architettura del software, che deve essere fondata sulla modularità, ovvero sull'articolazione del software in moduli, a se stanti, che consentano ai potenziali partecipanti di poter lavorare in parallelo; collegata alla struttura modulare è l'esistenza di un buon disegno complessivo del programma.

2. Fase 2 – il programma diventa aperto: in questa fase gli iniziatori del progetto rendono il software aperto, scelgono la licenza open source che più si adatta alla sua creazione e in più devono assumere scelte fondamentali al fine di assicurare (1) la credibilità del progetto; (2) adeguati sistemi di comunicazione; (3) idonei sistemi di controllo delle versioni (CVS); (4) efficaci strategie di reclutamento; (5) appropriate strutture di governance del progetto e disegno istituzionale.
3. Fase – Crescita e declino del progetto: in questa fase, in misura strettamente correlata alla efficacia con cui sono state definite le scelte cruciali della fase 2, il progetto potrà veder il coinvolgimento di un crescente numero di sviluppatori, oppure potrà assistere alla riduzione dell'interesse verso il progetto e al suo conseguente declino.

Le scelte in termini di architettura del software, della sua modularità e dell'individuazione di adeguati sistemi sono fondamentali per l'organizzazione del lavoro all'interno della comunità, nonché per l'integrazione dei nuovi utenti nel progetto.

Un'analisi interessante è stata quella realizzata da Von Krogh, Spaeth, & Lakhani (2003), in riferimento alla comunità sorta attorno al progetto per la realizzazione del software peer-to-peer Freenet, in riferimento alla quale essi hanno focalizzato l'attenzione sui processi di integrazione dei nuovi utenti e sulla possibilità che essi acquisiscano una specializzazione nell'attività innovativa in cui sono coinvolti.

In questo senso il modello di integrazione offerto da Von Krogh et al., comporta l'acquisizione della conoscenza attraverso l'assunzione, da parte del newbie, del ruolo di "partecipante periferico legittimo", in una vera e propria comunità di pratica, che acquisisce gradualmente conoscenza e reputazione attraverso un processo di interazione sociale. Lave e Wenger (1991, cit in Tuomi, 2000) sostengono che l'apprendimento è fondamentale per diventare un membro accettato dalla comunità, in cui la competenza, l'identità e il senso di appartenenza al gruppo sono elementi inscindibili.

Nel loro caso empirico, Von Krogh et al. (2003) individuano quattro diversi livelli di partecipazione al progetto:

- Partecipanti alla mailing list: sono iscritti alla mailing list ma non hanno mai contribuito alla scrittura del codice;
- Joiner: contribuiscono al codice ma non hanno accesso al repository del CVS;
- Newcomer: hanno iniziato ad apportare modifiche al CVS;
- Sviluppatore: contribuisce attivamente alla scrittura del codice del programma.

Per quanto riguarda l'architettura del software, essi rilevano la presenza di 16 moduli, articolati in 56 "features", che consentono

l'attribuzione di tasks differenziati e specializzati ai diversi programmatori. Un altro aspetto rilevante per il successo del progetto è la presenza di una notevole "massa critica" di potenziali partecipanti, derivanti dall'alto numero di downloads.

Von Hippel & Von Krogh (2003) rilevano poi un elevato turnover dei partecipanti, associato ad un ridotto numero di partecipanti al progetto a cui, effettivamente, è consentita la modifica del software in qualità di vero e proprio sviluppatore.

La loro analisi del processo di integrazione dei newcomer negli sviluppatori parte dall'esame del tipo e della quantità di lavoro offerto, attraverso lo studio delle e-mail inviate alla lista di discussione del progetto. La quantità (in termini di numero di e-mail inviate) e il tipo di contributo della cd. "joining script" vengono rilevati per individuare quali caratteristiche ha il newbie che viene ammesso nel ridotto gruppo degli sviluppatori.

Per quanto riguarda l'aspetto quantitativo, Von Hippel & Von Krogh rilevano come la quantità di e-mail inviate dal newcomer prima di diventare developer possa essere quantificato, nel caso di Freenet, in almeno di 23,4, con una devianza standard di 37,8 a causa della presenza di un caso in cui un nuovo partecipante ha inviato circa 194 e-mail prima di essere ammesso al repository del programma.

Per quanto riguarda il livello di attività, cruciale è, secondo gli Autori, la prima e-mail inviata: nel 10% dei casi essa contiene una descrizione delle proprie esperienze e capacità tecniche, mentre nel 40% contiene un contributo ad una discussione tecnica già in atto. In questo caso, il newcomer ha la possibilità concreta di iniziare il proprio processo di integrazione nella comunità, attraverso

l'apprendimento dell'architettura – non semplice – di Freenet, e di segnalare le proprie capacità agli sviluppatori. E' interessante evidenziare che non è sufficiente mostrare il proprio interesse per poi essere parte della comunità: solo il 12,3% di coloro che hanno partecipato a discussioni tecniche senza offrire un contributo in termini di codice è stato ammesso al progetto. Nel 16,7% di coloro che sono poi diventati joiners si è avuto un contributo in termini di risoluzione di un bug o di un software che è andato ad arricchire l'architettura del software.

Cruciale è stato per gli Autori seguire l'evoluzione di coloro che dichiarato il proprio interesse a diventare sviluppatori: solo per il 16,7% dei casi, tale interesse si è tradotto in una attività concreta; nel 36,7% dei casi in cui il newcomer ha richiesto agli sviluppatori di essere ammessi come programmatori, essi non hanno avuto risposta, anche se nel 56,7% dei casi gli sviluppatori del progetto hanno invitato i membri della comunità a trovare la parte del programma che fosse ad essi più confacente, in termini di capacità ed esperienza necessarie, per lavorarci su. Solo nel 16,7% dei casi sono stati assegnati compiti specifici.

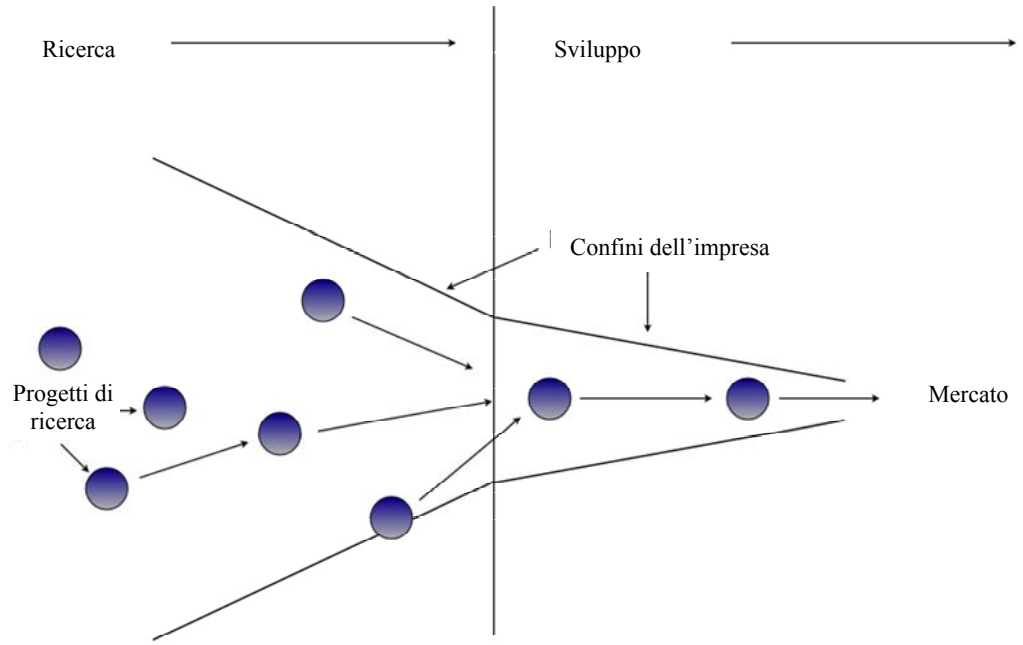
Capitolo 5

FLOSS e mercato: business models per un'open innovation

5.1 L'appropriabilità dell'innovazione e l'open innovation

Come rilevato da Chesbrough (2003) in diversi settori, soprattutto quelli hi-tech, negli ultimi anni si è assistito ad una situazione paradossale, per cui, nonostante la ricchezza di idee e capitale esistenti all'esterno, diverse grandi imprese non riescono a gestire più l'innovazione seguendo quello che è stato ed è tutt'ora il paradigma dominante della *innovazione chiusa*.

Fig. 5.1: Il modello dell'innovazione chiusa



Fonte: Ns adattamento su Chesbrough (2003)

In base a questo paradigma, l'innovazione richiede il controllo – e la proprietà – delle fonti di innovazione: le imprese devono generare le proprie idee grazie al proprio dipartimento di ricerca e sviluppo, e quindi svilupparle, finanziarle, ricavarne un'innovazione di prodotto o di processo e quindi procedere al loro sfruttamento commerciale.

Questo paradigma implica una profonda integrazione verticale tale da sfociare in manifestazioni patologiche come la "Not Invented Here Syndrome" (NIHS) (Katz & Allen, 1982), a causa della quale l'impresa rifiuta di ricorrere a fonti esterne di innovazione per via della difficoltà nel testarne l'affidabilità proprio perché non proveniente dal suo interno.

In diversi settori il paradigma dell'innovazione chiusa si è rivelato obsoleto, a causa di diversi fattori che ne hanno determinato l'erosione:

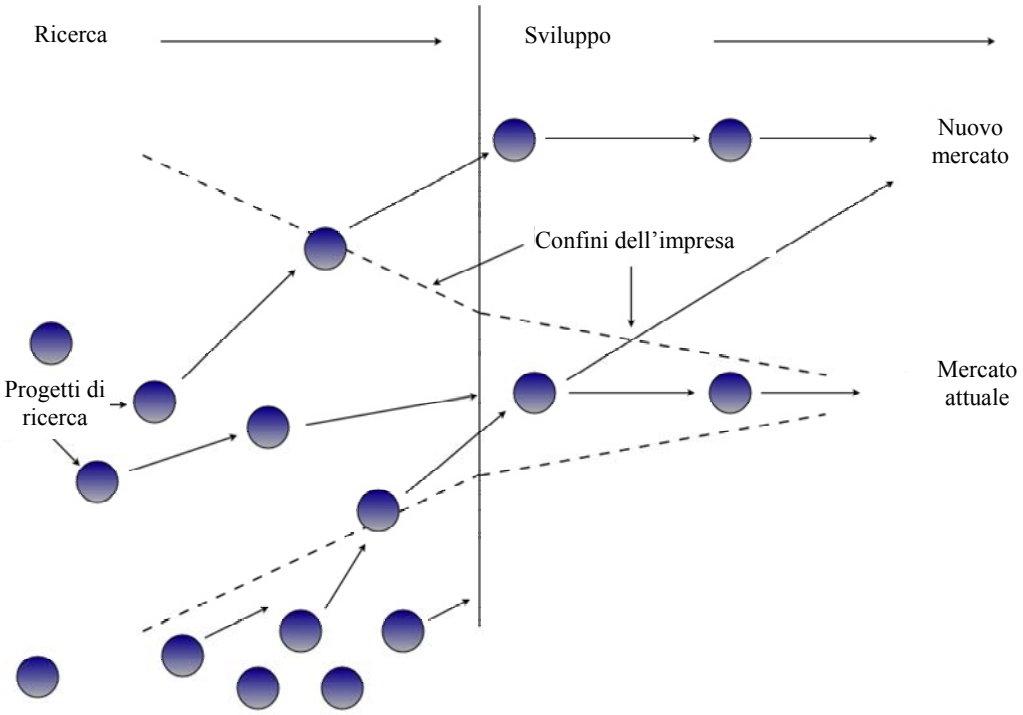
1. *La crescente disponibilità di lavoratori qualificati*, che a causa della loro crescente mobilità, trasferiscono la conoscenza acquisita nelle imprese in cui magari domina il modello della closed innovation verso i propri nuovi datori di lavoro, clienti, verso le università etc. Emblematico, in tal senso è l'esempio dato da un ex ingegnere della IBM, Al Shugart, che nel 1969 lasciò la Big Blue, fino a quel momento leader incontrastata nel settore dei dischi rigidi per pc, per passare alla Memorex; successivamente, nel 1973 fondò la Shugart Associates, creando un nuovo disco da 8" e, infine, creò la Seagate, che produceva dischi da 5" ¼. Ad ogni passaggio Shugart portò via anche altre persone dalle imprese che lasciava e così fu per altri dipendenti IBM. Come conseguenza, nel settore dei disk-drivers per pc circa il 23% delle 99 imprese operanti nel mercato aveva ex dipendenti IBM tra i propri fondatori.
2. *La crescita del mercato del venture capital*, interessato da una crescita esponenziale dal 1980 fino ad oggi, con un picco raggiunto nel 2000, quando, negli Stati Uniti, si stima un investimento in vc di oltre 80 miliardi di dollari. Il venture capital rende disponibile il capitale necessario per lo sviluppo di nuove iniziative imprenditoriali caratterizzate da un elevato tasso di rischio ma, al contempo, da un potenziale rilevante ritorno economico, molto attraente soprattutto per i dipendenti delle stesse imprese che hanno investito ingenti risorse per la creazione di conoscenza.
3. *Le crescenti capacità dei fornitori esterni*, dovute anche ai due processi sopra descritti, che rende possibile l'esistenza di fonti

esterne di innovazione di qualità uguale o superiore rispetto a quella acquisibile dall'impresa al suo interno, a causa anche della crescente importanza delle cosiddette tecnologie complesse (Kingston, 2001) e a quelle multipurpose, che rende impensabile il ricorso alle sole tecnologie interne. Questo fattore rappresenta un'arma a doppio taglio, perché, soprattutto nel caso delle general purpose technologies (GPT), la divisione del lavoro nell'attività di ricerca e sviluppo rende possibile anche alle imprese concorrenti accedere a fonti di innovazione esterne attraverso i *markets for technology* (Arora, Fosfuri, & Gambardella, 2001).

In un simile contesto, il modello della closed innovation presenta dei limiti crescenti, superabili solo grazie al passaggio ad un nuovo paradigma, che Chesbrough (2003) definisce come *open innovation*. Grazie a questo paradigma, in un contesto economico caratterizzato dalla disponibilità di enormi quantità di conoscenza, un'impresa deve organizzare la propria attività di ricerca e sviluppo al fine di:

- Identificare, comprendere, selezionare e connettersi al pool di conoscenza disponibile al proprio esterno;
- Occuparsi di creare internamente solo quella conoscenza non disponibile al suo esterno;
- Integrare la conoscenza interna ed esterna al fine di creare combinazioni tecnologiche più complesse, nuovi sistemi ed architetture;
- Generare profitti addizionali vendendo i prodotti della ricerca alle altre imprese.

Fig. 5.2: il modello della open innovation



Fonte: Ns adattamento su Chesbrough (2003)

L'open innovation può essere definita come quel paradigma in base al quale l'impresa può usare sia idee esterne che idee interne per lo sviluppo dell'innovazione, determinato da alcuni fattori chiave, come la globalizzazione, per cui l'estensione dei mercati geografici rende possibile una maggiore specializzazione e divisione del lavoro; il miglioramento di alcune istituzioni del mercato, come i diritti sulla proprietà intellettuale; il venture capital e gli standards tecnologici che fanno sì che le risorse trascendano più facilmente i confini delle imprese; il cambiamento tecnologico che riduce la scala minima di efficienza, determinando l'incremento della specializzazione; la crescente mobilità del mercato del lavoro, soprattutto per il personale specializzato.

In aggiunta a questo il cambiamento tecnologico, come quello indotto dalla rivoluzione ICT, ha influenzato il modo con cui le imprese comunicano, collaborano e si coordinano.

E' quindi evidente che, soprattutto il settore del software, sia stato interessato enormemente dallo sviluppo delle ICT, perché la duplicazione del prodotto può essere realizzata elettronicamente a costi marginali quasi nulli; il trasporto del prodotto può essere ottenuto attraverso la trasmissione elettronica ad un costo marginale prossimo a zero; il software viene installato su una base hardware generica, che genera ridotti problemi di compatibilità (Dahlander & Wallin, 2006).

Come conseguenza di ciò, è possibile ideare nuove forme di organizzazione del lavoro innovativo, che rendono possibile una notevole specializzazione. La specializzazione, ovviamente, richiede poi l'ideazione di forme di integrazione dei compiti e delle responsabilità.

L'incentivo ad attingere da fonti tecnologiche esterne è particolarmente rilevante quando il settore è caratterizzato da elevati livelli di opportunità tecnologiche ed esistono significativi investimenti nella ricerca da parte delle altre imprese, in un contesto di ridotta appropriabilità delle rendite da innovazione che, se da un lato, riduce l'incentivo ad investire in ricerca e sviluppo, dall'altro lato riduce il costo della ricerca (Klevorick, Levin, Nelson, & Winter, 1995).

In un regime caratterizzato da ridotta appropriabilità, gli assets complementari, ovvero il complesso dei servizi necessari per la traduzione del know-how in un innovazione commerciale, diventano

cruciali per l'innovazione sia nella fase paradigmatica che matura del settore e si distinguono in (Teece, 1986):

- Assets generici (*generic assets*), che sono tecnologie o altri assets *general purpose*, che possono essere utilizzati da qualsiasi impresa senza che debbano essere ritagliati sull'innovazione in questione;
- Assets specializzati (*specialized assets*) sono quelli caratterizzati da una dipendenza unilaterale tra l'innovazione e l'asset complementare;
- Assets co-specializzati (*co-specialized assets*) sono quelli per i quali esiste una dipendenza bilaterale tra l'innovazione e l'asset.

Gli assets complementari sono assets presenti nell'impresa ed integrati nella sua catena del valore. Nel caso dello sviluppo del Floss la catena del valore è disintegrata, in quanto gli individui possono assumere il ruolo di fornitori di codice del software ma anche di pionieri nell'adozione del prodotto e la comunità assume il ruolo di asset complementare (Dahlander & Wallin, 2006).

Fig. 5.3: Contratti, strategie di integrazione e risultati per l'innovatore negli assets specializzati

	Forte appropriabilità legale/tecnica	Ridotta appropriabilità legale/tecnica	
		Innovatore ben posizionato vs imitatori rispetto agli assets complementari	**Innovatore mal posizionato vs imitatori rispetto agli assets complementari**
Innovatori ed imitatori in posizione di vantaggio rispetto a fornitori assets complementari	Contratto Innovatore Vince	Contratto Innovatore dovrebbe vincere	Contratto Inno-vatore o imitatore vincono e forni tori senza vantaggio
Innovatori ed	Contratto se con-viene, integrare		Contratto per limitare es-

Grado di protezione della proprietà intellettuale

Potere di mercato di innovatori/imitatori Vs. proprietari assets complementari

imitatori in posizione di svantaggio rispetto a fornitori assets complementari	se necessario Inno-vatore può vincere e divide profitti con fornitore assets	Integrare Innovatore dovrebbe vincere	posizione Innova-tore perde vs imitatori e fornitori assets c.

Fonte: Ns. traduzione da Teece (1986)

Secondo la visione di Teece (1986), in un regime di ridotta appropriabilità dell'innovazione, è per l'impresa cruciale l'assunzione del controllo degli assets complementari attraverso il ricorso a due forme di coordinamento tra loro antitetiche, ovvero il contratto o l'integrazione verticale (fig 5.3)

Le comunità di sviluppatori del software free ed open source possono essere viste come un asset complementare, in quanto il lavoro da esse sviluppato può essere, a determinate condizioni, incorporato dalle imprese in un proprio business model.

A causa delle norme e dei meccanismi, sulla carta molto forti, ideati dalle comunità floss per impedire l'open source hijacking (Ciffolilli, 2004), non è pensabile ricorrere ai markets for technologies per acquisire il controllo su queste risorse nonostante il carattere di general purpose technology del software (Arora, Fosfuri, & Gambardella, 2001), anche perchè l'impresa non ha un interlocutore avente natura di persona fisica o giuridica con cui interagire[35] (O'Mahony, 2002)

[35] O'Mahony (2002) riporta le perplessità di un manager di una software company inserita tra le top 500 aziende da Fortune in merito ai problemi di interazione con le comunità open source: *"We all set around and Jim goes 'how am I going to do a deal with a webpage?' And I said well we will figure something out."*

Per questo motivo le comunità open source non possono essere acquisite dalle imprese come assets complementari nelle forme classiche individuate da Teece (1986) ma è necessario che queste ultime acquisiscano legittimazione presso le comunità attraverso la partecipazione ai progetti, l'interazione e l'apprendimento (Dahlander & Wallin, 2006). Data la diversità in termini di obiettivi e di caratteristiche che caratterizzano le comunità open source e le software companies, può essere interessante capire quali possono essere le motivazioni sottostanti tale collaborazione.

5.2 Motivazioni dell'open innovation tra comunità floss ed imprese

Sia le comunità open source che le imprese possono trarre benefici dalla collaborazione ad un comune progetto open source (O'Mahony, 2002).

Innanzitutto, per le comunità floss possono acquisire un beneficio in termini di maggiore visibilità del progetto. Infatti in questo modo il loro lavoro può raggiungere un'audience più ampia, che è, anche nella prospettiva della sopravvivenza del progetto nel lungo periodo, un fattore fondamentale, in quanto maggiore popolarità indica una potenziale maggiore massa critica di futuri sviluppatori che potranno essere interessati a partecipare al progetto.

La maggiore visibilità deriva anche dal fatto che in genere le software companies hanno come target market un mercato di massa, a differenza dei progetti open source che, invece, hanno in genere un audience molto tecnica, composta dagli stessi sviluppatori. Talvolta ciò implica una maggiore difficoltà d'uso dei software open source da parte di chi è sprovvisto di adeguato bagaglio tecnico, in quanto il

codice scritto dagli sviluppatori è prevalentemente destinato a se stessi. La partecipazione delle imprese può, in questo senso, offrire un incentivo, anche economico, agli sviluppatori a realizzare un prodotto più user-friendly, dato che non essi non sono intrinsecamente motivati a farlo. Oltre che offrire veri e propri incentivi economici agli sviluppatori a lavorare in aree della programmazione ad essi non gradite, la collaborazione con le imprese comporta la possibilità di accedere all'assistenza nella creazione di interfacce disegnate per i mercati di consumo.

La collaborazione con le imprese può inoltre rappresentare uno stimolo intellettuale derivante dall'offerta di progetti ricadenti nella frontiera delle scienze informatiche, che dà agli sviluppatori l'opportunità di mettere alla prova le proprie capacità nonché di acquisirne di nuove.

Infine, il terzo beneficio, probabilmente il più importante, è l'offerta di sostegno finanziario, utile al progetto open source soprattutto al raggiungimento di una determinata soglia dimensionale, che comporta una maggiore complessità del progetto. L'impresa può offrire strutture come server, hosting e altri servizi utili alla comunità per sostenere lo sviluppo del progetto su una più ampia scala.

Per quanto, invece, riguarda le imprese, una prima motivazione è legata ai bassi costi opportunità dei knowledge spillover a cui esse vanno inevitabilmente incontro, date le clausole delle licenze d'uso open source. I costi derivanti dalla disclosure del proprio codice sorgente deriva dal fatto che i punti di forza e di debolezza delle comunità open source da un lato e delle software companies dall'altra sono complementari: i prodotti realizzati dalle comunità open source

non sono destinati ad un mercato di consumo, e quindi non si porranno in competizione con i prodotti realizzati dalle imprese.

In realtà il problema della competizione si pone in modo serio se il business model della software company dipende in modo esclusivo dalla proprietà del software e dalla cessione a titolo oneroso delle relative licenze d'uso: in questo caso la possibilità che essa collabori con una comunità open source è molto remota, e infatti non è un caso che le imprese più coinvolte nei progetti open source siano quelle aventi un business model fondato sui componenti hardware (come IBM) o sulla vendita di servizi complementari. In quest'ultimo caso, la convenienza per le imprese nel collaborare è dovuta alla possibilità di utilizzare sulle proprie macchine o di offrire un servizio complementare ad un software per il quale non paga alcuna licenza d'uso.

Un'altra motivazione è dovuta al fatto che, senza alcuna preventiva autorizzazione, spesso i dipendenti di numerose imprese informatiche lavorano, durante il tempo libero, a progetti open source, e quindi il fatto che esistano progetti tali da interessare i propri dipendenti anche al di fuori del tempo da essi dedicato al lavoro e che di fatto diversi software open source vengano da questi utilizzati nello svolgere il proprio lavoro nell'impresa, rende il tutto molto interessante agli occhi del management.

Risulta poi essere determinante anche l'interesse del pubblico (utenti finali e sviluppatori) verso il progetto open source, che può essere determinante nel rivitalizzare progetti maturi, il cui costo di manutenzione da parte dell'impresa sarebbe eccessivo rispetto al potenziale ritorno economico. E'stato questo, per esempio, il caso di

Mozilla, originariamente il browser Navigator della Netscape, che rilasciò il codice sorgente del programma nel 1998 sotto la neonata licenza Mozilla Public Licence, e che poi venne rivitalizzato dalla comunità sorta attorno ad esso, diventando una vera e propria suite chiamata SeaMonkey, composta dal browser Mozilla Firefox e dal client di posta Mozilla Thunderbird. Il successo della rivitalizzazione di questo progetto è stato decisamente rilevante: al 26 ottobre 2008, Mozilla Firefox aveva oltre il 20% di share nel mercato dei browser (Net Applications, 2008).

Un'altra determinante molto forte della partecipazione delle imprese ai progetti open source è la volontà di rompere un monopolio di una software company in un particolare settore, e questo per esempio il caso di Microsoft e di Windows.

Ancora, la partecipazione delle software companies può essere spiegata dalla possibilità di accedere rapidamente e in itinere (anziché dover aspettare il termine del progetto) al processo di creazione del codice sorgente che è pubblico e in genere molto rapido, ed è sostenuto da sviluppatori dotati di notevoli capacità .

Infine, decisamente rilevante è la possibilità di avere rapporti con gli sviluppatori coinvolti nella comunità open source in alcuni casi leader nel proprio campo anche a livello internazionale, la cui fama, è, a sua volta, un forte elemento di attrazione per altri sviluppatori di talento, magari in un'area di sviluppo non contigua con quella dell'impresa. In questo senso, quindi, la perdita del controllo sul software è compensata dalla possibilità di reclutare sviluppatori di talento, di accedere ad un grande mercato preesistente, con una notevole base di utenti, a cui si aggiunge, dato il particolare modus

operandi seguito nei progetti FLOSS, di acquisire un vantaggio temporale nello sviluppo del prodotto.

5.3 Forme di partecipazione delle imprese

Come anticipato precedentemente, le motivazioni sottostanti la partecipazione delle imprese ai progetti open source sono prevalentemente economiche (cfr. tab. 5.1) (Bonaccorsi, Lorenzi, Merito, & Rossi, 2007), a differenza delle motivazioni prevalentemente non (immediatamente) economiche degli sviluppatori (cfr. par. 4.4).

Come visibile nella tabella sotto riportata, le motivazioni sottostanti la partecipazione degli sviluppatori e delle imprese ai progetti open source sono diverse e ciò origina dei possibili problemi di interazione. Secondo Dahlander & Magnusson (2005) è possibile identificare tre possibili diversi approcci usati dalle imprese per interrelarsi con le comunità open source:

1. *approccio simbiotico*
2. *approccio commensalistico*
3. *approccio parassitico*

Tab. 5.1: Le motivazioni – Confronto fra imprese e sviluppatori

Tipologia motivazione	Imprese	Sviluppatori
Economica	Indipendenza da politiche di licenza e di prezzo delle grandi software firms Evoluzione software come prodotto consumer-driven Profitti da servizi complementari Sfruttamento innovazione delle comunità open source Assumere buoni specialisti IT	Compensi economici Bassi costi opportunità Acquisire una reputazione presso i propri pari Acquisire futuri benefici nella carriera
Sociale	Adeguamento ai valori open source Condivisione codice e conoscenze con la comunità open source Software non come bene proprietario	Divertimento nella programmazione Altruismo Senso di appartenenza alla comunità Lotta contro il software proprietario
Tecnologica	Riduzione costi di sviluppo e miglioramento del software	Apprendimento Contributi e feedback dalla comunità

	attraverso sfruttamento contributi e feed back di sviluppatori e users Riduzione costi dell'hardware Promozione della standardizzazione	Lavorare con le ultime tecnologie Soddisfare un proprio bisogno

*Fonte:*Ns. adattamento da Bonaccorsi & Rossi (2003)

L'approccio parassitico si ha quando l'impresa persegue unicamente i propri obiettivi, senza tener conto di proprie azioni che potrebbero danneggiare la comunità. Tale approccio potrebbe essere la conseguenza di una involuzione dell'approccio commensali stico, nel momento in cui l'impresa comincia ad essere percepita come portatrice di un'influenza negativa sulla comunità, sia perché magari viola le norme basilari su cui questa si fonda, sia perché viene percepita come un free rider. Questo tipo di approccio non viene perseguito volontariamente dalle imprese, anche perché genererebbe un business model non sostenibile nel tempo. Però, date le notevoli differenze tra le ragioni della partecipazione ai progetti open source rispettivamente delle imprese e degli sviluppatori, il confine tra un atteggiamento parassitico e uno commensalistico è abbastanza labile.

L'approccio simbiotico comporta il tentativo da parte dell'impresa di co-sviluppare se stessa e la comunità, tenendo conto al contempo, al momento dell'assunzione delle decisioni strategiche, degli effetti sulla comunità. Affinché ciò sia possibile, è necessario che l'impresa sia direttamente coinvolta nella comunità, dato che la legittimazione all'interno di quest'ultima può essere acquisita solo grazie allo status riconosciuto dagli altri membri della comunità, in base ai suoi valori e alle sue norme.

Un modo per acquisire questo ruolo è, secondo Dahlander & Wallin (2006), la *sponsorship*, consistente nella scelta delle software companies di affidare ad alcuni dipendenti il compito di partecipare

alle comunità FLOSS, scelta che è stata fatta da diverse imprese nel caso di studio da essi analizzato, relativo al progetto GNOME, un desktop environment per Linux. Analizzando il flusso di e-mail degli iscritti alla mailing list dello sviluppo del software, essi hanno analizzato le relazioni che legavano i partecipanti alla comunità in qualità di dipendenti di imprese operanti nel settore al resto della comunità, ottenendo interessanti risultati. Innanzitutto hanno rilevato, conformemente alla letteratura esistente (Krishnamurthy, 2002) come pochi individui, corrispondenti al gruppo degli esperti, sia responsabile della maggioranza delle e-mail. Un altro elemento interessante riguarda i partecipanti sponsorizzati dalle imprese, che hanno risorse tali da consentir loro di acquisire posizioni favorevoli all'interno del network, Essi, infatti, tendono a ricercare in misura maggiore rispetto a chi collabora per hobby, l'interazione con gli altri membri della comunità, anche se gli altri membri della comunità tendono a contattare i partecipanti sponsorizzati in misura minore rispetto a quanto facciano loro. Ciò è probabilmente dovuto, da un lato, all'obiettivo dei partecipanti sponsorizzati di acquisire una legittimazione all'interno del gruppo, mentre, viceversa, gli altri membri della comunità tendono ad essere abbastanza sospettosi rispetto ai loro scopi aziendali. Nonostante questo, non esiste alcuna prova che i partecipanti sponsorizzati riescano a dirigere o stimolare il dibattito all'interno della comunità FLOSS, anche se esistono delle distinzioni, a seconda della tipologia di impresa che finanzia il partecipante. Infatti, nel caso di GNOME, Dahlander & Wallin (2006) hanno potuto verificare come questo problema riguardi soprattutto i dipendenti delle grandi imprese operanti nel settore del software,

mentre decisamente diverso è il caso dei dipendenti delle imprese aventi un business model fortemente basato sul software FLOSS, che risultano essere più attivi ed importanti per la comunità.

Anche nell'approccio simbiotico l'impresa non ambisce al controllo della comunità, ma comunque riesce ad acquisire una minima capacità di influenzare la direzione del progetto. Coerentemente con la gift economy che regola i rapporti all'interno della comunità, anche l'impresa deve restituire i benefici ricevuti, e un modo immediato è senza dubbio il rilascio, con le condizioni delle licenze open source, di parte del proprio codice sorgente alla comunità. Un altro modo per ricambiare quanto ricevuto dagli sviluppatori FLOSS è per l'impresa l'offerta di un'infrastruttura tecnologica che faciliti le performance dei diversi compiti di sviluppo e permetta una interazione intellettualmente stimolante.

Un modo intermedio per interagire con le comunità open source è rappresentato dall'approccio commensalistico, che consiste nella scelta dell'impresa di beneficiare dell'apporto della comunità senza arrecarle alcun danno. Sostanzialmente, l'idea sottostante questo approccio è cercare di sfruttare le risorse comunitarie che vengono continuamente rigenerate mantenendo minimo il coinvolgimento dell'impresa.

5.4 Problemi manageriali

Nel relazionarsi con le comunità open source, le software companies devono affrontare alcuni possibili problemi (Dahlander & Magnusson, 2005).

Innanzitutto le norme e i valori condivisi dai membri della comunità ed utilizzati a tutela del lavoro da questi realizzato, che

devono essere rispettati affinchè l'influenza dell'impresa non venga percepita come deleteria per il progetto. In questo senso può essere utile per l'impresa ricorrere a forme di interazione face-to-face con gli altri sviluppatori del progetto, attraverso, ad esempio, l'organizzazione di workshops ed eventi, che hanno come effetto l'adattamento proattivo delle norme e dei valori della comunità rispetto alle esigenze della software company.

Una scelta strategica cruciale è poi quella relativa alla licenza open source da utilizzare nel progetto, in quanto influenzerà il regime di appropriabilità da parte dell'impresa del codice creato, soprattutto rispetto ai moduli già creati dalla comunità, la cui licenza d'uso potrebbe non essere compatibile con il software proprietario creato dall'impresa. La scelta della licenza verso tipologie GPL e derivate ha però un effetto simbolico rilevante, in quanto comunica il sostanziale allineamento dell'impresa rispetto ai valori della comunità. La scelta della licenza deve essere assunta con l'obiettivo di risolvere il fondamentale problema del controllo e della proprietà del software open source. Le soluzioni adottate dalle imprese sono state diverse: alcune scelgono la *dual license*, attraverso l'adozione combinata di una licenza GPL e di una specifica dell'impresa pe r distinguere la parte di codice che può essere liberamente usata da quella per la quale occorre pagare una licenza d'uso; altre, invece, ricorrono alla licenza GPL per l'intero prodotto, salvo poi vendere, come vero e proprio software proprietario, applicazioni ed add-ons.

Un altro problema fondamentale delle imprese è quello di mantenere alto l'interesse dei programmatori rispetto al progetto, cosa non facile vista la notevole competizione tra i tantissimi progetti

nell'attirare l'attenzione ed il contributo degli sviluppatori. L'obiettivo delle imprese deve essere, in questo senso, offrire sfide e progetti divertenti ai programmatori e, allo stesso tempo, creare prodotti che siano abbastanza semplici da attirare utenti.

Un altro tema riguarda le risorse che le software companies devono devolvere alla comunità al fine di creare e mantenere le relazioni con essa, che sono talvolta ingenti: si pensi al caso di MySQL, la cui comunità è stata creata dopo il rilascio del database, e durante la quale MySQL ha continuato a rilasciare aggiornamenti.

Esiste poi la necessità per l'impresa di allineare le diverse concezioni del lavoro tra l'impresa e gli sviluppatori della comunità, concepito, nel primo caso, come lo svolgimento di compiti routinizzati, e come vere e proprie sfide intellettuali nel secondo.

5.5 Open source come strategia di open innovation

L'open source come strategia di open innovation si caratterizza per due elementi chiave: la presenza di diritti d'uso condivisi della tecnologia e lo sviluppo collaborativo della tecnologia. A differenza di molti partecipanti individuali ai progetti open source, le imprese devono tener conto di un terzo fattore: ottenere un ritorno economico dal proprio investimento. E' così possibile individuare quattro possibili approcci per l'integrazione dell'innovazione esterna proveniente dalle comunità open source (West & Gallagher, 2006).

Una prima forma di open innovation si ha con la ricerca e sviluppo consorziata (*R&D pooling*). La ricerca cooperativa ha come funzione il risparmio dei costi, ma è anche tipica di quelle situazioni in cui le imprese non possono appropriarsi degli spillovers derivanti

dalle proprie ricerche, tipicamente nei settori in buona parte dipendenti sulle scienze avanzate.

E' stato questo il caso dell'Open Source Licensing Lab rispetto a Linux e del già citato caso di Mozilla.

L'Open Source Development Lab è un consorzio, fondato nel 2000, avente l'obiettivo di essere riconosciuto come il centro di gravità del business gravitante attorno a Linux e di accelerarne l'utilizzo da parte delle imprese operanti nel settore dell'informatica. I partecipanti a questo consorzio sono eterogenei, e possono essere raggruppati in quattro categorie: i venditori di pc e di sistemi di telecomunicazione, produttori di microprocessori, distributori di Linux e sviluppatori di software ad esso complementari.

Nel caso di Mozilla, invece, nel 2003, al termine del processo di sviluppo del progetto seguito al rilascio del codice sorgente nel 1998 da parte di Netscape, imprese come la IBM, HP e Sun, specializzate nella vendita di sistemi Unix, rimasero senza un browser di riferimento per il collegamento ad Internet delle loro workstation. Per questo motivo assegnarono alcuni loro ingegneri informatici al progetto, al fine di accelerarne lo sviluppo e assicurare la compatibilità di Mozilla ai propri sistemi.

Attraverso il r&d pooling, sia nel caso di Mozilla che dell'OSDL le imprese massimizzano i profitti derivante dal loro investimento alla comunità ottenendo una piattaforma comune che potranno adattare alle proprie esigenze. Nel caso di Mozilla essi potranno integrare il browser nei propri sistemi, mentre nel caso dell'OSDL i membri potranno cooperare nel raggiungere lo scopo comune, mentre competeranno nella vendita dei propri prodotti.

Un'altra possibile strategia di open innovation attuabile dalle software companies è il ricorso agli spinouts, con il trasferimento delle proprie tecnologie al di fuori dei confini aziendali, mantenendo comunque un coinvolgimento aziendale (West & Gallagher, 2006). In sostanza con gli spinouts le imprese progetti interni di sviluppo in progetti open source visibili.

Il vantaggio derivante da questa strategia consiste nella possibilità di generare una domanda, a favore del donatore, per i prodotti ed i servizi complementari a quanto donato. E' stato questo ad esempio il caso di IBM con il linguaggio di programmazione Java, inizialmente lanciato dalla Sun Microsystems, a cui è seguito il compilatore per Java Jives. Questa strategia è utile per le tecnologie che non sono ancora state commercializzate (come nel caso di Jikes) o che diventeranno commodities e perciò di ridotto valore, come nel caso dei tool di sviluppo per Java. Gli obiettivi di questa strategia consistono in:

- aiutare l'impresa ad imporre una tecnologia come standard di fatto, e quindi evitare che, nell'orizzonte temporale di riferimento, essa debba ridisegnare la propria tecnologia per conformarsi a standand concorrenti;
- attirare miglioramenti e prodotti complementari che rendono la tecnologia più attraente;
- insieme, la tecnologia ed il componente rendono possibile la vendita di altri prodotti collegati;
- condividere la tecnologia con gli utenti, che poi saranno futuri clienti delle tecnologie complementari;

- ridurre i costi legati allo sviluppo della tecnologia, al contempo mantenendo una conoscenza della stessa tale da offrire supporto ai propri clienti.

Una terza strategia consiste nella vendita di prodotti complementari, che può consistere nella focalizzazione, rispetto al paradigma hardware-software (Teece, 1986), su quelle parti dell'architettura di un sistema caratterizzate da una rapida evoluzione e da una ridotta imitabilità, rispetto invece alle componenti mature o "commodificate". E' stato questo, ad esempio, il caso del browser Safari, sviluppato da Apple a partire dalle librerie del browser open source Konqueror, sotto licenza GNU LGPL (West & Gallagher, 2006). Il codice sorgente di Safari è stato solo in parte rilasciato da Apple per la parte definita Webcore, sotto una licenza simile alla BSD, mentre nessun codice è stato rilasciato per la parte rimanente, compresa la GUI.

Fino al 1997 il browser di default del sistema operativo dei pc Apple, il Mac OS X, era Navigator, che Netscape in quell'anno smise di produrre per la Apple, senza il subentro di Mozilla. Successivamente, il browser è diventato Microsoft Explorer, la cui versione per OS X risultava sempre costantemente meno aggiornata rispetto alla versione per Windows, finchè, nel 2003, la Microsoft non ha cessato definitivamente di adattare il proprio browser per la Apple.

Un'altra strategia è quella di ricorrere alle licenze duali, in cui il codice sviluppato dall'impresa viene rilasciato sia con licenza open source, gratuitamente, sia con licenza proprietaria, come prodotto commerciale. In questo modo, gli utenti più sensibili al prezzo possono acquisire gratuitamente un prodotto privo dell'assistenza

offerta dall'impresa e di limiti alla redistribuzione, in cambio del sostegno all'impresa nello sviluppo del software. Invece, altre tipologie di utenti, come quelli business, avranno la possibilità di acquistare un prodotto completo dell'assistenza.

Come evidenziato precedentemente, l'adozione di licenze duali limita l'afflusso di innovazione esterna, a causa della maggiore diffidenza degli sviluppatori del progetto open source rispetto alla software company, ma al contempo si caratterizza come una vera e propria strategia di marketing per attrarre più utenti al prodotto e creare effetti di network.

Un'ultima strategia consiste nella possibilità per le software firms di trarre profitto dai componenti regalati dagli utenti, ed è ad esempio questo il caso dei "mods" dei giochi per pc (Nieborg, 2005).

I grandi concorrenti dei giochi per pc sono la Playstation Nintendo e il Microsoft Xbox, caratterizzati da una risoluzione grafica e quindi da un realismo delle immagini decisamente maggiore rispetto ai primi. L'unico elemento possibile di vantaggio dei giochi per pc è rappresentato quindi dalla possibilità per gli utenti di modificare il gioco (West & Gallagher, 2006), attraverso dei kit di sviluppo che consentono agli utenti di creare scenari, nuove ambientazioni o reinventare totalmente il gioco.

In questo modo il ciclo di vita del prodotto, abbastanza breve, può essere rivitalizzato dagli stessi utenti, che hanno le stesse motivazioni dei programmatori open source, cioè la ricerca di un beneficio personale in termini di evoluzione del gioco, le motivazioni intrinseche e la segnalazione all'esterno delle proprie capacità di game developer. Esempi noti di mods creati dagli utenti sono quelli relativi

ai *first person shooter*, come l'"Unreal Tournament" della Unreal Universe, e il "Desert Combat" del Battlefield franchise.

5.6 I business models

L'open innovation, affinchè sia veramente tale, richiede la capacità dell'impresa di convertire il potenziale tecnologicio, interno ed esterno, acquisito in valore economico. Tale capacità è espressa dal business model, che assolve le seguenti funzioni (Chesbrough, 2003; Chesbrough & Rosenbloom, 2002):

- definire la value proposition, cioè il valore creato per gli utenti attraverso l'offerta basata sulla tecnologia;
- identificare il segmento di mercato, cioè gli utenti a cui la tecnologia può essere utile e lo scopo per il quale potrebbe essere usata;
- definire la struttura della catena del valore dell'impresa, e determinare gli assets complementari necessari per sostenere la posizione dell'impresa nella catena;
- specificare il meccanismo di generazione del profitto e stimare la struttura dei costi e i margini per la produzione dell'offerta, data la proposizione del valore e la struttura della catena del valore scelti;
- descrivere la posizione dell'impresa all'interno del network del valore che lega fornitori e clienti, ed identificare le potenziali imprese complementari e quelle concorrenti;
- formulare la strategia competitiva attraverso la quale l'impresa innovatrice otterrà e manterrà il proprio vantaggio competitivo sugli altri.

Nell'ambito del software open source, le possibili fonti di profitto dipendono dal business model adottato dall'impresa, che può assumere il ruolo di distributore, di produttore di software (GPL e non GPL), e di fornitore di servizi terzi (Krishnamurthy, 2005).

Il *distributore* offre l'accesso al codice sorgente e al software, come ad esempio nel caso di Red Hat, Caldera e SUSE con Linux. I distributori possono basare la propria offerta sui seguenti modelli:

1. Offrendo il prodotto su CD piuttosto che il suo download a tutti gli utenti che richiedono un supporto fisico per il software;
2. Offrendo servizi ed assistenza al segmento business, come l'aiuto nell'assistenza, nell'implementazione di soluzioni customizzate e nell'addestramento del personale all'uso del software;
3. Offrendo servizi per l'aggiornamento, ed in questo modo l'utente (in genere un'impresa) si può assicurare la fornitura dell'ultima versione del programma.

Il *produttore* di software (non su licenza tipo GPL) può trarre vantaggio dal software open source in due modi: attraverso l'incorporazione del suo codice sorgente in un programma più ampio, oppure attraverso il suo bundling con un proprio prodotto a codice sorgente chiuso. Microsoft, ad esempio, ha utilizzato parti di programma floss sotto licenza BSD che, come visto precedentemente (cfr. capitolo 2), a differenza delle licenze tipo GPL, non impone a chi incorpora il codice sorgente in propri prodotti l'obbligo di redistribuirli sotto la stessa licenza.

Il *produttore* di software su licenza GPL e non ha, di fronte a se, sei possibili business model (Daffara, 2007).

Il primo consiste nel ricorso alle licenze duali, attraverso le quali diventa possibile per le software companies cedere il software sia sotto licenze open source che sotto licenza commerciale. Questo modello è utilizzato principalmente da imprese che producono software o tools destinati a sviluppatori, e funziona grazie al ricorso alla licenza GPL, imponendo quindi che la parte del programma che si basa sul programma protetto da questa licenza sia redistribuito alle stesse condizioni. Al contempo, si consente alla software company di conservare un diritto di sfruttamento esclusivo sulla parte del software invece coperta da una licenza d'uso di tipo commerciale. Sulla carta questo business model sembra essere il giusto compromesso tra i principi delle comunità open source e le esigenze commerciali delle imprese, in realtà è stato evidenziato (cfr. par. 5.3) come in realtà questa situazione comporti un ridotto apporto da parte della comunità open source, visti i maggiori rischi di hijack avvertiti dai contributori floss.

Un altro possibile business model consiste nella netta divisione tra il prodotto open source base e la versione commerciale dello stesso che è basato sul codice open source ma presenta l'aggiunta di plug ins proprietari. La licenza che meglio si presta a questo modello è la Mozilla Public License, e in questo modo l'impresa non ha bisogno di ricorrere alle licenze duali, scelta, quest'ultima, che, come visto precedentemente, riduce la propensione a collaborare da parte della comunità floss. I limiti di questa scelta consistono nella possibilità che altre software companies possano costruire un proprio prodotto, aggiungendo propri plugins, sulla stessa base open source.

Un terzo modello è il cd. badgeware, che consiste in una rivisitazione della licenza pubblica Mozilla e della BSD, grazie alla quale si impone la visualizzazione del marchio o di altri elementi visibili dell'impresa sull'interfaccia utente del software, anche se quest'ultimo viene rivenduto attraverso rivenditori indipendenti. In questo modo l'impresa può tutelare il proprio marchio, e gli sviluppatori ricevere il riconoscimento del proprio contributo grazie all'esatta identificazione del software.

Un ulteriore business model è offerto dai *product specialists*, ovvero dalle imprese che hanno creato o sostengono un singolo software e lo distribuiscono con licenze floss. Il loro core business è basato su servizi complementari, come la formazione e il supporto al cliente, ed è basato sull'assunto che questi servizi possono essere offerti in modo ottimale da chi, come loro, ha contribuito allo sviluppo del software. Il punto debole di questa strategia è, nonostante tutto, le ridotte barriere all'entrata che gli eventuali competitors si trovano davanti, in quanto l'unico loro ostacolo è l'acquisizione di skills specialistiche sul software open source alla base del business model.

Esistono poi i providers di piattaforme, ovvero software companies che creano un prodotto complesso, una vera e propria piattaforma integrata, attraverso la selezione, l'integrazione, il supporto e l'assistenza di un insieme di programmi. Queste piattaforme sono concesse in uso secondo le licenze floss e, al contempo, prevedono come forma di tutela della piattaforma, il ricorso al marchio o ad una tutela del diritto d'autore per il proprio logo.

L'ultimo modello analizzato è quello delle imprese che offrono un servizio di selezione e/o di consulenza, che propongono ai propri clienti l'analisi dei software floss più adatti alle loro esigenze e, spesso, sono imprese che non presentano sviluppatori al proprio interno, non incidendo quindi in alcun modo sulle comunità floss.

E' quindi evidente come le imprese commerciali, nel creare un modello di business basato sul software open source, adottino generalmente una strategia ibrida, e questo è confermato da una ricerca europea, condotta nel 2004, su un campione di circa 770 imprese operanti in Finlandia, Germania, Italia, Portogallo e Spagna (Bonaccorsi, Piscitello, Merito, & Rossi, 2006).

Sulla base di questa ricerca, sono stati individuati diversi clusters di software companies, le cui strategie verso il mondo floss variano, da una pura strategia open source, che consiste nell'adozione di un business model basato esclusivamente sul software floss, ad una strategia ibrida, in cui il ricorso al codice aperto via via si riduce, e che è tipica, rispettivamente, dei clusters denominati *more OSS oriented* (MOSS), *less OSS oriented* (LOSS) e *no OSS oriented* (NOSS), queste ultime imprese il cui business model si basa esclusivamente su programmi proprietari a codice chiuso.

Le imprese appartenenti ai due cluster caratterizzati da una strategia ibrida, MOSS e LOSS, sono stati individuati a partire da alcune variabili:

- Percentuale di ricavi derivanti dalle soluzioni oss rispetto ai ricavi totali conseguiti nel 2003;
- Percentuale di soluzioni oss rispetto al totale delle soluzioni offerte;

- Tipi di soluzioni offerte, in cui il valore 3 indica soluzioni prevalentemente oss, il 2 soluzioni oss e proprietarie in egual misura, e 3 soluzioni proprietarie;
- Intensità d'uso della licenza GPL, sul totale delle licenze utilizzate dall'impresa;
- Atteggiamento rispetto ai valori delle comunità open source, variabile fittizia che assume valore 1 se l'intervistato motiva il proprio coinvolgimento nel movimento open source con il desiderio di agevolare la diffusione di standards aperti.

Come visibile nella tabella 5.2, sotto riportata, la distribuzione nei quattro gruppi (POSS, MOSS, LOSS e NOSS) è abbastanza simile in Finlandia, Germania e Italia, mentre la percentuale di imprese NOSS del campione è maggiore in Spagna e Portogallo.

Le imprese con un maggiore grado di apertura (POSS e MOSS) sono risultate essere più piccole delle altre, in quanto buona parte di esse, il 65,17%, erano piccole e medie imprese aventi un numero di dipendenti inferiore a 10. Inoltre buona parte di esse sono risultate essere imprese di recente costituzione, dato che sono entrate nel mercato recentemente: l'anno mediano di fondazione era il 2000 per le imprese MOSS e il 2001 per le imprese POSS. La maggior parte delle imprese, l'83, 56%, dell'intero campione ha adottato il software open source dopo il 1998, anno della creazione dell'Open Source Definition.

Tab. 5.2: Distribuzione per paese dei quattro gruppi

Paese	POSS	MOSS	LOSS	NOSS	Totale
Finlandia	2	15	40	78	135
	1,48	11,11,	29,63	57,78	100
	10,53	21,43	15,18	15,18	17,56
Germania	0	9	27	57	93
	0	9,68	29,03	61,29	100

	0	12,86	16,27	11,09	12,09
Italia	6	22	63	152	243
	2,47	9,05	25,93	62,55	100
	31,58	31,43	37,95	29,57	31,6
Portogallo	0	3	12	83	98
	0	3,06	12,24	84,69	100
	0	4,29	7,23	16,15	12,74
Spagna	11	21	24	144	200
	5,5	10,5	12,0	72,0	100
	57,89	30	14,46	28,02	26,01
Totale	19	70	166	514	769
	2,47	9,1	21,59	66,84	100
	100	100	100	100	100

Fonte: Bonaccorsi, Piscitello, Merito, & Rossi (2006)

Le software companies hanno come target market le piccole e medie imprese, mentre le aziende POSS e MOSS risultano lavorare per il settore pubblico.

Infine le software firms POSS e MOSS hanno fondatori e dipendenti più istruiti, e questo a conferma del fatto che il paradigma open source si basa sulla capacità del programmatore di analizzare il codice sorgente e di modificarlo, piuttosto che sull'applicazione del software esistente.

Tab. 5.3: Le imprese e i brevetti

Pensiamo che i brevetti[a]	SI (%)			
	MOSS N=49	LOSS N=142	NOSS N=370	TOT. N=561
Promuovano l'innovazione	4,08	27,46	37,57	32,09
Danneggino l'innovazione	73,47	52,82	34,86	42,78
Non impediscano ai ns competitor l'ingresso nel mercato	71,35	75,35	67,35	71,70
Necessitino di un iter legale troppo lungo	87,76	69,72	65,95	68,81
Siano costosi	81,63	76,06	70,00	72,55
Limitino il rilascio di nuove versioni	61,22	50,70	35,41	41,53
Offrano informazioni sui prodotti delle altre imprese	18,37	28,87	25,68	25,85

[a]Possibili risposte: si, no, forse.
Fonte: Bonaccorsi, Piscitello, Merito, & Rossi (2006)

Per quanto riguarda il problema dell'appropriabilità, Bonaccorsi et al. (2006) hanno chiesto, alle imprese del campione, quali erano le proprie idee sull'efficacia dei brevetti nel settore del software, ottenendo come risposta, conforme ai risultati ottenuti in altri settori (Levin, Klevorick, Winter, Gilbert, & Griliches, 1987), che in realtà il brevetto non rappresenta un incentivo per l'innovazione, mentre per le imprese MOSS e LOSS esso addirittura costituisce un ostacolo per l'innovazione (tab. 5.3)

Gli Autori evidenziano invece un atteggiamento più positivo nei confronti delle licenze d'uso, la cui importanza economica per l'impresa è stata misurata ricorrendo ad una scala di Likert a 5 punti. Buona parte degli intervistati ha risposto attribuendo un valore elevato alle licenze d'uso, mentre le imprese MOSS hanno attribuito punteggi inferiori rispetto alle LOSS.

Tab. 5.4: Le imprese e le licenze d'uso

Pensiamo che le licenze[a]	SI (%)			
	MOSS N=49	LOSS N=142	NOSS N=370	TOT. N=561
Contribuiscano notevolmente alle vendite dei ns prodotti	22,45	43,66	54,59	49,02
Richiedano accordi contrattuali complessi	28,57	29,58	18,92	22,46
Limitino il rilascio di nuove versioni	22,45	17,61	12,97	14,97
Ci rendano dipendenti dal ns fornitore/licenziante	34,69	43,66	39,46	40,11
Ci aiutino a controllare i ns. prodotti	36,73	65,49	67,30	64,17
Creino reti di imprese per la condivisione della conoscenza	28,57	29,58	32,16	31,19
Ci aiutino a recuperare gli investimenti in R&S	36,73	54,93	59,19	56,15

[a]Possibili risposte: si, no, forse.
Fonte: Bonaccorsi, Piscitello, Merito, & Rossi (2006)

Per quanto riguarda, invece, la composizione del portafoglio prodotti, in virtù del ruolo assunto come assets complementare dalle comunità open source, si evidenzia come le imprese che hanno adottato il paradigma dell'open source abbiano un portafoglio prodotti più ampio rispetto alle software companies che si basano sul paradigma del software a codice chiuso (tab. 5.5).

Tab. 5.5: Numero di prodotti per tipologia di impresa

Cluster	N.	Min	Max	Media	p50	p75	p95	Dev. st.
POSS	19	3	18	9,10	8	13	17	3,77
MOSS	70	0	18	9,17	10	14	18	5,46
LOSS	166	0	18	6,72	5	11	17	5,28
NOSS	514	0	18	4,82	3	8	12	4,55
TOTALE	769	0	18	5,65	4	10	16	4,99

Fonte: Bonaccorsi, Piscitello, Merito, & Rossi (2006)

Tale valore non risulta essere correlato con la dimensione dell'impresa, dato che il coefficiente di Pearson assume valori poco significativi.

Risultati simili sono stati ottenuti considerando l'offerta di servizi complementari al software (tab. 5.6): rispetto alle imprese NOSS, le software firms che si basano sul paradigma open source offrono un maggior numero di servizi complementari e ciò è reso possibile dallo sfruttamento di fonti di innovazione esterne all'impresa.

Tab. 5.6: Numero di categorie di servizi offerti dalle imprese

Cluster	N.	Min	Max	Media	p50	p75	p95	Dev. st.
POSS	19	4	11	8,58	9	11	11	2,06
MOSS	70	0	11	9,09	10	11	11	2,17
LOSS	166	0	11	7,61	8	10	11	2,73

NOSS	514	0	11	7,44	8	10	11	2,73
TOTALE	769	0	11	7,65	8	10	11	2,71

Fonte: Bonaccorsi, Piscitello, Merito, & Rossi (2006)

In questo senso, il paradigma dell'open innovation riduce in modo notevole il costo della conoscenza, generando quindi un nuovo business model, in cui i costi fissi per la produzione di beni e servizi è minimo, i costi per gli assets complementari sono trascurabili, e tutta l'attività dell'impresa è indirizzata alla customizzazione, all'adattamento del software alle esigenze del cliente, che si riflette in maggiori costi variabili. Questo modello, quindi, risulta avere un break even point molto basso, ed è particolarmente profittevole per le piccole imprese.

Tab. 5.7: Offerta delle imprese in 18 categorie di prodotti

Classe[a]	Categoria di prodotto	NOSS		LOSS		MOSS		POSS	
		N = 514		N = 166		N = 70		N = 19	
		N	%	N	%	N	%	N	%
S	Web servers	163	31,71	93	56,02	51	72,86	11	57,89
S	Altri tipi di server	153	29,77	73	43,98	47	67,14	11	57,89
N	Sistemi di backup	146	28,40	57	34,34	40	57,14	10	52,63
N	Firewall	135	26,26	73	43,98	39	55,71	8	42,11
N	Antispam	117	22,76	69	41,57	39	55,71	8	42,11
N	Antivirus	130	25,29	68	40,96	37	52,86	7	36,84
N	User e identity management	123	23,93	53	31,93	35	50	10	52,63
W	Client di posta elettronica	133	25,88	63	37,95	40	57,14	10	52,63

W	Instant messaging	82	15,95	43	25,90	28	40	11	57,89
W	Web browser	37	7,20	38	22,89	22	31,43	3	15,79
W	Sistemi di firma digitale	41	7,98	28	16,87	13	18,57	6	31,58
W	Content Management Systems	127	24,71	73	43,98	45	64,29	12	63,16
W	Soluzioni per l'e-commerce	150	29,18	70	42,17	38	54,29	10	52,63
W	Tools per l'e-learning	63	12,26	39	23,49	27	38,57	10	52,63
O	Software gestionali	333	64,79	84	50,60	38	54,29	13	68,42
O	Software per la gestione dei dati	274	53,31	82	49,40	44	62,86	16	84,21
O	Sistemi di workflow	130	25,29	42	25,30	25	35,71	10	52,63
O	Pacchetti per l'office automation	140	27,24	67	40,36	34	48,57	7	36,84

[a]I prodotti sono stati raggruppati in S: prodotti per i server; N: prodotti per le infrastrutture per le reti; W: prodotti per la Rete; O: altri tipi di prodotti. Fonte: (Bonaccorsi, Piscitello, Merito, & Rossi, 2006)

Fonte: Bonaccorsi, Piscitello, Merito, & Rossi (2006)

L'analisi dei prodotti realizzati dai quattro gruppi consente di evidenziare un dato interessante, ovvero come le imprese che non basano il proprio business model sul software open source realizzino prodotti per il quali è già presente un *dominant design*, e ciò è evidenziato dal fatto che le imprese NOSS operino prevalentemente con prodotti per l'office automation, nei software gestionali e nei database, in cui esistono leader del mercato che hanno imposto il loro modello (uno fra tutti Microsoft Office nel settore dell'office automation e SAP nei software gestionali e nei database).

Le imprese che hanno adottato il paradigma dell'innovazione aperta, invece, sono presenti in segmenti in cui non si è ancora stabilito un modello di riferimento, e quindi sono presenti con prodotti quali antivirus, antispam, firewall, content management systems (Bonaccorsi, Piscitello, Merito, & Rossi, 2006).

Bibliografia

American Sociological Association. (s.d.). *Franklin H. Giddings*. Tratto il giorno Ottobre 11, 2008 da Sito Web American Sociological Association: http://www.asanet.org/

Apache Software Foundation. (2004, January). *Apache License, Version 2.0*. Tratto il giorno luglio 2008, 25 da Sito Web Apache Software Foundation: http://www.apache.org/licenses/LICENSE-2.0

Arnould, E. J., & Thompson, C. J. (2005). Consumer culture theory (CCT) - Twenty years of research. *Journal of Consumer Research , 31*, 868-882.

Arnould, E. J., & Thompson, C. J. (2005, March). Consumer Culture Theory (CCT) - Twenty years of Research. *Journal of Consumer Research* , p. Vol. 31 868-882.

Arora, A., Fosfuri, A., & Gambardella, A. (2001). *Markets for technology - The economics of innovation and corporate strategy.* Cambridge, MA.; London, UK: The MIT Press.

Arora, A., Fosfuri, A., & Gambardella, A. (2001). *Markets for technology - The economics of innovation and corporate strategy.* Cambridge, Mass. - London, Uk: The MIT Press.

Arvidsson, A. (2005). Brands - A critical perspective. *Journal of Consumer Culture , 5* (2), 235-258.

Bagozzi, R. P. (2000). On the concept of intentional social action in consumer behavior. *Journal of Consumer Research , 27* (December), 388-396.

Bagozzi, R. P., & Dholakia, U. M. (2006). Open source software user communities: a study of participation in Linux user groups. *Management Science , 52* (7), 1099-1115.

Baker, R., & Yandle, B. (1994). Financial markets and the AT&T antitrust settlement. *Eastern Economic Journal , 20* (4), 429-440.

Barbrook, R. (2005). The Hi-tech gift economy. *First Monday* (Special Issue no. 3).

Benkler, Y. (2002). Coase's penguin or Linux and the nature of the firm. *Yale Law Journal , 112*, 369-446.

Benussi, L. (2006). The history of the Free/Libre/Open Source Software: stories from the Open Source evolution. In A. Bonaccorsi, & C. Rossi, *Economic perspectives on open source software - Intellectual property, knowledge-based communities, and the software industry* (p. 9-59). Milano: Franco Angeli.

Bitzer, J., Shrettl, W., & Schroder, P. J. (2007). Intrinsic motivation in open source software development. *Journal of Comparative Economics , 35*, 160-169.

Bonaccorsi, A., & Rossi, C. (2003). *Comparing motivations of individual programmers and firms to take part in the Open Source movement. From community to business.* Pisa: Laboratorio di Economia e Gestione SSSUP - Working Paper.

Bonaccorsi, A., & Rossi, C. (2003). Licensing schemes in the production of Open Source Software - An empirical investigation. *Mimeo* . Pisa: S. Anna School of Advanced Studies.

Bonaccorsi, A., Lorenzi, D., Merito, M., & Rossi, C. (2007). Come le imprese partecipano ai progetti open source - Alcune evidenze preliminari e un'agenda per la ricerca. *Economia e Politica Industriale* (3), 201-213.

Bonaccorsi, A., Merito, M., Rossi, C., & Piscitello, L. (2005). *Open source and the software industry - How firms do business out of an open innovation paradigm.* Mimeo: Working Paper - preliminary draft.

Bonaccorsi, A., Piscitello, L., Merito, M., & Rossi, C. (2006). Profiting from "Open Innovation". Teece's building blocks meet the Open Source production paradigm. In A. Bonaccorsi, & C. Rossi, *Economic perspectives on open source software - Intellectual property, knowledge-based communities, and the software industry* (p. 222--248). Milano: Franco Angeli.

Chesbrough, H. (2003). *Open innovation - The new imperative for creating and profiting from technology.* Boston: Harvard Business School Press.

Chesbrough, H., & Rosenbloom, R. S. (2002). The role of the business model in capturing value from innovation: evidence from Xerox Corporation's technology spin-off companies. *Industrial and corporate change , 11* (3), 529-555.

Ciffolilli, A. (2004). *The economics of open source hijacking and declining quality of digital information resources: a case for copyleft.* Ancona: Dipartimento di Economia Università Politecnica delle Marche.

Cohen, W. M., Goto, A., Nagata, A., Nelson, R. R., & P., W. J. (2002). R&D spillovers, patents, and the incentive to innovate in Japan and the United States. *Research Policy , 31*, 1349-1367.

Cohen, W. M., Nelson, R. R., & Walsh, J. P. (2000). *Protecting their intellectual assets: appropriability conditions and why U.S. firms patent (or not).* Cambridge, MA.: National Bureau of Economic Research Working Papers.

Cova, B. (1997). Community and consumption - Towards a definition of the "linking value" of products or services. *European Journal of Marketing , 31* (3/4), 297-316.

Cova, B., & Dalli, D. (2007). Community made: from consumer resistance to tribal entrepreneurship. In S. Borghini, M. A. Mc Grath, & C. Otnes (A cura di), *European Advances in Consumer Research Volume 8* (p. in corso di pubblicazione). Milan: Association for Consumer Research.

Cova, B., & Pace, S. (2006). Brand community of convenience products: new forms of consumer empowerment - the case "my Nutella the community". *European Journal of Marketing , 40* (9/10), 1087-1105.

Cucco, R., & Dalli, D. (2008). 500 wants you: un caso di convergenza tra retro-marketing, cooperative innovation e community management. *Economia & Management* (2), 53-72.

Daffara, C. (2007). *Business models in FLOSS-based companies.* Mimeo.

Dahlander, L., & Magnusson, M. G. (2005). Relationship between open source software companies and communities: observations from Nordic firms. *Research Policy , 34*, 481-493.

Dahlander, L., & Wallin, M. W. (2006). A man on the inside: Unlocking communities as complementary assets. *Research Policy , 35*, 1243-1259.

David, P. A. (2004). Understanding the emergence of "open science" institutions: functionalist economics in historical context. *Industrial and Corporate Change , 13* (4), 571-589.

De Certeau, M. (1990). *L'invenzione del quotidiano.* Roma: Edizioni Lavoro.

Firat, F. A. (2006). Theoretical and philosophical implications of postmodern debates: some challenges to modern marketing. *Marketing Theory , 6* (2), 123-162.

Firat, F. A., & Venkatesh, A. (1995, December). Liberatory postmodernism and the reenchantment of consumption. *Journal of Consumer Research* , p. 239-267.

Fournier, S. (1998). Consumer resistance: societal motivations, consumer manifestations, and implications in the marketing domain. *Advances in Consumer Research vol. 25* (p. 88-90). Association for Consumer Research.

Freeman, S. (2007). The material and social dynamics of motivation: contributions to open source language software development. *Science studies , 20* (2), 55-77.

Giesler, M. (2006). Consumer gift systems. *Journal of Consumer Research , 33*, 283-290.

Gilbert, R. (2004). *Converging doctrines? US and EU antitrust policy for the licensing of intellectual property.* Berkeley: University of California, Working Paper no. CPC04-44.

Gilbert, R., & Shapiro, C. (1990). Optimal patent length and breadth. *The RAND Journal of Economics , 21* (1), 106-112.

Graham, S. J., & Mowery, D. C. (2006). The use of intellectual property in software: implications for open innovation. In H. Chesbrough, W. Vanhaverbeke, & J. West, *Open innovation: researching a new paradigm* (p. 184-201). Oxford: Oxford University Press.

Hardin, G. (1968). The tragedy of the commons. *Science , 162* (3859), 1243-1248.

Harhoff, D., Henkel, J., & Von Hippel, E. (2003). Profiting from voluntary information spillovers: how users benefit from free revealing their innovations. *Research Policy , 32*, 1752-1769.

Hars, A., & Ou, S. (2001). Working for free? Motivations of partecipating in open source projects. *Proceedings of the 34th Hawaii International Conference on System Sciences.*

Hebdige, D. (1983). *Sottocultura: il fascino di uno stile innaturale.* Genova: Costa & Nolan.

Heller, M. A., & Eisenberg, R. S. (1998). Can patent deter innovation? The anticommons in biomedical research. *Science , 280*, 698-701.

Hemetsberger, A. (2005). Creative cyborgs: how consumers use the Internet for self realization. In G. Manon, & A. R. Rao (A cura di), *Advances in Consumer Research vol. 32. 32*, p. 653-660. Duluth, MN: Association for Consumer Research.

Hertel, G., Nieder, S., & Herrmann, S. (2003). Motivation of software developers in open source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy , 32*, 1159-1177.

Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivations of software developers in open source projects: an Intenet-based survey of contributors to the Linux kernel. *Research Policy , 32*, 1159-1177.

Holt, D. B. (1997). Poststructuralist lifestyle analysis: conceptualizing the social patterning of consumption in postmodernity. *Journal of Consumer Research , 23* (4), 326-350.

Holt, D. B. (2002). Why do brands cause trouble? *Journal of Consumer Research , 29*, 70-90.

IDC. (2007, Maggio 31). *Worldwide Revenue from Standalone Open Source Software Will Grow 26% to Reach $5.8 Billion by 2011, IDC Research Indicates* . Tratto il giorno Ottobre 4, 2008 da Sito web IDC: http://www.idc.com/getdoc.jsp;jsessionid=LVAKJ3AXG5SG2CQJAFICFGAKBEAUMIWD?containerId=prUS20711507

Katz, R., & Allen, T. J. (1982). Investigating the not invented here (NIH) syndrome: A look at the performance, tenure, and communication patterns of 50 R & D Project Groups. *R&D Management , 12* (1), 7-20.

Kingston, W. (2001). Innovation needs patent reform. *Research Policy , 30*, 403-423.

Kingston, W. (2001). Innovation needs patent reform. *Research Policy* , 403-423.

Klevorick, A. K., Levin, R. C., Nelson, R. R., & Winter, S. G. (1995). On the sources and signficances of interindustry differences in technological opportunities. *Research Policy , 24*, 185-205.

Kollock, P. (1999). The economies of online cooperation - Gift and public goods in cyberspace. In M. A. Smith, & P. Kollock, *Communities in cyberspace* (p. 219-237). London: Routledge.

Kortum, S., & Lerner, J. (1998). What is behind the recent surge in patenting? *Research Policy , 28*, 1-22.

Kozinets, R. (1999). E-tribalized marketing? The strategic implications of virtual communities of consumption. *European Management Journal , 17* (3), 252-264.

Kozinets, R. V. (2002). Can Consumers Escape the Market? Emancipatory Illuminations from Burning Man. *Journal of Consumer Research , 29* (1), 28-38.

Krishnamurthy, S. (2005). An analysis of open source business models. In J. Feller, B. Fitzgerald, S. A. Hissam, & K. R. Lakhani, *Perspectives on free and open source software* (p. 279-296). Cambridge, Massachusetts; London, England: The MIT Press.

Krishnamurthy, S. (2002). Cave or community? An empirical investigation of 100 mature open source software projects. *First Monday , 7* (6).

Lakhani, K. R., & Wolf, R. G. (2005). Why hackers do what they do: understanding motivation and effort in free/open source software software. In J. Feller, B. Fitzgerald, S. A. Hissam, & K. R. Lakhani, *Perspectives on free and open source software* (p. 3-21). Cambridge, Mass.; London, UK: The MIT Press.

Lamoreaux, N. R., & Sokoloff, K. L. (2002). *Intermediaries in the U.S. market for technology, 1870-1920.* Cambridge, MA.: National Bureau of Economic Research - working paper no. 9017.

Lamoreaux, N. R., & Sokoloff, K. L. (1999). *Inventive activity and the market for technology in the United States, 1840-1920.* Cambridge, MA.: National Bureau of Economic Research - working paper no. 7107.

Lamoreaux, N. R., & Sokoloff, K. L. (1997). *Inventors, firms, and the market for technology: U.S. manufacturing in the late Nineteenth and earlyTwentieth centuries.* Cambridge, MA.: National Bureau of Economic Research Historical Paper no. 98.

Lee, G. K., & Cole, R. E. (2003). From a firm-based to a community-based model of knowledge creation: the case of the Linux kernel development. *Organization Science , 14* (6), 633-649.

Lee, S. H. (1999). *Open source software licensing.* Mimeo.

Lerner, J., & Tirole, J. (2002). Some simple economics of open source. *The Journal of Industrial Economics , 50* (2), 197-234.

Levin, R. C., Klevorick, A. K., Winter, S. G., Gilbert, R., & Griliches, Z. (1987). Appropriating the returns from industrial research and development. *Brooking Papers on Economic Activity - Special Issue on Microeconomics , 1987*, 782-831.

Luthje, C. (2000). Characteristics of innovating users in a consumer goods field - An empirical study of sport related product consumers. Arbeitspaper nr. 8.

Majerus, L. (2003). *Court Evaluates Meaning of "Derivative Work" in an Open Source License.* Tratto il giorno Giugno 20, 2008 da Sito Web Società FindLaw: http://library.findlaw.com/2003/Jun/16/132811.html

Mauss, M. (2002). *Saggio sul dono - Forma e motivo dello scambio nelle società arcaiche.* Torino: Einaudi.

Mazzoleni, R., & Nelson, R. R. (1998). The benefits and costs of a strong patent protection. *Research Policy , 27*, 273-284.

Mc Gowan, J. (2001). Legal implications of open source software. *University of Illinois Law Review* , 242-304.

Mc John, S. M. (2000). The paradoxes of free software. *George Mason Law Review* , 25-68.

Mc Kusick, M. K. (1999). Vent'anni di Unix a Berkeley - Dalla AT&T alla ridistribuzione gratuita. In C. DiBona, S. Ockman, & M. Stone, *Open Sources - Voci dalla rivoluzione Open Source* (p. 31-48). Milano: Open Press - Apogeo.

Melucci, A. (1996). *Challenging codes - Collective action in the information age.* Cambridge: Cambridge University Press.

Moisio, R. J., & Askegaard, S. (2002). "Fighting culture" - Mobile phone consumption practices as means of consumer resistance. *Asia Pacific Advances in Consumer Research Vol. 5* (p. 24-29). Association for Consumer Research.

Mozilla Foundation. (s.d.). *Mozilla Public License Version 1.1*. Tratto il giorno 4 luglio, 2008 da Sito della Mozilla Foundation: http://www.mozilla.org/MPL/MPL-1.1.html

Muniz, A. M., & O'Guinn, T. C. (2001). Brand Community. *Journal of Consumer Research , 31* (March), 412-432.

Nelson, R. R. (2004). The market economy, and the scientific commons. *Research Policy , 33*, 455-471.

Net Applications. (2008). *Usage Share Trend for 'Firefox'.* Tratto il giorno Novembre 9, 2008 da Sito web della Net Applications: http://marketshare.hitslink.com/report.aspx?sample=25&qprid=32&qpdt=1&qpct=4&qpcustom=Firefox&qptimeframe=D&qpsp=3561&qpnp=31

Nieborg, D. B. (2005). Am I mod or not? - An analysis of First Person Shooter modification culture. *Creative Gamers Seminar - Exploring Participatory Culture in Gaming.* Tampere: University of Tampere - Hypermedia Laboratory.

Nonaka, I., & Takeuchi, H. (1997). *The knowledge-creating company: creare le dinamiche dell'innovazione.* Milano: Guerini Associati Editore.

Olson, M. (1983). *La logica dell'azione collettiva - I beni pubblici e la teoria dei gruppi.* Milano: Feltrinelli.

O'Mahony, S. C. (2002). *The emergence of a new commercial actor: community managed software Project - Unpublished Ph.D. dissertation.* Stanford: Department of Management Science and Engineering Science University of Stanford.

Ostrom, E. (1990). *Governing the commons - The evolution of institutions for collective action.* Cambridge MA.: Cambridge University Press.

Peñaloza, L., & Price, L. L. (1993). Consumer resistance: a conceptual overview. *Advances in Consumer Research vol. 20* (p. 123-128). Association for Consumer Research.

Perens, B. (1999). La Open Source Definition. In C. DiBona, S. Ockman, & M. Stone, *Open Sources - Voci dalla rivoluzione Open Source* (p. 185-203). Milano: Open Press - Apogeo.

Poster, M. (1992). The question of agency: Michel De Certeau and the history of consumerism. *Diacritics , 22* (2), 94-107.

Raymond, E. S. (1999). *La cattedrale e il bazaar.* Apogeo.

Rheingold, H. (1993). *The virtual community: homesteading on the electronic frontier.* Reading MA: Addison Wesley.

Rivette, K. G., & Kline, D. (2000). *Tesori in soffitta - Scoprire e sfruttare il valore della proprietà intellettuale nell'impresa.* Milano : ETAS.

Ryan, R. M., & Deci, E. L. (2000). Self determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist , 55* (1), 68-78.

Schouten, J. W., & McAlexander, J. H. (1995). Subcultures of consumption: an ethnography of new bikers. *Journal of Consumer Research , 22* (1), 43-61.

Schweik, C. M., & Semenov, A. (2003). The institutional design of open source programming: implications for addressing complex public policy and management problems. *First Monday , 8* (1).

Shah, S. K. (2006). Motivation, governance & the viability of hybrid forms in open source development. Mimeo: Working Paper.

Shah, S. K. (2003). Understanding the nature of participation and coordination in open and gated source software development communities. *Dissertazione dottorale non pubblicata* . Cambridge MA.: Massachusetts Institute of Technology.

Shah, S. (2000). Sources and patterns of innovation in a consumer product field: innovation in sports equipment. M.I.T. Sloan Working Paper # 4105.

Sherry, J. F. (1983). Gift giving in anthropological perspective. *Journal of Consumer Research , 10* (2), 157-168.

Stallman, R. (1999). Il progetto GNU. In C. DiBona, S. Ockman, & M. Stone, *Open Sources - Voci dalla rivoluzione Open Source* (p. 57-76). Milano: Open Press - Apogeo.

Teece, D. (1986). Profiting from technological innovation: implications for integration, collaboration, licensing and public policy. *Research Policy , 15*, 285-305.

Thompson, C. J. (2004). Marketplace mythology and discourses of power. *Journal of Consumer Research , 31*, 162-180.

Thompson, C. J., & Coskuner-Balli, G. (2007). Countervailing market responses to corporate co-optation and the ideological recruitment of consumption communities. *Journal of Consumer Research , 34*, 135-152.

Tuomi, I. (2000). *Internet, innovation, and open source: actors in the netword.* Finland: SITRA .

Ubertazzi, L. C., Galli, P., & Sanna, F. (2003). *Codice del diritto d'autore.* Milano: Giuffrè.

Valimaki, M. (2005). *The rise of Open Source Licensing - A challenge to the use of intellectual property in the software industry.* Finland: Turre Publising.

Veale, K. J. (2003). Internet gift economies: voluntary payment schemes as tangible reciprocity. *First Monday , 8* (12).

Verona, G., & Prandelli, E. (2006). *Collaborative innovation - Marketing e organizzazione per i nuovi prodotti.* Roma: Carocci Editore.

Von Hippel, E. (2005). *Democratizing innovation.* Cambridge, Mass.: The M.I.T. Press.

Von Hippel, E. (1988). *The sources of innovation.* New York - Oxford: Oxford University Press.

Von Hippel, E., & Von Krogh, G. (2002). Exploring the open source software phenomenon - Issues for organization science. Working Paper.

Von Hippel, E., & Von Krogh, G. (2003). Open source software and the "private-collective" innovation model: issues for oganization science. *Organization Science , 14* (2), 209-223.

Von Hippel, E., & Von Krogh, G. (2003). Open source software and the "private-collective" innovation model: issues for organization science. *Organization Science , 14* (2), 209-223.

Von Krogh, G., Spaeth, S., & Lakhani, K. R. (2003). Community, joining, and specialisation in open source software innovation: a case study. *Research Policy , 32*, 1217-1241.

Wathieu, L. e. (2002). Consumer control and empowerment: a primer. *Marketing Letters , 13* (3), 297-305.

Wellman, B., & Gulia, M. (1999). Virtual communities as communities - Net surfers don't ride alone. In M. A. Smith, & P. Kollock, *Communities in cyberspace.* London: Routledge.

West, J., & Gallagher, S. (2006). Patterns of open innovation in Open Source software. In H. Chesbrough, W. Vanhaverbeke, & J. West, *Open Innovation: Researching a new paradigm* (p. 82-106). Oxford: Oxford University Press.

Wikipedia. (2008). *Wikipedia, l'enciclpedia libera.* Tratto il giorno Luglio 23, 2008 da Sito web Wikipedia: http://it.wikipedia.org/wiki/Unix

Wynants, M., & Cornelis, J. (2005). *How open is the future? Economic, social & cultural scenarios inspired by Free & Open Source Software.* Brussels: Crosstalks & VUB - Brussels University Press.

Zeitlyn, D. (2003). Gift economies in the development of open source software: anthropological reflections. *Research Policy , 32*, 1287-1291.

Zwick, D., Bonsu, S. K., & Darmody, A. (2008). Putting consumer to work: 'co-creation' and new marketing govern-mentality. *Journal of Consumer Culture , 8* (2), 163-196.