

Il modello di business dell'Open Source

3.1 Il modello standard di business del software proprietario, fallimenti di mercato e le dinamiche di mercato dell'industria del software

Il software proprietario, cioè quello commercializzato e venduto attraverso licenze, è indifferenziato, ossia distribuito in versioni tutte eguali per poter spalmare il costo di realizzazione su numeri alti di copie e rendere economicamente appetibile il programma e risponde ai requisiti di performance e qualitativi di chi lo ha creato. Spesso il produttore decide la cadenza degli aggiornamenti e la *roadmap* di sviluppo del prodotto per massimizzare il profitto in ogni momento. Si tratta di un mercato con forti economie di scala. Una volta che il software è stato sviluppato, i costi marginali per commercializzare una copia aggiuntiva sono quasi nulli (a limite pari al costo del supporto magnetico del software). Non si stupisce quindi che questo mercato sia dominato da colossi multinazionali come Microsoft che capitalizza in borsa centinaia di miliardi di dollari.

Mediante le grandi imprese del software, quando rilasciano un prodotto, tendono a coprire le esigenze, in termini di funzionalità sviluppate nel software, della maggior parte dei clienti potenziali: empiricamente si è misurata la percentuale del 80% (fonte IDC). Particolarmente penalizzato sono quindi il 20% dei clienti finali, nel caso in cui abbiano bisogno di una determinata funzione e vorrebbero adattare il programma alle loro esigenze invece di adattare le esigenze al programma. Il cliente può rivolgersi ai creatori del software, ma la risposta sarà sicuramente funzione del suo potere economico nei confronti del produttore: nel caso in cui non sia particolarmente rilevante, il signor Tizio potrebbe non vedere mai implementata la sua richiesta, mentre invece la grande azienda multinazionale avrà maggiori probabilità di successo. Entrano allora in gioco, specie nella realtà italiana caratterizzata da un alto numero di PMI, le piccole e medie software house, attori del processo economico in grado di ritagliare, come un artigiano, un prodotto software sulle esigenze specifiche del cliente.

Spesso il rapporto con il cliente è diretto e personale. La software house può proporre al cliente un programma proprietario sviluppato da altri. In questo caso la soluzione è semplice, ma i margini della rivendita sono bassi spesso costituiti dal servizio di installazione, assistenza e formazione “personalizzare” per il cliente un software già esistente. Spesso più che vere modifiche al codice, si risolve in sviluppi che utilizzano funzionalità già presenti nel software proposto, come ad esempio implementare la base dati relativa ad un programma in base alle

specifiche del cliente. Da considerare che se il software di partenza é proprietario, la software house deve prima acquistare dalla casa madre la licenza di sviluppo.

La possibilità di sviluppare ex novo la soluzione in base alle esigenze del cliente diventa rara, in quanto il cliente finale dovrebbe accollarsi l'intero costo di sviluppo e successivo mantenimento di un programma ex novo.

Indipendentemente dal tipo di azione emergono due fatti essenziali:

- La software House guadagna non tanto dal margine di rivendita del software, ma dai servizi aggiuntivi (installazione, personalizzazione, formazione). Sul software che fornisce non ha però alcuna possibilità di modifica.
- Il cliente finale spende soldi per l'acquisto di software fornito con una licenza che non permette di fare nessuna modifica per adattare il prodotto o ottimizzarlo.

Entrambi quindi sono in balia delle scelte della casa produttrice del software e da essa dipendono per le scelte future. Nel caso in cui essa decida di stoppare la produzione e lo sviluppo del software, si dovrà adattare per forza nuovo software con tutte le spese connesse. Vedremo come il paradigma del software Open Source risponda a questa domanda.

Come primo passo cerchiamo di capire la natura del software come bene. Il software é per sua natura un bene non escludibile e non rivale¹. Un bene é escludibile² se il produttore é in grado di impedire di usare il bene a coloro che non ne hanno pagato il prezzo. Se un bene é non escludibile, non é possibile venderlo, visto che si può averlo gratuitamente. Il sistema dei diritti di proprietà intellettuale, seppure sviluppato per garantire incentivi di mercato nella produzione di idee e nello sviluppo di beni software, va in tale contesto perdendo di efficacia. Esso infatti rende privatizzabile un bene pubblico rendendolo artificialmente escludibile.

Riguardo invece alla rivalità all'uso, un bene é rivale se é impossibile che più consumatori lo usino allo stesso tempo. Se l'individuo A indossa un maglione, l'individuo B può indossare contemporaneamente un altro maglione dello stesso modello, ma non lo stesso modello. Quindi, come regola generale possiamo dire che i beni tangibili sono rivali, mentre quelli intangibili No. Nel nostro caso il software é chiaramente un bene non rivale, in quanto più utenti possono usare contemporaneamente lo stesso software, installato chiaramente su hardware differente. Se un bene é non rivale, il suo costo di produzione é lo stesso qualunque

¹ Bosi P., *Corso di scienza delle finanze*, il Mulino Bologna 2003 cap. IV

² T. Cozzi, S. Zamagni, "*Manuale di Economia Politica: un testo europeo*", Il Mulino, Bologna 1999, pp. 52-68, 240-287

sia il numero di utenti, o in alternativa, il suo costo marginale é nullo, a partire ovviamente dal secondo utente in poi.

Siamo quindi di fronte a un dilemma, noto come dilemma dell'innovazione³. Il punto della questione é che il costo del supporto materiale con cui si vende il software non é in relazione con il costo del bene intangibile (il software) incorporato. Ma se si fa pagare per il bene rivale un prezzo al singolo utente, il benessere collettivo non é massimizzato. Efficienza vorrebbe che i beni non rivali fossero offerti gratuitamente, ossia che si paghi solo per il loro supporto materiale e le attività strettamente collegate (come i servizi di installazione e configurazione del software). Ma poiché tale prezzo é molto basso e addirittura quasi nullo (se l'utente ad esempio si installa il software da solo), ne risulta che non é conveniente produrre il bene software. Da qui nasce il dilemma: se il prezzo di un bene (in questo caso un programma software) é più alto del costo di produrne una copia in più, qualcuno, che nel "paradiso degli economisti" lo userebbe, non lo farà. Ma se il prezzo non é più alto del costo di produzione (costo dell'installazione e del supporto materiale), il produttore non potrà coprire i costi necessari per sviluppare il programma software e non vi saranno incentivi economici per l'attività di innovazione. Da qui la necessità di introdurre brevetti, ma così facendo si limita la concorrenza tra le imprese, senza la quale i teoremi dell'economia del benessere non valgono. Siamo quindi in presenza di un fallimento di mercato.

A rendere ancora più grave, dal punto dell'economia del benessere, il mercato del software proprietario é la constatazione che questo mercato gode di rendimenti di scala crescenti che orientano il mercato verso monopoli o oligopoli.

In un caso economico in cui i prodotti sono essenzialmente materie prime con poco valore aggiunto incorporato, i rendimenti sono generalmente costanti o decrescenti al crescere della produzione. Nel caso invece, come quello del software, in cui i prodotti sono essenzialmente valore aggiunto congelato, si ottengono rendimenti crescenti al crescere della produzione. Nel primo caso nel mercato possono convivere molte imprese, perché la maggiore dimensione non é di per se un vantaggio. Nel secondo caso la convenienza a produrre aumenta con la scala della produzione, favorendo quindi la nascita di monopoli. Tutti i vendor di software spingono per vendere pacchetti software, come dimostra il fatto che per le prime 100 aziende statunitensi di software il margine operativo lordo derivante dalla vendita delle licenze é pari all'87%, contro

³ Clayton Christensen, *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*, HBS Press Book, 1997, pp. 29-61

un 54% dei servizi⁴. Basta in tal caso citare il caso di Microsoft nel mercato dei sistemi operativi.

Nel caso di scarsa concorrenza di imprese la maggiore efficienza di produzione di tramuta non in prezzi più bassi per i consumatori ma in accumulo di extraprofiti per le imprese monopolisti. Nel caso del mercato dei software “chiusi” le ragioni che spingono verso monopoli naturali sono:

- Effetti di network: molti prodotti hightech sono utili solo se compatibili con quelli usati da altri utenti. Più tali prodotti si diffondono e maggiore è la possibilità che emergano come standard.
- Effetti di apprendimento da parte dei consumatori. I prodotti software sono in genere difficili da usare ed imparare a farlo richiede tempo e risorse per i corsi di training. Una volta che un consumatore abbia compiuto questo investimento nell'apprendere un prodotto software, sarà per lui conveniente usarne le versioni successive, invece che passare a un concorrente, anche se migliore. Questo spiega ad esempio il comportamento dei grandi produttori di software che non contrastano in maniera eccessiva la proliferazione di copie illegali (copie “pirata”) dei loro prodotti.

Vista la natura dell'informazione e della conoscenza i monopoli sono quindi destinati a crescere, spinti dal fatto che le economie di scala in molte categorie di beni si applicano sia dal lato dell'offerta che quello della domanda, attraverso le esternalità della rete.

Non ultima, un'altra ragione che porta al fallimento del mercato del software è l'asimmetria informativa tra le informazioni che sono note all'acquirente e quelle invece note al venditore. Quando queste informazioni sono fondamentali per stabilire il prezzo di un prodotto, si ha fallimento di mercato quando l'incontro tra domanda ed offerta non è in grado di determinare il prezzo giusto. È fin troppo facile per un venditore di software mascherare al futuro acquirente difetti ed eventuali malfunzionamenti del proprio prodotto, in particolare considerando il grande dinamismo nel lancio e rinnovo dei prodotti del mercato del software che rende difficile comparare in maniera obiettiva tra di loro i vari prodotti.

L'asimmetria informativa coinvolge anche il produttore di software, in quando anche lui non è in grado di controllare in maniera efficace il comportamento degli acquirenti del software. Essendo il software un bene assimilabile a quello della conoscenza, quindi intrinsecamente modellabile e trasferibile, è possibile che l'acquirente, al di là dei casi di copia illegale,

⁴ La fonte è McKinsey

modifichi, anche solo leggermente, il software acquistato e che lo rivenda a terzi, magari sfruttando delle clausole poco chiare della licenza d'uso.

3.1.1 I costi di sviluppo del software (TCO)

Cerchiamo adesso di affrontare meglio la tematica relativa allo studio dei costi di produzione del software. Sviluppare software è complesso perché richiede personale qualificato con diverse professionalità, fasi di sviluppo che si alternano a fasi di test, nonché la creazione di un gruppo che segua e supporti il prodotto una volta che si è incominciato a vendere.

Anche chi acquista un prodotto software spesso deve effettuare un esborso non indifferente. Vediamo adesso di capire quanto vale (o quanto costa) esattamente un prodotto software. Il software come le altre classi di strumenti o beni strumentali, hanno due tipi distinti di valore economico:

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

Molto spesso si presuppone che il software condivida le caratteristiche di valore di un tipico bene manifatturiero, quindi il tempo impiegato dagli sviluppatori è retribuito dal valore di vendita, che a sua volta è proporzionale ai costi di sviluppo e dal valore d'uso. In realtà se si guardano le statistiche, il software come prodotto per la vendita è una piccola parte del software scritto nel mondo. Si stima che la quota sia inferiore al 10%. Nella stragrande maggioranza dei casi, il software viene scritto per adattare o mantenere software già esistenti. Anche la teoria del valore di vendita nel mercato del software perde di significato. Pensate che cosa succede quando un software va fuori di produzione. Il suo valore diventa nullo.

Analizziamo più dettagliatamente il costo di acquisto, che non si riduce al solo costo di licenza. Definiamo come Total Cost of Ownership⁵ la “*somma di tutte le spese ed i costi associati all'acquisto ed all'uso di equipaggiamenti, materiali e servizi*”. Il TCO è uno strumento che permette di includere i costi sulla base della teoria contabile basata sull'Activity Based Costing (ABC), che è un sistema contabile basato sui costi che attribuisce loro appropriati driver in base

⁵ Total Cost of Ownership, *North Carolina State University, Supply Chain Management Resource Center*, materiale a cura di Ron Handfield <http://scm.ncsu.edu/scm/TotalCosofOwnership.html>

alla loro componente di valore aggiunto. Esso pone come al centro del calcolo dei costi le attività e non i prodotti. Nel caso di applicazione di questo modello al caso del software, otteniamo che per ottenere il TCO bisogna tenere conto delle seguenti attività:

- La fase di pianificazione e di valutazione della soluzione software più appropriata e conveniente per la propria azienda. Questa fase comporta costi connessi all'utilizzo di risorse interne, e visto che spesso sono richieste competenze specifiche, di consulenti esterni all'azienda stessa
- La valutazione degli impatti sugli altri sistemi. Spesso installare del nuovo software richiede ritocchi più o meno significativi ai sistemi esistenti, come ad esempio l'acquisto di ulteriore hardware, di nuove infrastrutture informatiche, oppure modifiche al software già esistente
- La valutazione degli impatti sull'efficienza operativa. L'acquisto di un nuovo sistema software può richiedere costi inizialmente non pianificati, ma che di cui si deve tener conto. Basti pensare ai problemi che possono causare la presenza di errori nel software o di dati non compatibili con il formato usato da altri programmi, o ancora una interfaccia utente poco intuita o eccessivamente complessa.
- Il training per gli utenti presenti in azienda. L'introduzione di nuovo software in una azienda comporta anche la spesa necessaria per effettuare training o l'insegnamento ex-novo delle conoscenze necessarie ai dipendenti dell'azienda.
- La valutazione dei costi dei servizi accessori, che comprendono la manutenzione evolutiva del programma, gli aggiornamenti del software, il servizio di supporto (in caso di servizi *mission critical*, l'intervento sul sito può essere anche molto costoso). Spesso questi servizi vengono quotati come un canone annuo calcolato come percentuale del software acquistato (solitamente varia dal 10% al 20%).
- La valutazione dei costi incidentali. Possono essere causati dal fattore umano, come ad esempio errori dovuti ad una svista che causano perdita di dati. Ma possono essere anche dovuti a problemi intrinseci del software: se ad esempio esso viene configurato male, potrebbe causare problemi di sicurezza con possibile perdita di informazioni sensibili.

3.1.2 Le dinamiche di mercato e l'emergere dell'outsourcing

Per comprendere i benefici che il software Open Source sta portando al mercato dobbiamo prima analizzare la situazione caratterizzata dall'uso di software proprietario.

Il mondo dell'Information Technologies (IT) ha conosciuto negli ultimi tre anni la prima vera crisi strutturale dopo un trend di sviluppo senza interruzioni di durata quasi ventennale. I cali, registrati nel 2002 e a livello di trend nell'anno in corso, sono dovuti in parte all'esaurirsi della prima ondata di investimenti infrastrutturali per le reti a banda larga, in parte al ritardo nell'introduzione di significative novità sul fronte delle telecomunicazioni mobili che, pur crescendo ancora, configurano ormai un mercato sempre più maturo se non addirittura saturo. L'industria del software rimane comunque importante: si stima (fonte IDC anno 2005) che oggi il suo giro di affari mondiale abbia raggiunto una dimensione stimata di 175 miliardi di dollari annui.

Solo le imprese maggiori hanno continuato ad investire in IT, all'insegna dell'integrazione Internet/Intranet con il mercato e al proprio interno. Ma in modo conservativo, sostenendo progetti già avviati o guardando ad un ritorno di brevissimo periodo, e privilegiando il taglio dei costi di gestione. Le PMI e i privati hanno invece preferito rinviare a tempi futuri progetti di innovazione per poter affrontare il periodo di recessione dell'economia mondiale.

Se a livello macro l'IT ha mostrato una contrazione significativa che vedrà segnali di ripresa solo nel 2006 a livello di singoli settori la realtà si presenta articolata e non sono mancati casi che non solo hanno tenuto, ma stanno crescendo spesso con tassi a due cifre. A fronte del calo degli investimenti IT, soprattutto da parte delle PMI e dei privati, ciò che progredisce, ancora, sono soprattutto servizi su rete mobile e gli investimenti in termini di sviluppo rete da parte delle grandi aziende.

Secondo una recente ricerca svolta dalla Bocconi, il 70% delle piccole e medie imprese italiane ha dichiarato l'utilizzo di internet "inutile". Non può più reggere la "strategia del non ci sono così mi distinguo", sostiene Emmanuelli, Presidente di Smau, dove per il non-ci-sono intende non-sono-presente-in-rete. Il web non è solo la navigazione online, ma comporta una serie di soluzioni a problemi gestionali e strategici.

Chi non s'adega resterà fuori dal mercato. Un mercato ormai planetario che è reso sempre più competitivo dall'affacciarsi sull'arena del commercio mondiale del gigante cinese.

A livello di dinamica di mercato, l'outsourcing, che potremmo succintamente tradurre con "esternalizzazione", ovvero l'affidamento a terzi di specifiche funzioni o servizi, sembra accreditabile di una crescita decisamente consistente, se non impetuosa, nei prossimi anni. Non sta cambiando solo il modo di sviluppare il software, ma anche il modo di usarlo. Il software perde si dematerializza e rimane il servizio, spesso erogato, e consumato, direttamente da Internet, quindi non più fisicamente in azienda.

Oggi giorno, nella maggior parte dei casi, consta nell'affidamento a terzi della manutenzione delle reti informatiche e telematiche, della gestione del centro elaborazione dati o degli archivi interni. Risulta però abbastanza diffuso l'outsourcing di funzioni più complesse: il 57% delle strutture pubbliche e private ha affidato a terzi la manutenzione e gestione dei propri immobili, il 44% fa gestire in outsourcing i servizi logistici, il 21% si rivolge a terzi per la gestione e amministrazione del personale (rapporto CENSIS 2003).

Se si ragiona sul fatto che anche il Pentagono, tempio della Difesa statunitense, ha affidato alla società privata KBR-Halliburton la totale gestione dello sviluppo dei piani logistici per il rapido dispiegamento delle Forze Armate USA in 13 diversi Paesi del pianeta⁶, ci si può rendere conto di quanto oggi sia divenuto organizzativamente impensabile ed economicamente insostenibile (per enti pubblici e privati) la gestione "in proprio" di tutta la propria attività.

Ancora più interessante è rilevare che quasi il 60% delle imprese private, ed il 40% delle grandi strutture delle Amministrazioni dello Stato Italiano contattate dal sondaggio del CENSIS, prevedono di intensificare il ricorso all'outsourcing nei prossimi due anni e per contro, solo per una piccola quota del campione (mediamente il 5%) l'uso di questa forma di servizio diminuirà. Dall'indagine di CENSIS-TESS emerge inoltre con chiarezza come le potenzialità di sviluppo della domanda appaiono ampie se si tiene conto che circa il 60% del panel analizzato sembra caratterizzarsi per una contenuta spinta all'esternalizzazione ma, nel contempo, per una forte propensione all'innovazione dell'organizzazione interna. Si tratta pertanto di una potenziale area di mercato in cui non vi è una chiusura rispetto all'outsourcing, per cui spetterà alla capacità delle aziende di proporre servizi innovativi per incentivarne la propensione all'uso.

Uno studio di Booz Allen Hamilton (2003) su 40 gestori di telefonia fissa, mobile ed accesso internet, evidenzia la tendenza a dare in outsourcing le attività meno strategiche , per esempio la manutenzione della rete, la logistica, la produzione. Per ridurre i costi logistici gestionali del 15/20%. Se sarà confermata la tendenza nei prossimi anni si potrà creare un giro di affari di 9 miliardi di Euro.

⁶ La fonte è *Outsourcing war* - Business Week del 15/09/2003

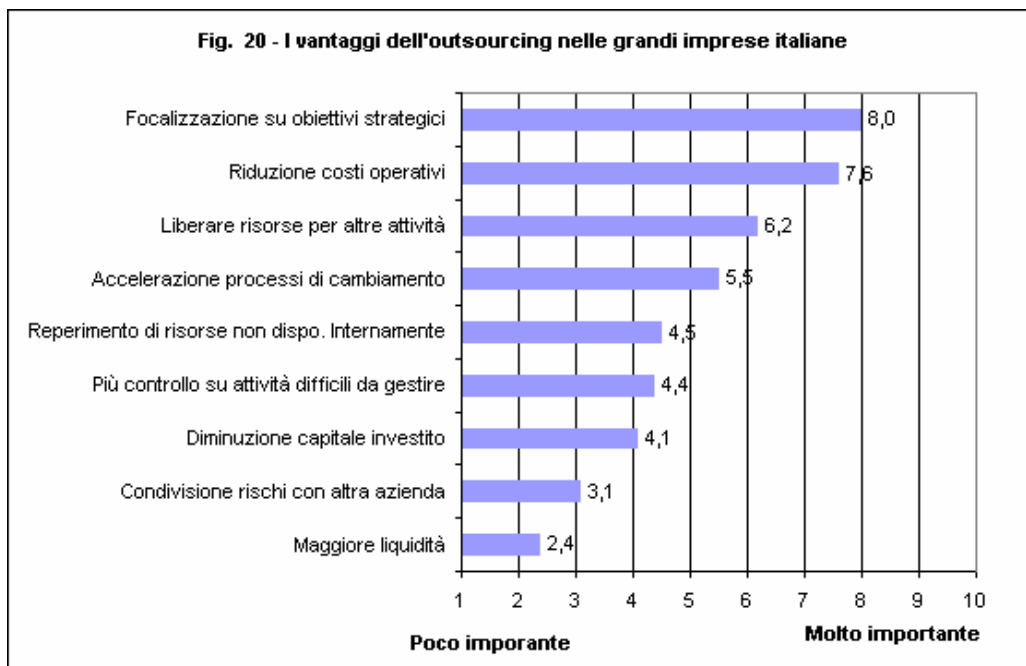


Figura 1 I vantaggi dell'outsourcing nelle grandi imprese italiane

I dati e le statistiche sopra esposti saranno sicuramente indicativi del fenomeno outsourcing, ma, giunti a questo punto, ci si deve necessariamente domandare perché in un determinato segmento di mercato, scevro da preclusioni all'uso di strumenti di gestione innovativi, potrà risultare efficace e produttivo affidare a terzi specifiche funzioni o servizi: ci è d'aiuto l'indagine 2003 del più volte citato CENSIS, da cui si evince chiaramente come i fattori che attraggono maggiormente i potenziali committenti pubblici e privati sono rinvenibili in pochi ma estremamente chiari aspetti:

- l'offerta di servizi in outsourcing, particolarmente tarata sulle specifiche richieste e esigenze dei committenti, permette un elevato livello di personalizzazione e quindi di proficuo sfruttamento;
- dell'efficienza e della rapidità dei servizi in outsourcing rispondono contrattualmente le società cui sono affidati;
- l'outsourcing permette piena modulabilità, ovvero la possibilità di apportare rapidamente e facilmente le modifiche ai singoli servizi che man mano si rendono necessarie;
- la possibilità di usufruire di servizi che comportino non solo l'espletamento di determinate funzioni, ma che permettano anche al committente di monitorare costantemente il corretto svolgimento delle attività esternalizzate;
- la possibilità di usufruire di alcuni servizi attraverso le reti telematiche e informatiche.

OUTSOURCING E LE AZIENDE EUROPEE: Con un tasso annuo di crescita dell'8%, la spesa delle aziende europee in tecnologie aumenterà del 3% in più rispetto alle grigie previsioni di qualche mese fa, quando la crisi economica non lasciava presagire un'inversione di tendenza di tale portata. A trainare il settore ICT (Information and Communication Technology) sarà in particolare il segmento dell'outsourcing, la cui spesa nel 2008 rappresenterà il 43% degli investimenti totali in ICT delle aziende europee.

Il driver della ripresa, come detto, sarà quindi il ricorso a servizi in outsourcing, che nel biennio 2004 - 2005 dominerà la spesa tecnologica del vecchio continente.

“Mentre un numero crescente di grandi aziende si sta orientando all'outsourcing per negoziare contratti vantaggiosi - ha dichiarato Martha Bennett, vice president di Forrester Research - allo stesso tempo sta anche aumentando il ricorso a società esterne, per mettere a punto strategie di outsourcing a 360°”. In particolare, grazie a soluzioni a basso costo, basate sulla formula di pagamento del pay-per-use e dell'affitto calcolato in base all'effettivo utilizzo delle tecnologie e non a pacchetti di utilizzo standard che corrono il rischio di essere utilizzati solo in parte.

Ogni strategia di penetrazione del mercato deve tenere conto, tuttavia, di un aspetto fondamentale al quale i potenziali committenti sembrano particolarmente interessati: la capacità degli operatori dell'outsourcing di ascoltare le esigenze delle imprese o delle Amministrazioni pubbliche che si intendono servire e che si aspettano dai processi di outsourcing non tanto e non solo risultati misurabili in termini di riduzione dei costi di gestione ma, soprattutto, un progressivo accrescimento dell'efficienza interna.

L'outsourcing deve quindi combinare solo efficienza nei costi ma implementare soluzioni complete ed integrate nel business aziendale per consentire all'aziende cliente un traghetamento morbido verso le nuove piattaforme tecnologiche in grado di ridurre il time to market ed erogare tutta una serie di servizi aggiuntivi al fine di fidelizzare ed assistere il cliente nelle fasi pre e post vendita. La sfida sul mercato globale, dominato dall'emergere di nuovi global player come la Cina, può essere vinta dalle realtà imprenditoriali che sono supportate da un sistema-paese capace di tradurre rapidamente in prodotto e servizio le innovazioni tecnologiche in cui il costo sta diventando fattore critico di successo.

3.2 Perché chiudere il codice sorgente?

In questa sezione intendiamo rispondere alla seguente domanda: che cosa si protegge esattamente, quando si chiude l'accesso al codice sorgente?

Una risposta a questa domanda è stata formulata da Raymond⁷. Partiamo da un esempio. Supponiamo di incaricare qualcuno di scrivere un pacchetto software di contabilità, specifico per la vostra impresa. Il problema non sarà più facile da risolvere, a seconda che il codice sorgente sia aperto o chiuso: l'unica motivazione razionale per tenerlo segreto è se si intende vendere il pacchetto ad altre persone o impedirne l'uso da parte della concorrenza. La risposta più ovvia è che stiate proteggendo il valore di vendita, ma questo non si applica al 95% dei software scritti per uso interno.

La seconda possibilità (proteggere il vantaggio competitivo) richiede un esame più attento. Supponiamo, sempre in questo ipotetico esempio di dover scrivere un software di contabilità, che si opti per sviluppare il codice sorgente secondo l'approccio Open Source. Il pacchetto si diffonderà e migliorerà, grazie ai perfezionamenti apportati dalla comunità. Ma anche la concorrenza inizierà a usarlo, ne trarrà beneficio senza pagare costi di sviluppo e minaccerà l'impresa che ha sviluppato il pacchetto software. Ma questo non può considerarsi un argomento contro l'Open Source. Il problema reale è se il beneficio ottenuto dalla ripartizione degli oneri di sviluppo supera le perdite dovute all'aumento della competizione a causa del *free riding*. Molti tendono a fare ragionamenti semplicistici su questo argomento:

- ignorando il vantaggio funzionale derivante dall'impiego di più addetti allo sviluppo
- non considerando i costi di sviluppo come costi sommersi. Tanto per formulare un'ipotesi, dal momento che i costi di sviluppo sarebbero da pagare ugualmente, sarebbe errato calcolarli come costi dell'Open Source (qualora si scelga questa strada).

⁷ Raymond E. S., *The Cathedral and the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly&Associates, 1999, capitolo II, III, IV

Ci sono altre ragioni puramente irrazionali che potrebbero motivare la scelta di sviluppo proprietaria. Si potrebbe pensare, per esempio, che il software commerciale tuteli l'impresa contro i cracker e gli intrusi. Cosa assolutamente non vera, in quanto è meglio non fidarsi della sicurezza dei programmi commerciali. Infatti la sicurezza è un aspetto dell'affidabilità: soltanto gli algoritmi e le applicazioni che siano stati accuratamente controllati e rivisti da più persone potranno essere considerati sicuri.

Abbiamo quindi mostrato come sia labile il sistema di protezione del software attraverso i diritti di licenza, e che comunque si dovrebbe applicare solo per il software destinato alla commercializzazione. Vedremo nelle prossime sezioni come anche il modello Open Source sia sostenibile dal punto di vista economico dalle aziende che commercializzano soluzioni software.

3.3 La catena del valore nello sviluppo del software: lo spostamento del valore dalla produzione all'erogazione dei servizi

Internet ha creato un nuovo modello di impresa, che agisce nella realtà virtuale omogenea ed interconnessa, dove la competizione si svolge sul piano della velocità, della qualità e dell'efficienza in una corsa sfrenata alla soddisfazione dei bisogni e delle necessità anche quelle più velleitarie. Le aziende Open Source rientrano solo in parte nel modello dell'Azienda Virtuale⁸. Utilizzano Internet e si basano sulla stessa risorsa: l'informazione, tuttavia differiscono dal modello di Davidow e Malone per le caratteristiche e sulla stessa natura del prodotto su cui si basano. Il software libero e il loro modello di sviluppo, il progetto Open Source, non obbediscono alle leggi ed ai tempi del mercato.

La struttura dei costi e dei ricavi delle aziende fondate sul software libero o comunque Open Source differisce da quelle tradizionali: la soglia di ingresso è estremamente bassa e basso è il costo di produzione, ma anche il valore del bene è estremamente basso in quanto è facilmente reperibile dai possibili clienti, in quanto disponibile gratuitamente su Internet.

In realtà ciò che viene offerto sul mercato non sono i beni ma i servizi, il valore di vendita è indiretto. Come esempio riportiamo le distribuzioni Linux commerciali, come quella di Red Hat. Red Hat non vende il software ma i servizi ad esso collegati: come il valore aggiunto nell'assemblaggio dei diversi software che compongono le garanzie di funzionamento e compatibilità con altri sistemi, il servizio di assistenza remota, nonché sviluppi per determinati progetti, la formazione e la vendita di consulenza specialistica.

Le grandi aziende tradizionali di sviluppo software hanno tenuto una strategia di marketing tendente ad avere prezzi di acquisto molto alti e contributi per l'assistenza molto bassi o quasi nulli, in quanto l'ufficio assistenza non genera forti ricavi (spesso è considerato un costo puro). Se quindi i profitti provengono dalla vendita, tutti gli sforzi sono concentrati a produrre nuove versioni del programma e mandarle fuori produzione quelle esistenti il prima possibile. Queste aziende alla comparsa dell'Open Source hanno reagito con diverse tattiche. La più grande e nota, la Microsoft, ha preferito rispondere con tattiche FUD (Fear, Uncertainty, Doubt) ovvero esprimendo forti dubbi sull'affidabilità delle soluzioni Open, in modo da alzarne i costi di adozione. Negli ultimi anni Microsoft sta però incominciando a dare i primi segni di apertura rispetto ai metodi Open.

⁸ W.H. Davidow e M.S. Malone, *The Virtual Corporation*, Harper Collins, 1992, pp. 23-32

Altre aziende si sono invece completamente convertite, distribuendo i propri prodotti completi dei sorgenti e con licenze Open Source, come é avvenuto per SUN, che ha rilasciato prima StarOffice, la suite di gestione documentale, sotto licenza GPL, poi il proprio sistema operativo Solaris.

L'approccio Open Source prevede che per far fronte alla reale struttura dei costi del ciclo vitale del software serve una struttura dei prezzi fondata su contratti per determinati servizi. Vi é quindi tramite questi abbonamenti uno scambio di valore continuo tra produttore e consumatore in uno schema dominato dal paradigma servizio/costo. Solo il valore di vendita é minacciato dal passaggio dal software commerciale all'Open Source, contrariamente al valore d'uso, che rimane integro.

Comunque tutte le aziende, anche quelle tradizionali, si stanno muovendo sul concetto del software come servizio esso stesso. La nostra disamina sul TCO (si vede il paragrafo 3.1.1) ha messo in evidenza come il costo della licenza sia solo una componente, seppur importante, dell'intero costo di acquisto. Il business si sta sempre più spostando dalla vendita dei programmi alla vendita dei servizi correlati. Anzi la dinamica delle offerte, come si può vedere negli Stati Uniti va verso pacchetti on inclusive, che, in cambio di un canone fisso mensile, permettono all'utente di usufruire delle funzionalità del software, in outsourcing o includendo l'affitto dell'HW necessario, senza però acquistare il software e pagare le licenze. Spesso anche il supporto e la configurazione dei sistemi sono inclusi. Il margine di profitto, tolti i costi di sviluppo, vengono distribuiti sul ricavo derivante dai servizi, che in genere sono forniti durante un arco temporale pluriennale.

In questo modo le imprese acquirenti, in cambio di un corrispettivo fisso mensile possono usufruire del servizio senza effettuare investimenti importanti e correre il rischio di imbarcarsi in complicate integrazioni di software. In questo modo anche i rischi che il software diventi obsoleto si abbassano notevolmente.

3.4 I modelli di Business dell'Open Source

Dal passaggio dal software commerciale all'Open Source, è solo il valore di vendita del software ad essere impattato, teoricamente azzerato, mentre invece il valore d'uso viene completamente preservato. Se quindi, come abbiamo precedentemente illustrato, lo sviluppo con il modello Open Source è veramente più efficace e permette di ottenere un prodotto di qualità più elevata rispetto a quello commerciale, allora diventa plausibile che il solo valore d'uso sia sufficiente per finanziare lo sviluppo dell'Open Source in modo sostenibile.

E, in effetti, non è difficile individuare almeno due study case che hanno dimostrato la capacità del software Open Source di sostenersi dal punto di vista economico. La storia del web server *Apache* generalizza un modello in cui gli utenti di software trovano vantaggioso sostenere lo sviluppo dell'Open Source perché, così facendo, ottengono un prodotto migliore di come potrebbe essere altrimenti, a un costo inferiore.

L'Open Source serve non tanto ad abbassare i costi, quanto a ripartire il rischio. Tutte le parti in gioco si accorgono che la disponibilità del codice sorgente, così come la presenza di una comunità che agisce in collaborazione, finanziata da più fonti di reddito indipendenti, fornisce una garanzia di sopravvivenza che ha un valore economico intrinseco o, comunque, un valore sufficiente per finanziarla.

L'Open Source rende piuttosto difficile trarre un valore di vendita diretto dai software. Non si tratta di una difficoltà tecnica: il codice sorgente non è né più né meno copiabile del binario e l'istituzione del copyright e di diritti di licenza che permettano la riscossione del valore di vendita, non sarebbe necessariamente più difficile per i prodotti Open che per quelli commerciali.

La difficoltà sta piuttosto nella natura del contratto sociale che si trova alla base dello sviluppo Open. Per tre motivi che si rinforzano a vicenda, la principale licenza Open Source (la GPL) proibisce la maggior parte delle forme di restrizione d'uso, redistribuzione e modifica che faciliterebbero la riscossione di profitti derivanti dal valore di vendita. Per comprendere questi motivi, occorre esaminare il contesto sociale nel quale si sono evolute le licenze: la cultura hacker di Internet. Se è vero che una minoranza di hacker resta ancora ostile alla logica del profitto, la volontà generalizzata da parte della comunità di cooperare con aziende i cui prodotti si basano su Linux, come aziende come Red Hat, SUSE dimostra che la maggioranza degli hacker sarebbe felice di lavorare a fianco del mondo dell'impresa, quando è nei loro interessi. Le ragioni reali per cui gli hacker guardano con scetticismo alle licenze che consentono guadagni diretti, sono assai più sottili e interessanti.

Una delle ragioni riguarda la simmetria. Mentre la maggior parte degli sviluppatori Open Source non oppone obiezioni sostanziali al fatto che altri traggano profitto dai loro doni, i più chiedono che nessuno (ad eccezione, eventualmente, del creatore di una parte del codice) si trovi in una posizione privilegiata per fare profitti. Il signor Rossi Hacker è d'accordo che la Bianchi tragga profitti dalla vendita del suo software o delle sue patches, ma solo nel caso che anche lui stesso, almeno in teoria, possa farlo.

Un'altra ragione riguarda le conseguenze indesiderate. Gli hacker hanno osservato che le licenze che comportano la restrizione e il pagamento di contributi per uso "commerciale" o vendita (il tentativo più comune e, a prima vista, nemmeno tanto assurdo, di recuperare il valore di vendita diretto) hanno effetti davvero inquietanti. Un caso specifico è la possibilità di controllare segretamente la legalità di attività come la ridistribuzione di antologie economiche su CD-ROM, che in linea di massima sarebbero da incoraggiare. Più in generale, le restrizioni sull'uso, la vendita, le modifiche, la distribuzione (e altre clausole presenti sulle licenze) comportano spese aggiuntive per la conformità della distribuzione e (con l'aumento dei pacchetti in circolazione e in utilizzo) una combinazione esplosiva di incertezza soggettiva e potenziali rischi legali. Poiché questi risultati si considerano dannosi, si registra una forte pressione sociale per mantenere le licenze semplici ed esenti da restrizioni.

L'ultima e più importante ragione riguarda la dinamica di revisione e di cultura del dono descritta nel libro di Raymond. Le restrizioni sulle licenze, designate per proteggere la proprietà intellettuale o per percepire un valore di vendita diretto, hanno spesso l'effetto di rendere legalmente impossibile la biforcazione dei progetti (questo avviene, per esempio, per le licenze di SUN, le cosiddette "Community Source", per Jini e Java di SUN). Se la biforcazione incontra molto scetticismo ed è considerata come l'ultima risorsa, la presenza di quest'ultima risorsa si considera di capitale importanza in caso di incompetenza o defezione del gestore (a causa, per esempio, di una licenza troppo restrittiva).

La comunità hacker mostra una certa elasticità in materia di simmetria: per questo, tollera licenze, come NPL di Netscape, che concedono alcuni privilegi in materia di profitti agli autori originali del codice (nel caso specifico di NPL, il diritto esclusivo di utilizzare il codice Open Source di Mozilla in prodotti derivati, anche commerciali). Ma ha meno elasticità nei confronti delle conseguenze indesiderate e nessuna sul mantenimento della possibilità di biforcazione (ed è per questo che gli schemi elaborati dalla Sun per la "Community License" di Java e Jini sono stati in gran parte rifiutati dalla comunità).

Queste ragioni spiegano le clausole della Open Source Definition, scritta per esprimere il parere congiunto della comunità hacker sui requisiti essenziali delle licenze standard (GPL, licenza BSD, licenza MIT e Licenza Artistica). Queste clausole hanno l'effetto (ma non l'intenzione) di rendere molto difficile l'ottenimento di un valore di vendita.

Ciononostante, esistono modi per costruire mercati nei servizi legati al software che comportino un valore di vendita indiretto. Sono cinque i modelli conosciuti, come presentati dal libro di Raymond e due quelli teorici (ma se ne potranno sviluppare altri in futuro). In figura 22 li abbiamo presentati in maniera grafica, per poi spiegarli dettagliamene in seguito.

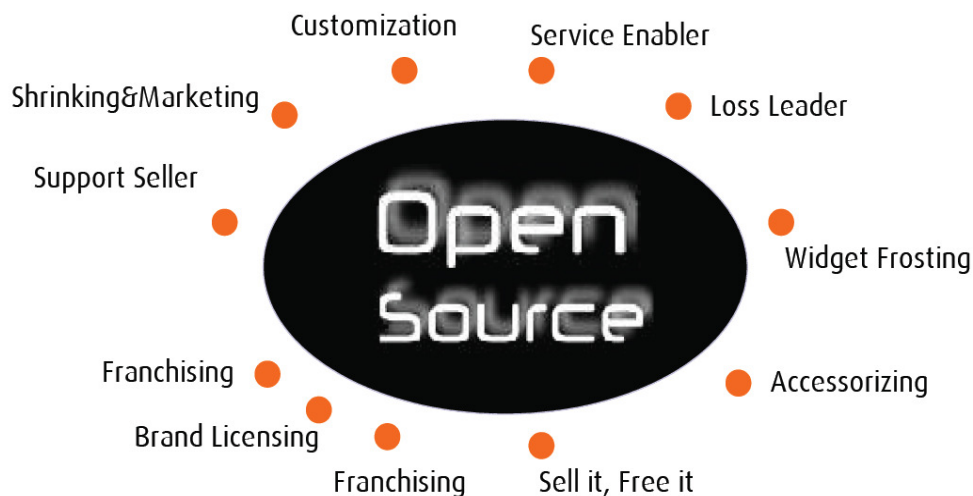


Figura 2 Modelli di Business per il Software Open Source

Vendita di servizi

Si tratta del modello più semplice, che abbiamo già precedentemente trattato. I venditori di servizi software (chiamati anche integratori) sono aziende che vendono sia il software che i servizi di supporto al cliente: configurazione ed installazione, mantenimento evolutivo, supporto postvendita e garanzia. Nel commercio tradizionale del software chiuso, il reddito che deriva da vendite del software è pari al 60% del totale, dal supporto arriva il rimanente 40%. Un modello Open Source sposta drammaticamente il ricavo dalle vendite di licenze software al 0% mentre il supporto sale al 100%. Tuttavia, il software libero attrarrà più utenti e quindi più reddito derivante dai servizi di supporto. Poiché il software è Open Source, l'azienda trarrà beneficio dalle esternalità positive che derivano dal bug fixing e dalle nuove funzionalità implementate

dagli utenti. Quindi in questo modello, si rende disponibile il codice sorgente di un software per creare una nicchia di mercato non nel settore del software commerciale, bensì nei servizi. Come esempio possiamo citare Red Hat e altri distributori di Linux. Ciò che vendono, in realtà, non sono i software, ossia i bit, ma il valore aggiunto nell'assemblaggio e nelle prove di un sistema operativo funzionante e con garanzie (anche solo implicite) di commerciabilità e compatibilità con altri sistemi operativi della stessa marca. Altri elementi del valore da essi offerto comprendono l'assistenza gratuita per l'installazione e l'offerta di opzioni per contratti di assistenza continuativi.

Come ulteriore esempio riportiamo lo study case di Zope, un programma per la creazione di siti web, sviluppato dalla Digital Creations. La Digital Creations, autrice di questo programma molto valido tecnicamente, contattò un esperto di venture capital per raccogliere fondi e poter crescere. L'esperto dopo una attenta analisi consigliò di pubblicare come Open Source tutta la suite. Stando agli standard tradizionali dell'industria del software, questa sembrerebbe una mossa completamente folle. La saggezza convenzionale, acquisita alla scuola di economia, ci insegna che una proprietà intellettuale vitale come Zope rappresenta la punta di diamante di un'impresa e che non deve essere venduta in alcun caso. Ma l'esperto aveva due argomenti a suo sostegno: il primo, che il vero patrimonio di Zope, in realtà, erano i cervelli e le abilità delle persone in esso impegnate; il secondo, che probabilmente, Zope avrebbe prodotto più valore nel costruire un mercato che come strumento segreto. Per capire tutto questo, paragonate le due situazioni. In quella convenzionale, Zope rimane l'arma segreta della Digital Creations. Ammettiamo pure che sia molto efficace. Il risultato è che l'impresa sarà in grado di ottenere una qualità superiore in tempi brevi, ma nessuno lo saprà. Sarà facile soddisfare i clienti, ma più difficile sarà iniziare con l'accattivarsi una clientela di base. L'esperto, invece, capì che uno Zope open source avrebbe rappresentato una pubblicità importantissima per il vero patrimonio della Digital Creations: le persone. Ipotizzò che i clienti, valutando le prestazioni di Zope, avrebbero considerato più efficace contattare gli esperti, piuttosto che sviluppare una gestione interna di Zope. Da allora, uno dei rappresentanti di Zope conferma pubblicamente che la strategia open source ha "aperto molte porte che non si sarebbero trovate altrimenti". I potenziali clienti non mancano di rispondere alla logica sottesa a questa situazione e, come previsto, Digital Creations è in auge.

Articoli civetta

In questo modello, si pubblica gratuitamente una versione software Open Source per creare pubblicità e magari un brand o difendere una posizione sul mercato, mentre invece i profitti

vengono ottenuti attraverso una versione proprietaria del software. Nella variante più comune, la versione Open Source ha capacità o modi d'uso ridotti rispetto alla versione commerciale. Oppure la versione Open Source realizza solo la funzione utente del software, lasciando alla versione commerciale le ben più lucrose attività sul server centrale. Come esempio possiamo citare il caso di Netscape/Mozilla, in cui un software client "libero" agevola le vendite di un software server, e il caso di un software di comunicazione di Voce su Internet (VoIP), realizzato in software da una ditta chiamata XTEN⁹, da installare sul proprio PC, Open Source, ma senza la possibilità di instaurare videoconferenze o fruire di un servizio di Instant Messaging, servizi disponibili solamente nella versione commerciale del software. XTEN commercializza inoltre centralini VoIP e sistemi di videoconferenza, ovviamente venuti attraverso licenze commerciali. Il successo della versione Open Source ha permesso ad XTEN significativi vantaggi commerciali, azzerando di fatto le spese di marketing per la pubblicità. A volte il profitto consiste solamente da abbonamento/pubblicità in relazione al portale Web da cui si scarica il software.

Perdita della leadership

La decisione di una azienda di rilasciare il codice sorgente in modalità Open Source può dipendere anche da scelte strategiche di marketing. Come esempio citiamo la Netscape Communications che all'inizio del 1998, rese disponibile il codice sorgente del browser Mozilla. La decisione fu dovuta al calo del giro d'affari legato al browser che ammontava al 13% dei guadagni, in seguito al lancio del browser¹⁰ Internet Explorer da parte della Microsoft, che lo vendeva (e tuttora lo vende) in maniera congiunta con il sistema operativo e software. Netscape temeva che la Microsoft intendesse monopolizzare il mercato dei browser, per poi sfruttare il proprio controllo de facto sull'HTML, per eliminare Netscape dal mercato server. In effetti, rendendo disponibile il codice sorgente del browser Netscape, ancora assai diffuso, Netscape ha negato alla Microsoft la possibilità di istituire un monopolio del browser, accelerando lo sviluppo di Netscape attraverso lo spirito di collaborazione dell'open source. Ha inoltre costretto che Microsoft Internet Explorer si sarebbe trovato costretto a tenere il passo e dunque impossibilitato a definire l'HTML in modo esclusivo.

⁹ Il sito e' www.xten.com

¹⁰ Un browser e' il programma che permette di navigare e visualizzare i contenuti dei siti Internet.

"Widget frosting"

Questo modello riguarda i produttori di hardware¹¹. Le pressioni del mercato hanno costretto le società che trattano hardware a scrivere e gestire software (dai device driver agli strumenti di configurazione, fino a interi sistemi operativi), ma il software stesso non è fonte di profitti, bensì una spesa aggiuntiva, spesso consistente.

In questa situazione, la disponibilità del codice sorgente non dà molto da pensare all'azienda produttrice. Non ci sono guadagni da perdere, né aspetti negativi. Ciò che il produttore acquista è una risorsa di sviluppo di gran lunga maggiore, una risposta più rapida e flessibile alle esigenze dei clienti e maggiore affidabilità, grazie alla possibilità di revisione reciproca. Inoltre, apre la strada ad altri ambienti gratuitamente e, probabilmente, accresce anche la fedeltà dei clienti, in quanto il personale tecnico dedica più tempo al codice per esaudire le richieste di personalizzazione della clientela.

Ma i produttori, solitamente, sollevano alcune obiezioni specifiche nei confronti della disponibilità del codice sorgente per i driver hardware. Il margine di sicurezza dell'Open Source è particolarmente evidente nel campo del debugging di elementi di interfaccia ("widget frosting"). I prodotti hardware hanno produzione e tempi di assistenza limitati, scaduti i quali, i clienti si ritrovano da soli. Ma se hanno accesso al codice di altri driver, con la possibilità di utilizzarli per produrre patches secondo il bisogno, è più probabile che rimangano clienti della stessa società.

Un esempio assai illuminante dell'adozione del "widget frosting" ci è fornito dalla decisione da parte di Apple Computer, a metà marzo del 1999, di rendere disponibile il codice sorgente di "Darwin", il fulcro del sistema operativo MacOSX server.

Vendere accessori

In questo modello, si vendono accessori per il software OpenSource. Allo scalino più basso, tazze da tè e magliette, magari con l'effigie del pinguino di LINUX; a quello più alto, documentazione redatta e prodotta a livello professionale.

¹¹ Hardware, in questo contesto, comprende tutto a cominciare da schede Ethernet e simili fino a interi sistemi

La O'Reilly Associates, editrice di molti manuali eccellenti sul software open source, è un ottimo esempio di impresa fornitrice di accessori. Per la precisione, O'Reilly impiega famosi hacker operanti nell'open source (tra cui Larry Wall e Brian Behlendorf), come metodo per costruirsi una reputazione nel mercato di sua scelta.

Come Altro esempio, RedHat vende, per una cifra tra i 50 e gli 80 dollari, un CD con contenete l'installazione ufficiale della distribuzione Linux di Redhat ed include un manuale e il supporto tecnico degli esperti di RedHat per 90 giorni. Tuttavia, l'intero sistema, compreso il manuale, può anche essere scaricato dal loro Website gratuitamente, o persino essere comprato assieme ad una rivista per pochi dollari. Redhat realmente non vende i programmi: vende "il buon fattore di tatto" di software già imballato con tanto di manualistica e pacchetto di supporto postvendita. RedHAt sta andando bene finora. Ed é entrata in affari con le grandi industrie come l'IBM e Dell.

Liberare il software, vendere il marchio

Questo modello rappresenta una strategia aziendale in prospettiva. Si procede all'open sourcing di una tecnologia software, mantenendo una sequenza di prove o insieme di criteri di compatibilità, per poi vendere agli utenti un marchio che certifichi la compatibilità dello strumento tecnologico in loro possesso con tutti gli altri della stessa marca.

Liberare il software, vendere il contenuto

Questo modello, ancora una volta, rappresenta una strategia aziendale in prospettiva. Immaginiamo una specie di servizio di borsa telematica in abbonamento. Il valore non è da ricercare né nel software client, né nel server, bensì nell'oggettiva affidabilità delle informazioni fornite. Quindi, si rende disponibile il codice sorgente di tutti i software e si vendono abbonamenti per accedere ai contenuti. Via via che gli hacker aprono il client a nuove prestazioni e lo migliorano in vari modi, il mercato si espande automaticamente.

3.5 I vantaggi dell'adozione del software Open Source

Ricapitolando possiamo affermare che l'approccio commerciale consente di raccogliere profitti dai propri bit segreti; d'altra parte, chiude le possibilità di un'autentica revisione reciproca e autonoma. L'approccio Open Source pone le condizioni per la revisione autonoma da parte di più individui, ma non fa riscuotere alcun profitto per i bit segreti. Il vantaggio di tenere segreti i bit si comprende facilmente: per tradizione, le strategie aziendali centrate sul software sono state costruite intorno a esso. Fino a poco tempo fa, invece, il vantaggio della revisione autonoma non era di facile comprensione.

Se si considera che le aziende che usano software libero o comunque Open Source devono supportare costi di sviluppo nulli si capisce come questo sia un vantaggio competitivo rispetto alle imprese tradizionali. Vedremo più avanti anche i limiti del modello Open Source. Quello che possiamo dire è che si tratta di un nuovo modo di sviluppare, distribuire e licenziare il software. Alle imprese che comprendono i fattori economici, culturali e volendo organizzativi che risiedono in una effettiva strategia Open Source, questo modello offre buone aspettative. È infatti significativo che le grosse imprese hanno incominciato, già da alcuni anni, e con trend crescente, ad usare software libero ed Open Source come base per i propri prodotti commerciali, ed in molti casi hanno contribuito allo sviluppo di questo software attraverso donazioni in denaro alla Free Software Foundation.

Dal lato delle aziende produttrici di software possiamo elencare i seguenti motivi che possono spingere all'adozione del modello Open Source:

- Evoluzione del prodotto. Il modello Open Source è molto adatto, per la matrice dei costi, per quelle aziende che hanno bisogno di creare nuovi prodotti o di arricchire di funzionalità quelli esistenti.
- Aumento della qualità e della affidabilità del prodotto Open Source. Abbiamo visto come lo sviluppo in base al modello Open Source permette di correggere gli errori software in maniera più veloce ed approfondita. In un mercato competitivo, perciò, una clientela alla ricerca di alta fedeltà e qualità premierà i produttori di software che scelgono l'Open Source scoprendo come mantenere un flusso di profitti basato sui servizi, sul valore aggiunto e su altri mercati complementari a quello del software.

- Mantenimento del prodotto. La presenza di una comunità rende più semplice la ricerca di personale qualificato a prezzi convenienti. Inoltre offre maggiore sicurezza. Se il codice sorgente è disponibile, il cliente ha una via d'uscita in caso di fallimento del produttore. Ciò si rivela particolarmente importante per il debugging degli elementi di interfaccia, poiché l'hardware tende ad avere cicli di vita brevi, ma l'effetto è più generico e si traduce in un maggior valore del software Open Source.
- Reclutamento e fidelizzazione dei collaboratori. Il modo di lavorare Open Source da molte più soddisfazioni alle persone che vi lavorano, in quanto potendo disporre del sorgente e di una comunità a cui porre domande, è possibile poter crescere professionalmente maggiormente.
- Facilità a creare e proporre standard liberi e costruire mercati intorno ad essi. La fortissima crescita di Internet deve molto al fatto che nessuno possiede il protocollo TCP/IP su cui si base; nessuno, cioè, è in possesso di un "lucchetto" con cui controllare i protocolli di base di Internet. Inoltre è possibile creare un "effetto rete" che permette di creare fiducia e simmetria: le potenziali parti coinvolte in un'infrastruttura comune hanno ragione di fidarsi maggiormente, se possono vedere come funziona in tutte le sue applicazioni e preferiranno un'infrastruttura in cui tutte le parti abbiano pari diritti a una in cui una sola parte goda di una posizione privilegiata da cui trarre profitti ed esercitare il proprio controllo, evitando situazioni di monopolio.

Quando i profitti ottenuti dai bit segreti sono maggiori del rendimento da Open Source, è ragionevole, in termini economici, scegliere la soluzione commerciale. Quando invece è il rendimento da Open Source a superare i proventi dei bit segreti, è ragionevole, in termini economici, scegliere l'Open Source. In sé, questa è un'osservazione banale. Ma non è più banale, se teniamo conto che il rendimento dell'Open Source è più difficile da calcolare e da prevedere rispetto ai profitti derivanti dai bit segreti: inoltre, tale rendimento è ampiamente sottovalutato, più spesso che sopravvalutato. Anzi, prima che il mondo del business iniziasse a ripensare le proprie premesse in seguito alla diffusione del codice sorgente di Mozilla, all'inizio del 1998, i vantaggi dell'Open Source erano considerati, a torto ma assai diffusamente, equivalenti a zero.

Come valutare, dunque, i vantaggi dell'Open Source? È una domanda difficile, in generale, ma la si può affrontare come qualsiasi altro problema di natura predittiva. Si può iniziare da alcuni casi in cui l'approccio open source abbia trionfato o fallito. Si può poi cercare di generalizzare, pervenendo a un modello che dia almeno un'idea qualitativa dei contesti in cui l'open source sia la

formula vincente per l'investitore o per l'impresa che cerchi di massimizzare i rendimenti. Infine, si può tornare ai dati e tentare di raffinare il modello.

Dall'analisi presentata nel libro di Raymond, ci si potrebbe aspettare che l'Open Source abbia un alto rendimento quando (a) affidabilità, stabilità e misurabilità siano centrali e (b) la correttezza del design e dell'implementazione non siano efficacemente verificabili, se non attraverso la revisione reciproca e autonoma¹².

Il desiderio razionale da parte del consumatore di non ritrovarsi chiuso nel circuito monopolistico di un fornitore farà aumentare il suo interesse nei confronti dell'Open Source (e, di conseguenza, il valore sul mercato competitivo della scelta Open Source da parte dei fornitori) via via che il ruolo del software diventa centrale per il consumatore stesso. Perciò, un terzo criterio (c) spinge verso l'Open Source qualora il software sia un bene capitale di vitale importanza per l'impresa (come avviene, per esempio, nei dipartimenti aziendali di gestione informatica).

Per quanto riguarda l'area di applicazione, abbiamo osservato sopra come un'infrastruttura Open Source crei fiducia e simmetria che, nel tempo, tenderanno ad attrarre nuova clientela e a vincere nella competizione contro il software commerciale; e, spesso, è meglio avere una quota più piccola di questo mercato in rapida espansione piuttosto che una quota più grande di un mercato esclusivamente commerciale e in stagnazione. Di conseguenza, per l'infrastruttura software, una strategia Open che miri all'ubiquità ha più probabilità di rivelarsi remunerativa nel lungo periodo rispetto a una strategia commerciale che miri all'ottenimento dei diritti di proprietà intellettuale.

In effetti, la capacità da parte dei potenziali clienti di ragionare sulle future conseguenze delle strategie del produttore e la loro riluttanza ad accettare il monopolio di un fornitore implica una limitazione più stringente: senza godere ancora di un indiscusso potere sul mercato, si può scegliere o una strategia di ubiquità open source o una strategia che punti a un guadagno diretto derivante dal software commerciale, ma non entrambe¹³. L'esempio può anche essere presentato in termini meno negativi: dove domina l'effetto rete, è probabile che l'Open Source sia la formula vincente.

¹² Questo secondo criterio si ritrova in pratica nella maggior parte dei programmi non banali

¹³ Esempi analoghi di questo principio sono riscontrabili altrove, per esempio sui mercati elettronici, in cui, spesso, i clienti rifiutano di acquistare profili costituiti dal solo codice sorgente

Potremmo riassumere questa logica osservando che l'Open Source sembra raggiungere il massimo dell'efficacia nell'aumentare i profitti rispetto al software commerciale, per software che (d) istituiscano o abilitino una comune infrastruttura di calcolo e comunicazioni.

Infine, si può evidenziare il fatto che i fornitori di servizi unici o anche solo altamente differenziati hanno più ragioni di temere la copia dei loro metodi da parte della concorrenza, rispetto ai produttori di servizi i cui algoritmi centrali e conoscenze di base siano largamente noti. Di conseguenza, l'Open Source ha più possibilità di prevalere quando (e) i metodi di punta (o equivalenti funzionali) fanno parte delle comuni conoscenze ingegneristiche.

Il core software di Internet, Apache e l'implementazione Linux dell'Application Program Interface (API) Unix con standard ANSI rappresentano esempi canonici di tutti e cinque i criteri. Il cammino verso l'Open Source nell'evoluzione di tali mercati è bene illustrato dal ritorno di convergenza dei dati su reti TCP/IP a metà degli anni Novanta, dopo quindici anni di tentativi falliti di costruire imperi con protocolli commerciali come DECNET, XNS, IPX e simili.

D'altra parte, la scelta dell'Open Source è assai meno sensata per le imprese che abbiano in loro possesso esclusivo una tecnologia software generante valore (cioè che soddisfi strettamente il criterio (e)) che sia (a) relativamente insensibile ai guasti; (b) facile da verificare con mezzi diversi dalla revisione reciproca e autonoma; e che non sia (c) centrale per l'impresa. Inoltre, il suo valore non deve poter aumentare in modo significativo (d) per l'effetto rete o per l'ubiquità.

Riassumendo, le seguenti discriminanti fanno propendere per l'open source:

- (a) affidabilità/ stabilità/ misurabilità sono fondamentali;
- (b) la correttezza del design e dell'implementazione non possono essere verificate efficacemente se non tramite la revisione reciproca e indipendente;
- (c) il software è di vitale importanza per il controllo dell'impresa da parte dell'utente;
- (d) il software istituisce o abilita una comune infrastruttura di calcolo e comunicazioni;
- (e) i metodi di punta (o loro equivalenti funzionali) fanno parte delle comuni conoscenze ingegneristiche.

3.5.1 Il concetto di capitale sociale del Software Open Source

Il concetto di Capitale Sociale rappresenta una sistemazione teorica elaborata da Coleman [41] a partire dal concetto di “Capitale umano” elaborato negli anni sessanta da Becker. Il Capitale umano é costituito da un insieme di risorse per l’azione, rappresentate dalle abilità e dalle conoscenze acquisite dall’individuo. Quando due o più persone stabiliscono una relazione tra di loro, l’insieme delle loro risorse entra nella struttura del loro rapporto. Di conseguenza il Capitale Sociale é quindi quel potenziale di risorse, utile alle proprie strategie d’azione, a disposizione dei singoli partecipanti al network di relazioni in cui sono inseriti. Esso é fortemente connesso alla struttura della relazione.

Se applichiamo al mondo Open Source gli strumenti di indagine metodologica elaborati dalla “Teoria della società” di Coleman, in particolare le forme efficienti di capitale sociale osservabili nei network, possiamo trarne delle utili osservazioni.

Lo scambio di software, ma anche di consigli, strumenti di sviluppo e di conoscenza in senso lato all’interno della comunità Open Source, segue, come abbiamo visto trattando la storia del movimento hacker, la cultura del dono, basata su un debito di riconoscenza con aspettativa di restituzione, in inglese *credit slip*, in sostanza un credito esigibile non per diritto, ma per affidabilità o riconoscenza del debitore.

Un altro aspetto importante riguarda il “Potenziale informativo”, che é proprio di tutte le relazioni sociali; la sua importanza é data dal fatto che, all’aumentare dell’informazione, si hanno più elementi a supporto delle proprie azioni. Nelle comunità Open Source l’informazione non solo viaggia in tempo reale e in forma gratuita, grazie ad Internet, ma anche la qualità dell’informazione che viene scambiata é molto spesso molto elevata.

Coleman introduce il concetto di “Regole e sanzioni efficaci”. Una regola che costituisce un’importante forma di capitale sociale all’interno di una comunità é quella che impone di rinunciare ai propri vantaggi. Essa diventa efficace se sorretta da pressioni collettive all’applicazione, quali: approvazione sociale, status, onore ed altre ricompense o in caso di devianza, da sanzioni. Nella comunità del software Open Source la rinuncia al proprio interesse immediato, é compensato dal riconoscimento pubblico per il lavoro svolto, che può tradursi, anche se in un futuro incerto, in tornaconti economici, come la possibilità di trovare un buon lavoro grazie alla fama raggiunta.

Infine Coleman introduce due ultimi concetti. Il primo é l'”Organizzazione sociale appropriabile”, che prevede la possibilità di dirigere una rete di relazioni, nata con uno scopo, verso nuovi obiettivi. Questo avviene nel caso del software OS, in cui si tende a coinvolgere altri gruppi, e di conseguenza ad incrementare ulteriormente il capitale sociale. Il secondo concetto é quello delle “Organizzazioni intenzionali”, intendendo con questo le organizzazioni tradizionali costituiti da capitale sociale investito da un gruppo di attori per ottenere un qualche risultato. É questo il caso delle organizzazioni nate per gestire i progetti di sviluppo OS e delle organizzazioni a sostegno della sua diffusione.

A questo punto é evidente che il Capitale sociale ha le qualità del bene pubblico (si riveda il paragrafo 3.1): é inalienabile, é una risorsa con valore se in uso, non é facilmente sostituibile, non é proprietà delle persone che ne beneficiano. I fattori che aiutano la creazione e la conservazione del capitale sociale sono:

- L'immediatezza delle relazioni nel senso di connessioni dirette ed aperte tra i partecipanti al network.
- La stabilità delle relazioni che spesso é di tipo strutturato (gruppo di sviluppo, sviluppatori, utenti/tester, ...)
- L'ideologia non individualista e incentivante, l'aggregazione e la dipendenza reciproca basata sulla condivisione di conoscenze

Quindi a conclusione di questo discorso possiamo affermare che l'esito più importante del sistema istituzionale del software OS, che lo distingue nettamente dal software proprietario, é la presenza di un elevato Capitale Sociale di natura positiva, che aiuta lo sviluppo economico di tutta la società. La capacità di soddisfare i bisogni di informazione reciproca, la velocità con cui avviene lo scambio di informazioni nella comunità Open permette infatti di ottenere un sistema aperto verso l'esterno in cui la partecipazione anche come semplice utente novizio (*newbie*) non preclude la possibilità di diventare esperto e contribuire al progetto. Il modello proprietario invece, data la indisponibilità dei sorgenti del software, obbliga ad una chiusura su molti aspetti cruciali per l'innovazione, che rimangono, spesso in maniera sterile, custoditi all'interno di reti scollegate. Il capitale sociale che ne scaturisce, in molti casi, non é di natura positiva e favorisce esiti collusivi e monopolistici che danneggiano il sistema economico e sociale.

3.6 Gli svantaggi del software Open Source

Il software Open Source presenta anche delle criticità, sia da parte di chi lo adotta in azienda, sia di chi lo propone in una ottica di business. Per chi lo adotta (si comporta quindi come un cliente di una soluzione OS), le criticità da valutare attentamente sono in particolare i costi di gestione e migrazione. È necessario rispondere alle seguenti domande:

- ho disponibile internamente le conoscenze per gestire il software OS?
- Devo acquisire personale apposta?
- Ho un venditore affidabile per il supporto?
- Quanto costa il supporto?

Per chi sviluppa, o comunque si propone come “venditore”, il software OS può diventare insidioso per via delle questioni legali legate alle licenze, che limita spesso la possibilità di aggiungere software sviluppato internamente e che può costituire la fonte di reddito dell’azienda. Va quindi fatta una attenta analisi della questione legale legata allo sviluppo.

Vi sono inoltre dei freni all’adozione del software OS legati spesso alle difficoltà e costi di migrazione da piattaforma proprietaria a OS, non sempre fattibili o adatti al momento storico dell’azienda o dell’organizzazione. È notizia recente che il comune di Parigi ha rinnovato il contratto con Microsoft riguardo alla fornitura del sistema informativo dei PC di tutti i suoi uffici, perché ha calcolato che i costi di training necessari per riqualificare i propri tecnici all’uso di soluzioni Open, sarebbero stati superiore ai benefici derivanti dal risparmio delle licenze software pagate a Microsoft. È inoltre più difficile reperire personale con competenze OS, anche se la gran parte dei nuovi laureati ha skill adatti¹⁴.

I maggiori freni vengono comunque dalle imprese utenti. L’offerta si allarga di continuo, ma l’accettazione da parte degli utenti è sempre almeno un passo indietro: il software OS è generalmente una nuova esperienza in azienda, e il concetto stesso deve vincere una serie di resistenze culturali.

L’offerta di software OS può essere classificata in cinque aree, che partono da quelle relative al sistema operativo per arrivare al livello applicativo:

¹⁴ il 70% in base al Survey FLOSS 2003

- piattaforma/infrastruttura
- strumenti di sviluppo
- strumenti di integrazione/middleware
- strumenti di operation
- applicazioni di business

La maturità dell'offerta di software OS in ciascuna di queste aree è il fattore principale che guida la scelta del potenziale cliente. Mentre le soluzioni OS che riguardano l'infrastruttura e gli strumenti di sviluppo (le prime due aree dello schema) hanno una solida e lunga storia, che ha consolidato l'offerta attorno a pochi nomi (Linux, OpenBSD, Jboss, Eclipse, etc) di qualità riconosciuta, con servizi di supporto tecnico e skill diffusi sul mercato. Le utility a supporto dell'execution/operation (Xalan, Xerses, OpenNMS, etc) hanno un'offerta ampia e grazie alla loro natura specialistica consentono una scelta più tattica in funzione degli skill disponibili nell'azienda. In questo ambito anzi l'offerta OSS può essere l'unica alternativa, ovvero possono non essere disponibili soluzioni commerciali.

Le applicazioni di business soprattutto nelle grandi imprese richiedono una maggiore maturità prima di poter essere adottate al pari di soluzioni proprietarie. È chiaro che anche in questo settore l'OS giocherà un ruolo; questo è però un mercato ancora tutto da costruire. Ad oggi l'adozione di applicazioni OS al posto di applicazioni enterprise per sistemi mission-critical non è ancora diffusa.

Il modello Open Source presenta ovviamente dei limiti di cui bisogna essere consci. Uno di essi è la scarsa affidabilità, soprattutto in termini di tempi di rilasci, per i progetti che sono ancora allo stato embrionale, oppure che non hanno conosciuto quel successo tale a creare una comunità robusta e capace di automantenere lo sviluppo del progetto Open Source. Questo è abbastanza normale visto che lo sviluppo e il test sono affidati alla buona volontà di un nucleo ristretto di persone che, salvo casi di sponsorizzazione, non sono pagate per questo.

Altro aspetto è la alea che può circondare progetti che, sebbene molto innovativi, si blocchino perché non riescono ad attirare intorno a sé persone motivate ed interessate. E proprio per questo motivo vi sono moltissimi campi che non sono stati ancora coperti da buoni software free. In particolare si nota una mancanza di soluzioni Open Source in quei campi che affrontano tematiche "poco interessanti" per gli sviluppatori di software, come ad esempio i sistemi di controllo di guasti e di raccolta di statistiche. Per lo stesso motivo quasi tutte le attività a corredo del software, come la scrittura di documentazione puntuale, vengono trascurate

Sempre a sfavore citiamo la bassa **compatibilità con standard commerciali**. Il modello attuale di distribuzione del software Open Source, economicamente basato sull'offerta di servizi piuttosto che sulle licenze, è rischioso in un mercato aggressivo come quello dell'IT. Per questo motivo, alcuni distributori di OS si trovano in difficoltà. Ad esempio, Great Bridge, (creatore di PostgreSQL, un database) ha dovuto dichiarare fallimento. Anche grandi distributori di Linux come SuSE e Red Hat hanno avvertito difficoltà.

Infine la principale difficoltà, come riconosciuto dalla stessa comunità internazionale di utenti Linux, è la **mancanza di driver**: la maggioranza dei produttori di periferiche non forniscono driver per Linux, dunque la lista dell'hardware compatibile è limitata ai dispositivi a cui la comunità degli sviluppatori open source ha accesso. Quando viene lanciata sul mercato una nuova periferica, occorrono mesi prima che i driver siano disponibili, ammesso che i produttori forniscano le interfacce necessarie per lo sviluppo dei driver stessi; il problema è particolarmente evidente per le schede video e per i modem.

3.6.1 L'Open Source come bene pubblico: il problema del free riding

Sul sito della licenza GPL si dice che l'Open Source “*permette ad ognuno di dare un mattone per ricevere in cambio una casa*”. In realtà non é proprio così. É necessario che qualcuno abbia gettato le fondamenta. E se é in parte vero che le performance dei sistemi informatici siano guidate in maniera schiacciante dai progressi dell'hardware sottostante, piuttosto che dai progressi, in termini di algoritmi e architetture software, é anche vero che, affinché un progetto Open Source sia innovativo, richiede che abbia sotto un'architettura software complessa e sofisticata. É quindi necessario che un gruppo di persone dedichi molto tempo per progettare ed implementare lo scheletro del software, attività che richiede tempo e denaro, anche solo per il mantenimento primario degli sviluppatori.

Viceversa le rendite monopolistiche derivanti da posizioni di monopolio, che si possono ottenere con la brevettabilità del software, rendono convenienti l'attività di ricerca, in quanto consente di recuperare costi aziendali che vendendo al costo marginale andrebbero inevitabilmente perduti.

Il modello Open Source soffre pesantemente anche del cosiddetto problema del free riding. Grazie alla licenza GPL, il progetto Open Source diviene a tutti gli effetti un bene pubblico. Ricordiamo, come abbiamo già descritto nel paragrafo 3.1, che un bene pubblico é quel bene che ha le seguenti caratteristiche: 1) non rivale, cioè quando il suo consumo da parte di un soggetto non impedisce a un altro di godere dello stesso bene 2) non escludibile, ossia che non é possibile consentirlo ad un soggetto ed escluderlo ad altri. Il software Open Source, essendo offerto gratuitamente su Internet, soddisfa pienamente alle due condizioni elencate precedentemente. Il *free riding* si verifica quando i singoli individui sono tentati di evitare di pagare il prezzo di un bene, scaricandolo su qualcun altro. Ogni individuo sa che potrà beneficiare ugualmente del bene, senza che sia costretto a pagare e spera che siano gli altri consumatori a pagare al posto suo.

Sul modello cooperativo del software Open Source aleggia inoltre l'ombra della Tragedia dell'area comune di Garret Hardin¹⁵.

Hardin, ci chiede di immaginare un prato a uso comune in un villaggio di coltivatori, che vi conducono le proprie mandrie al pascolo. Ma gli effetti del pascolo degradano l'area comune, estirpando l'erba e lasciando macchie brulle in cui l'erba ricresce molto lentamente. Se manca una politica condivisa (e messa in atto!) per la distribuzione dei diritti di pascolo, al fine di prevenire gli eccessi, l'interesse di ciascuna delle parti starà nel far pascolare il maggior numero di capi possibile al più presto possibile, nel tentativo di trarre il massimo valore dal pascolo, prima che degeneri in un mare di fango.

¹⁵ F.A. Hayek, F.A. Hayek, *Road to Serfdom*, Routledge 2006

Se per altri beni pubblici (come l'illuminazione pubblica, le forze di sicurezza, ...) il problema del free riding viene risolto imponendo il pagamento attraverso la tassazione e di garantire l'offerta del bene mediante l'intervento pubblico, per il software Open source il discorso è più complesso.

Ritornando all'immagine del mattone e della casa presente sul sito della GPL, ci si può chiedere perché l'utilizzatore debba contribuire, anche solo minimamente, fornendo un solo mattone, al successo del progetto.

In realtà riguardo alla prima obiezione, quella riguardo lo scarso incentivo alla ricerca, il successo di Linux, per cui il volume di ricerca su Linux ha superato¹⁶ di gran lunga quanto Microsoft, regina dei monopoli software, spenda in ricerca, sta a testimoniare che esse non tengano conto di altri fattori. Nel caso di Linux infatti la crisi finanziaria scaturita dopo la bolla Internet ha costretto molti grossi produttori di telecomunicazioni a dimettere lo sviluppo di sistemi operativi proprietari per convergere su Linux, abbattendo così i costi in maniera significativa. Si parla in questo caso di funzione di input comune a molteplici processi produttivi. A dispetto del problema del free riding, la ricerca sullo sviluppo di un input comune a più processi in un contesto di GPL può risultare addirittura maggiore che in regime di monopolio protetto da brevetto. Se è presente un accrescimento abbastanza forte del profitto totale dell'industria dovuto alla qualità dell'input comune e alla conseguente crescente produttività dei processi produttivi delle singole imprese. L'impresa investe nell'input/bene comune nonostante così facendo vantaggi non solo se stessa ma anche la concorrenza, se al contempo accresce abbastanza la dimensione della torta da spartire. In realtà alla base di queste scelte c'è anche da considerare l'aspetto che il progresso tecnologico sta evolvendo così rapidamente che è impossibile, anche per le grandi aziende di poter seguire tutte le possibili tecnologie, favorendo quindi la nascita di consorzi e di progetti condivisi.

Ritornando al problema del free riding, la realtà empirica prova che la linea di tendenza è quella opposta. L'ampiezza e il volume dello sviluppo Open Source¹⁷ sono in crescita costante. È chiaro che il modello della tragedia dell'area comune non è in grado, per motivi sostanziali, di descrivere quello che sta succedendo in realtà.

¹⁶ P. Aghion e S. Modica, *Open Source without Free-Riding*, in preparazione, 2006

¹⁷ La diffusione dei progetti Open Source può essere misurata, per esempio, attraverso gli annunci pubblicati ogni giorno su www.freshmeat.net, che, come detto, è il sito che ospita i progetti Open Source.

Un aspetto della risposta si trova senz'altro nel fatto che l'utilizzo dei software, anche se affetto da free riding, non ne diminuisce il valore. Anzi la maggiore diffusione del particolare software oggetto di scambio ne accresce il valore d'uso e, nel tempo, aumenta le possibilità che arrivino nuovi partecipanti con il conseguente incremento di relazioni e di risorse per l'azione (informazioni, competenze, fiducia, reputazione, ecc ...). Oltre al valore d'uso aumenta anche il valore intrinseco del software, in quanto con la diffusione cresce la possibilità da parte degli utenti di aggiungere correzioni e funzionalità a loro piacimento ("patches" del codice). In quest'area comune all'inverso, l'erba cresce più alta quando si è in tanti a pascolare.

Lo spirito alla cooperazione può essere anche formalizzato attraverso la teoria dei giochi [24]. Chi contribuisce al progetto Open Source, scrivendo ad esempio del software che implementa una nuova funzionalità in un progetto Open Source ha solo due possibilità: o tenersi la patch, oppure metterla gratuitamente nello spazio comune. La prima scelta non presenta alcun guadagno. La seconda potrebbe non presentare un guadagno, ma potrebbe anche incoraggiare la distribuzione a catena ad altri utenti che, in futuro, si rivolgerebbero a questo signore per eventuali problemi. La seconda scelta, apparentemente altruistica, è in realtà egoista al punto giusto, nella prospettiva, appunto della teoria dei giochi.

In un'analisi di questa tipologia di cooperazione, è importante notare che, se esiste un problema di free riding¹⁸ esso non si aggrava in proporzione al numero degli utenti. La complessità e il costo delle comunicazioni che ruotano intorno a un progetto Open Source sono da considerarsi quasi del tutto una funzione del numero di sviluppatori coinvolti: la presenza di più utenti che non esaminano mai il codice sorgente, in realtà, non costa nulla, al massimo può far aumentare il numero di domande banali che compaiono nella mailing list del progetto.

I veri problemi creati dal free riding, nel campo del software Open Source, dipendono più che altro dai costi contingenti alla distribuzione stessa delle patches. Un collaboratore che abbia poco da perdere nel gioco culturale della reputazione, in assenza di un compenso in denaro, potrebbe pensare: "Non vale la pena distribuire questa correzione, perché dovrei ripulire la patch, scrivere un ChangeLog e firmare le carte di licenza della Free Software Foundation...". È proprio per questo che il numero dei collaboratori (e, in seconda istanza, il successo dei progetti) è fortemente e inversamente proporzionale alle difficoltà cui l'utente deve far fronte per collaborare a un progetto. Tali costi contingenti possono essere di natura sia politica, sia meccanica, e insieme

¹⁸ Il free riding può essere causato dal fatto che il lavoro può essere poco remunerativo in assenza di denaro o di altri compensi equivalenti

possono spiegare perché una cultura sconnessa e amorfa come quella di Linux abbia attratto enormemente più energie cooperative rispetto a sforzi organizzati sistematicamente e più centralizzati, come BSD, e perché la FSF abbia iniziato a perdere importanza con la diffusione di Linux.

3.7 Esempi di aziende Open Source (RedHat, SUN, MySql AB, Google, eBay ed Amazon)

Ci sembra inutile in questa sede presentare alcuni esempi di aziende che hanno adottato il modello Open Source.

RedHat ha lasciato alla community lo sviluppo della distribuzione Linux per desktop chiamato Fedora, garantendosi in tal modo un numero enorme di persone che volontariamente che volontariamente restano vari aspetti della distribuzione che possono poi essere introdotti nel vero business di Red Hat, ossia le distribuzioni Linux per server. Red Hat offre soluzioni Open Source con servizi end-to-end che comprendono la consulenza professionale di Red Hat Professional Consulting incentrata sulle infrastrutture multiplatforma e i servizi di engineering, i servizi per lo sviluppo di software e il porting del sistema operativo, nonché per le piattaforme e i dispositivi embedded, i servizi di supporto alle aziende per l'installazione e il supporto online/telefonico e i servizi di Red Hat Learning per la certificazione Linux (RHCE) e i corsi su Apache, i sistemi Embedded, lo sviluppo avanzato e l'e-Commerce. Red Hat Network assicura competenze uniche nell'impiego e nella gestione dei prodotti Open Source, nei servizi, nel supporto e nell'amministrazione delle informazioni online in tempo reale per aggiornare in modo affidabile le soluzioni basate sull'Open Source a costi convenienti.

Sun Microsystems è una delle società che più ha donato al mondo Open Source, ottenendo indietro risultati ancora difficili da percepire. Sun ha infatti donato (e finanziato) anni fa Star Office alla comunità che ha sviluppato l'attuale Open Office, software che sostituisce il pacchetto Office di Windows e che sta conoscendo un crescente successo.

Sun si è costituita fama e viene percepita dal cliente come produttore di server di qualità, non come software house.

MySQL AB, la società che ha rilasciato il nucleo del database MySQL ha licenziato il suo software con due modalità: un'alibera, con licenza GPL, e una a pagamento per chi vuole includere il database in programmi a codice chiuso.

Larry Ellison, CEO di **Oracle**, recentemente ha dichiarato che l'azienda si muoverà verso il modello di business del software Open Source, dove le applicazioni vengono fornite al cliente in modo gratuito e che invece prevede un pagamento per tutte le personalizzazioni, la manutenzione ed il supporto tecnico. Oracle quindi oltre ai suoi software proprietari, tra cui emerge il database Oracle, che è database commerciale più usato, agirà come system integrator di soluzioni Open Source, aumentando il proprio portafoglio servizi e prodotti. Oracle rappresenta quindi l'esempio

di azienda che produceva software proprietario e a cui vuole aggiungere una proposta commerciale.

Analizziamo adesso tre aziende molto note ai navigatori Internet come eBay, Amazon e Google. Queste aziende non distribuiscono od integrano software Open Source, come quelle descritte precedentemente. Esse usano soluzioni Open Source per offrire servizi. Modello di business che forse rappresenta il vero vincitore, in termini di fatturato complessivo, della sfida Open Source.

EBay é un esempio di compagnia che ha avuto successo grazie agli effetti della rete Internet. Il vantaggio competitivo é stato raggiunto grazie ad una massa critica di compratori e di venditori, che costituisce una vera e propria barriera d'ingresso per i nuovi entranti. Non c'è infatti nessun vantaggio per un compratore o un venditore ad utilizzare un nuovo sito.

Il caso di **Amazon** é ancora più interessante. Amazon vende prodotti disponibile anche da altri venditori. Ciò nonostante Amazon gode di un forte vantaggio competitivo. Non é solo costituito da prezzi migliori, servizi più efficienti, un marchio più conosciuto. Il vero fattore discriminante é costituito da come Amazon sfrutti la propria base di utenti. Quando si effettua una ricerca sul sito di Amazon, la pagina che mostra i risultati é divisa in quattro parti principali: sulla parte superiore vengono mostrati i tre prodotti più diffusi, subito sotto é possibile selezionare i criteri per ordinare i prodotti (in base al prezzo, alla valutazione degli altri utenti, oppure semplicemente in base all'ordine alfabetico). Sulla destra e sulla sinistra della pagina sono disponibili le liste "ListMania". Queste liste permettono ai clienti di condividere consigli per altri articoli relativi alla ricerca appena effettuata. La sezione "most popular" é il risultato di un complesso e proprietario algoritmo che combina non solo le vendite ma anche le valutazioni degli altri clienti, le raccomandazioni per articoli simili, e tutte le altre variabili che Amazon ritiene degne di nota. Amazon é riuscita a creare un vantaggio competitivo perché ha compreso una delle principali lezioni dell'Open Source: "tratta i tuoi utenti come co-sviluppatori".

Google rappresenta un diverso caso di studio. Google é un enorme database, basato su software Open, che continuamente si aggiorna con contenuto di Internet. L'innovazione iniziale di Google é stato l'algoritmo, proprietario, per valutare le pagine web, che "pesa" il contenuto semantico di una pagina rispetto alla stringa di ricerca, per permettere di ordinare le varie pagine in base alla pertinenza, ed offrire migliori risultati di ricerca ai propri utenti. Il business model di Google é basato sulla pubblicità, mirata, effettuata agli utilizzatori in base alle ricerche effettuate. Nel caso di Google la partecipazione degli utenti é solamente estrinseca e può essere copiata dagli altri competitori. Per questo si comprende come mai Google sia attivamente impegnata a sviluppare ed offrire nuovi servizi, come Gmail o orkut.

Google é un database del contenuto di Internet. Clay Shirky, che lavora per O'Reilly's, ha notato come siano tre i modi per costruire dei grandi database. Il primo, dimostrato per primi da Yahoo, consiste nel pagare gli altri affinché lo facciano al posto suo. Il secondo, é di utilizzare le metodologie collaborative dell'Open Source per trovare dei volontari che lo facciano al posto tuo. Basti citare Wikipedia. Nasper, e tutti gli altri sistemi Peer to Peer (come eMule, Kazaa, Ares, LimeWire e in parte anche Skype, ..) é invece la dimostrazione della terza via. Questi sistemi peer to peer fanno in modo che ogni utente condivida delle risorse multimediali (film, video, ebook e soprattutto brani musicali) con tutti gli altri utenti della rete. In questo modo ogni utente aiuta a costruire un database distribuito.