www.copyleft-italia.it

p r e s e n t a

AUTORE: Sabrina Ragusa

TITOLO: Sviluppo e ordinamento istituzionale nel mercato del software: il modello del software libero.

DATA DI RILASCIO AL PUBBLICO: febbraio 2005

TIPO DI OPERA: Tesi di laurea - Facoltà di scienze politiche

****

"Omnium hominum quos ad amorem veritatis natura superior impressit hoc maxime interesse videtur: ut, quemadmodum de labore antiquorum ditati sunt, ita et ipsi posteris prolaborent, quatenus ab eis posteritas habeat quo ditetur.
Longe nanque ab offitio se esse non dubitet qui, publicis documentis imbutus, ad rem publicam aliquid afferre non curat, non enim est lignum quod secus decursus aquarum fructificat in tempore suo, sed potius perniciosa vorago semper ingurgitans et nunquam ingurgitata refundens."1

---

1 Dante Alighieri, De Monarchia, Libro Primo, I, (1313-1318).
Il principale ufficio di tutti gli uomini, che una forza soprannaturale ha improntato all'amore della verità, questo spetta sopra ogni cosa: come essi si sono arricchiti dall'opera degli antichi, così a loro volta proiettino l'opera propria verso la posterità, tanto che questa trovi in essi di che arricchirsi.
Stia certo di essere lontano dalla propria missione colui che, formatosi agli insegnamenti ricevuti dalla società, non si cura di portare qualche contributo al bene comune: costui non è pianta che, cresciuta lungo un corso d'acqua, fruttifica alla sua stagione, ma piuttosto malefica voragine che sempre inghiotte, e mai rifonde.

****

L A L I C E N Z A

- -

Attribuzione-NonCommerciale-NoOpereDerivate 2.0 Italia

Tu sei libero:

- di distribuire, comunicare al pubblico, rappresentare o esporre in pubblico l'*opera*

Alle seguenti condizioni:

Attribuzione.
Devi riconoscere la paternità dell'*opera* all'*autore originario*.

Non commerciale.
Non puoi utilizzare quest'opera per scopi commerciali.

No opere derivate.
Non puoi alterare, trasformare o sviluppare quest'*opera*.

- In occasione di ogni atto di riutilizzazione o distribuzione, devi chiarire agli altri i termini della licenza di quest'opera.
- Se ottieni il permesso dal titolare del diritto d'autore, è possibile rinunciare a ciascuna di queste condizioni.

Le tue utilizzazioni libere e gli altri diritti non sono in nessun modo limitati da quanto sopra.

Questo è un riassunto in lingua corrente dei concetti chiave della licenza completa (codice legale) che è disponibile all'URL http://creativecommons.org/licenses/by-nc-nd/2.0/it/legalcode

INDICE

Introduzione

Un simpatico pinguino sorride dagli scaffali di un negozio di prodotti informatici. Tux è il logo-mascotte del sistema operativo Linux.
Linux ha appena compiuto 10 anni, è giovane ma è cresciuto talmente in questi pochi anni da indurre molti a pensare che stia cambiando le regole del gioco nel mondo informatico.
IBM, Sun Microsystems, Hewlett-Packard, e molte altre stanno investendo pesantemente nel suo sviluppo, lo ritengono strategico in un comparto economico strategico e confidano in lui per il futuro: nel 2001 la sola "Big Blue" ha investito un miliardo di dollari[1].
Cosa c'è di sorprendente o insolito?
Linux è Software Libero[2]. Nessuno mai ne diventerà proprietario esclusivo.
Lo sviluppo del software oltre che realizzarsi all'interno delle imprese e della comunità scientifica presso le università, avviene anche all'interno delle comunità di liberi programmatori che si dedicano gratuitamente a questa attività. Questo "strano mondo" viaggia parallelo, si sovrappone, collabora od entra in conflitto con gli altri due, in uno scenario complesso, fluido e in continua evoluzione.
Il Software Libero è il prodotto delle comunità di programmatori liberi, gli hacker[3]. Ha la particolarità di essere sviluppato con un sistema di tipo cooperativo ed aperto. È reso disponibile a chiunque in formato sorgente[4] ed è protetto da una tipologia di licenze che ne ammette il libero uso, copia e redistribuzione in forma anche modificata, gratis o a pagamento. Queste licenze, nel pieno rispetto della legge, sovvertono i divieti dei copyright tradizionali e assicurano per sempre al Software Libero la natura di bene pubblico puro.
L'idea cardine per gli hacker è che la condivisione delle informazioni sia un bene di per sé - e non solo in campo informatico - e che quindi occorra incoraggiarla e sostenerla in ogni sua forma.
L'informatica, però, è per prima cosa una scienza e della scienza segue logica e leggi. Il modello di sviluppo delle comunità del Software Libero è per molti versi analogo a quello della comunità scientifica e, di fatto, queste comunità sono parzialmente sovrapposte e condividono persone, mezzi e paradigmi: il "gioco" della reputazione, il presupposto della pubblicità e controllabilità del lavoro svolto, la revisione ed il giudizio critico da parte dei pari grado. Tant'è vero che spesso l'inizio dell'attività di hacking dei programmi, comincia negli anni dell'università presso gli istituti di informatica, impegno che poi difficilmente viene abbandonato con l'ingresso nel mondo del lavoro.
Quelle degli hacker sono comunità che, anche se molto variegate, mantengono un codice morale comune che guida il comportamento individuale.
La lunga tradizione culturale degli hacker, comprende abilità tecniche altamente specializzate, stabili reti di scambio delle informazioni, norme, gerarchie di status, linguaggi e significati simbolici condivisi.
I programmatori della comunità hacker che partecipano ai progetti di sviluppo software, tra i numerosissimi che avvengono in rete, seguono vincoli informali e formali ben precisi. Considerano il proprio contributo come bene pubblico e lo rilasciano alla comunità. Il dono del proprio lavoro, del codice che hanno sviluppato, non chiude un cerchio di reciprocità perché ciò che hanno restituito non è in alcun modo confrontabile con ciò che hanno a disposizione: il "Calderone Magico"[5], a cui possono attingere, ha un contenuto di conoscenze nuove che continua a crescere, ma attingere a queste conoscenze non le consuma affatto, inoltre restituire è un dovere morale che fa alzare ulteriormente il livello quantitativo e qualitativo.
La possibilità di attingere ad un vastissimo Capitale Sociale di reciprocità[6] è l'incentivo fondamentale; l'opportunità di utilizzare le strutture sociali per perseguire i propri fini, siano essi la diffusione libera ed incondizionata delle conoscenze, il puro divertimento o la disponibilità di software liberi e gratuiti per il proprio lavoro. Il tutto in un ambito trasparente e pubblico dove tutti sono collegati e sanno o possono rapidamente sapere chi fa e/o sa che cosa. In ogni momento è possibile ricostruire lo stato dell'arte e l'accesso alla conoscenza è relativamente facile ed immediato. Chi partecipa assimila competenze tecniche altrimenti difficili da acquisire e trova aiuto nella soluzione dei problemi. Il Software Libero tiene traccia di coloro che hanno contribuito, pertanto chi dona tempo e lavoro, ad esempio sotto forma di nuovo codice o di bugs-report, ottiene una sorta di "certificazione" delle proprie competenze e migliora la propria reputazione in relazione a ciò che ha donato. I nomi degli hacker che collaborano allo sviluppo del software libero vengono associati ai contributi che forniscono, nelle credit list o negli history files[7] allegati a ciascun progetto.

Una buona reputazione porta vantaggi immateriali come il riconoscimento di un elevato status nella comunità, ma anche una migliore quotazione nel mondo del lavoro che si può eventualmente tradurre in consistenti vantaggi materiali. Gli sviluppatori più prolifici attraggono l'attenzione e le aziende possono trovare nelle credit list i nominativi dei migliori programmatori da ingaggiare.
Guardando ai molteplici traguardi della comunità, il sistema operativo GNU/Linux[8] è oggi il software più conosciuto, anche tra gli utenti non informatici, e raccoglie sempre più attenzione tra gli operatori del settore. Ha contribuito molto al rapido acquisto di visibilità del Software Libero cui stiamo assistendo.
Il software che ha avuto più successo è Apache[9] che di fatto è il più usato tra tutti i web server. Ma la realizzazione più consistente del Software Libero è la stessa Internet, nemmeno immaginabile per ciò che è diventata, senza un ambiente pubblico di libere collaborazioni in grado di coordinare la connessione del suo intreccio caotico di soluzioni e programmi e soprattutto in grado di mantenere aperti gli standard.
La scoperta di Internet da parte del grande pubblico ha consentito al sistema, da chiuso che era, di divenire aperto, dando inizio alla partecipazione, anche estemporanea, di persone esterne alle comunità.

Il Software Libero gode di un fenomenale tasso di sviluppo, di una evoluzione rapida in diretta risposta alle esigenze dei suoi utenti ed è disponibile per qualsiasi uso, anche commerciale. È per questo che inizia ad attirare l'attenzione, ed ingenti investimenti, di alcune tra le maggiori aziende informatiche. Si sta imponendo in molti casi come valida alternativa al software proprietario sia come prodotto in sé sia come metodo di sviluppo e filosofia d'affari: un cospicuo patrimonio di software liberamente utilizzabile è già presente e le licenze Free ed Open Source agevolano la partnership aziendale per lo sviluppo congiunto di nuovi prodotti con la 'messa in comune' dei reparti R.&S. In questo modo viene ripartito il rischio, molto elevato per la particolare struttura dei costi dell'industria del software e per la distorsione del settore data dalla presenza di un concorrente in posizione pressoché di monopolio nei comparti strategici.
Le fonti di profitto si spostano dalla vendita delle licenze d'uso del software, all'offerta di assistenza e servizi a valore aggiunto, decisamente più apprezzati dalla clientela.
Inoltre, anche coloro che utilizzano in modo commerciale il Software Libero, ad esempio, utilizzando programmi Free per fornire servizi alla propria clientela, oltre alla disponibilità dei sorgenti per eventuali modifiche e aggiustamenti da parte dei propri tecnici interni, hanno accesso al suo capitale sociale: il contatto diretto con la specifica comunità di sviluppatori e la possibilità di trovare tecnici esperti direttamente nella credit list del software utilizzato, nella comunità di sviluppo e supporto o tra gli appassionati attraverso le liste di discussione. Per l'impresa, tutto ciò, costituisce un importante vantaggio competitivo.

Nonostante l'attrito tra la mentalità Free e quella Business, il Software Libero sta avendo un impatto rivoluzionario sull'intero mercato del software. Per superare le ultime resistenze sono in corso di formalizzazione nuove istituzioni e le organizzazioni nate a sostegno del Software Libero, la Free Software Foundation e la Open Source Initiative, in testa ma non da sole, sono molto attive su questo fronte.
Per la sua stessa essenza, il software Libero trasforma il confine tra impresa ed utente finale che diviene sfumato o scompare. Il passaggio di potere dalle imprese agli utilizzatori è indubbiamente enorme.

Per analizzare il modello economico del Software Libero, seguirò lo schema analitico-interpretativo proposto da D. C. North in "Istituzioni, cambiamento istituzionale, evoluzione dell'economia"[10]. In questo saggio North intreccia l'analisi istituzionale alla teoria neoclassica modificando la dottrina esistente. Costruisce un intera struttura teorica indicando il processo di sviluppo condizionato dai punti di partenza come la chiave per la comprensione analitica delle trasformazioni economiche di lungo periodo.
L'approccio collega gli elementi della scarsità-concorrenza e l'idea degli incentivi come forza trainante, presenti nella teoria neoclassica, quindi, unisce alla teoria i costi del processo di scambio detti di transazione (per definire, proteggere ed applicare i diritti di proprietà sui beni), i modelli soggettivi della realtà, l'informazione incompleta e i rendimenti crescenti, caratteristici delle istituzioni. Il risultato è un sistema teorico in grado di collegare il livello dell'attività microeconomica con quello degli incentivi macroeconomici forniti dal sistema istituzionale.
Il processo di sviluppo condizionato dai punti di partenza deriva da meccanismi a rendimento crescente che ne determinano il senso. La direzione presa è poi rafforzata dagli stessi meccanismi: in alcuni casi si creano situazioni di scambio irrigidite, mentre in altri casi si verificano cambiamenti dinamici ed evolutivi.

Scostamenti, deviazioni o inversioni del senso del processo provengono dagli effetti inattesi delle scelte e da forze esterne al sistema considerato.
Le istituzioni, le "regole del gioco" formali ed informali di una società, sono la struttura fondamentale dell'interazione che ha permesso nel tempo di costruire un ordine sociale, economico e politico. Sono la

chiave interpretativa delle interrelazioni tra politica ed economia e delle conseguenti ricadute sulla crescita economica. Riducono l'incertezza degli scambi; insieme alla tecnologia impiegata, determinano i costi di transazione e di trasformazione e quindi la redditività e le opportunità di chi si impegna in attività economiche.
Le varie organizzazioni, i "giocatori", sono gruppi di persone unite dal comune proposito di raggiungere uno scopo prefissato, per forma e finalità sono il prodotto dell'insieme di vincoli ed opportunità consentito dalla struttura istituzionale. Attraverso l'attività degli imprenditori che le dirigono, sono impegnate attivamente per il raggiungimento dei propri obiettivi. Ma nel corso della realizzazione dei propri fini, le organizzazioni modificano in maniera graduale il sistema istituzionale. La coppia istituzioni-organizzazioni è quindi il motore del cambiamento.
La fonte delle trasformazioni economiche, sono i guadagni ottenuti dalle organizzazioni e dai loro imprenditori a seguito dell'acquisizione di competenze, conoscenze ed informazioni, che migliorano la possibilità di concretizzare i propri obiettivi.
I vincoli imposti dai limiti dell'organizzazione, dal reddito e dalla tecnologia vanno considerati come fattori endogeni insieme al sistema istituzionale. La tecnologia determina il limite superiore dello sviluppo economico possibile. Se i costi di transazione fossero nulli, gli aumenti della conoscenza scientifica e della sua applicazione fornirebbero la soluzione per accrescere il potenziale benessere delle persone nelle diverse società. I costi di transazione non sono mai nulli ma possono essere ridotti dalla struttura istituzionale.
Ultimo, ma non meno importante, fattore considerato è la lotta interminabile per risolvere i problemi inerenti all'attività sociale di coordinamento e di cooperazione. Tema centrale, nel saggio di North, è il difficile problema del raggiungimento di soluzioni cooperative su base volontaria così da consentire ai sistemi economici di impadronirsi dei vantaggi degli scambi, trarre profitto dalla tecnologia e dai vari aspetti dell'impegno politico e sociale.
Ecco il punto a cui il North è arrivato: gli incentivi costituiscono le ragioni di fondo dell'evoluzione economica e variano molto nel tempo. L'attenzione su di essi dà la chiave interpretativa della evoluzione di un sistema economico. Occorre integrare l'analisi istituzionale nella teoria e nella storia economica, correggendo il concetto di razionalità e le sue implicazioni, includendo idee e ideologie nell'analisi dello studio dei costi di transazione per il funzionamento dei mercati - politici ed economici - e comprendendo gli effetti del processo di sviluppo condizionato dai punti di partenza sull'evoluzione dei sistemi economici.
L'oggi è condizionato dal passato semplicemente perché i vincoli di ieri limitano le scelte, ma ciò riflette un ruolo più profondo del processo di sviluppo condizionato, frutto dei rendimenti crescenti del sistema istituzionale.
L'accresciuta importanza che le organizzazioni politiche ed economiche hanno assegnato al sistema istituzionale, attraverso esternalità di rete e altre fonti di rendimenti crescenti, lascia il segno nell'evoluzione economica. Ma anche le organizzazioni provocano trasformazioni graduali in una combinazione di stabilità e cambiamento.
Lo schema del North infine orienta su due elementi che compongono il quadro istituzionale dei sistemi economici ritenuti all'origine di istituzioni efficienti: i vincoli informali e i costi di transazione relativi al processo politico.
I vincoli informali derivano dalla trasmissione dei valori attraverso la cultura tradizionale, dall'applicazione di regole formali per risolvere specifici problemi di scambio e dalla soluzione delle questioni di coordinamento. Egli segnala la loro ampia influenza sulla struttura istituzionale data dall'interazione con le regole formali. Le tradizioni di lavoro, di onestà e di integrità riducono i costi di transazione consentendo lo scambio complesso e produttivo. Quelle tradizioni sono sempre rafforzate dalle ideologie alla base degli atteggiamenti. Le convinzioni soggettive degli attori non sono solo il risultato di una cultura, ma sono continuamente modificate dall'esperienza filtrata da schemi mentali preesistenti e culturalmente condizionanti. Perciò le variazioni dei prezzi relativi alterano a poco a poco norme e ideologie e più sono bassi i costi di informazione, più sono rapide le alterazioni.
Per effettuare una corretta analisi occorre connettere tra loro il vincoli informali ed il costo di transazione relativo al processo politico. Questo infatti è molto elevato anche nel migliore dei mercati politici. Abitualmente però l'operatore politico possiede ampia libertà decisionale. Pertanto si avranno istituzioni efficienti solo in un sistema politico dotato di incentivi per la creazione e la tutela efficiente dei diritti di proprietà.
La struttura teorica, appena presentata, funziona da traccia per inquadrare il modello economico del Software Libero. L'autore la utilizza per descrivere e comprendere le performance economiche di entità territoriali ma è perfettamente adattabile ai più disparati "campi di gioco" - per riprendere una metafora dello stesso North -, come appunto uno specifico ambito come quello del software prodotto in rete, privo di confini, aperto e globalmente interconnesso.

Questo lavoro si suddivide in cinque capitoli.
Il primo contiene una ricostruzione della storia degli hacker: come e dove nascono le prime comunità ed il formarsi della loro cultura. Il fenomeno di cui parlo ha origine nell'ambiente accademico statunitense già negli anni cinquanta, segue prevalentemente le regole dell'ambiente scientifico ed accademico, ma anche alcune caratteristiche della società americana (contesto che premia l'attività delle organizzazioni e il loro sviluppo in termini di competenza e conoscenza scientifica). In questo fatto si può identificare il punto di partenza del sentiero di sviluppo[11].
Nel secondo capitolo, presento i vincoli informali della comunità. Oggi la comunità hacker "vive" innanzitutto sulla Rete e risulta uniforme nonostante la distanza fisica dei singoli, la diversa nazionalità, lingua e cultura di origine. Evidenzio in questo capitolo gli elementi che compongono la filosofia, ideologia, norme etiche e mentalità tipica delle diverse comunità di hacker, quindi, le relative consuetudini incentrate sulla libertà e condivisione del sapere e sulla logica del dono.
Confronto poi, l'ideologia, l'etica e le 'regole del gioco', seguite dalla comunità hacker, con quelle della comunità scientifica, sottolineandone le analogie. Proseguo, esponendo come è organizzato il processo produttivo del Software Libero: si realizza intorno al singolo progetto, con la cooperazione volontaria e gratuita dei programmatori, sempre molto numerosi.
Per progetti molto importanti come Linux possono arrivare a toccare le centinaia di migliaia. L'azione di coordinamento, stimolo e controllo dei leaders e le strutture che riescono a comporre sono fondamentali perché portano alla riuscita o al fallimento dei progetti. Attraverso l'esempio del progetto GNU, di Linux ed altri particolarmente significativi ne spiego il tipo ideale di funzionamento.
Infine espongo ciò che distingue strutturalmente e qualitativamente la programmazione 'Libera' rispetto a quella 'Proprietaria', attraverso il concetto di 'Capitale sociale', che, nelle comunità di programmatori del Software Libero è presente in quantità e qualità, molto differenti.
Nel terzo, affronto la questione cardine della natura del software: è un prodotto digitale/opera dell'ingegno che, in più, ha una particolare struttura dei costi. Il software, a differenza del prodotto industriale classico, è affetto da una diseconomia di scala dei costi di sviluppo rispetto alla dimensione del prodotto. Il costo di sviluppo di un programma cresce all'incirca con il quadrato delle sue dimensioni ed in più, è stato stimato che, il numero dei bugs cresca con il cubo delle dimensioni. Il costo di riproduzione, per contro, è pari quasi allo zero.
Direttamente collegato alla particolare natura del software c'è la questione della tutela della sua proprietà. Le licenze hanno il difficile compito di tutelare due proprietà ben distinte, quella morale e quella economica dell'autore del software. Mantengo questa distinzione per effettuare un confronto tra le licenze sul Software Libero - il "Copyleft" GNU/GPL, la Open Source Definition e le licenze OSI Certified - che tendono soprattutto a tutelare la proprietà morale dell'autore, e le licenze sui software "chiusi", che proteggono più la parte economica. Open e Closed Source hanno quindi due strutture completamente diverse dei costi totali.
Presento anche le principali organizzazioni senza scopo di lucro del Software Libero: la Free Software Foundation con il Progetto GNU e la Open Source Initiative. Descrivo brevemente, la loro azione di promozione e difesa di questo tipo di software, e le attività per far conoscere e diffondere la filosofia che vi sta dietro.
Poi mi occupo del modello del Software Libero come strategia di mercato. Nuove aziende sono nate per dedicarsi al Software Libero: "distribuzioni" e aziende di servizi, supporto e certificazione. Le regole del mercato restano dominanti, ma il Software Libero ne ha introdotte di nuove e le aziende si devono adeguare con le loro strategie per mantenere o raggiungere vantaggi competitivi, crescita della produttività e possibilità di risposte rapide e flessibili in un comparto estremamente dinamico.
Il Software Libero favorisce il comportamento cooperativo nel settore. Progetti collaborativi tra aziende sono messi in atto sempre più spesso, ottenendo riduzione dei costi e ripartizione del rischio.
A conclusione del capitolo, affronto il problema del terzo garante e della tutela dei diritti in ambito internazionale. L'applicazione delle regole formali in un sistema globalizzato ed in rapida evoluzione non è semplice ed è sicuramente molto costosa per chi la affronta e a volte è del tutto inutile dato che in alcuni stati, per ragioni politiche, economiche o sociali, restano volutamente inapplicate.
Diverse questioni risultano aperte, come ad esempio quella dei brevetti. Le decisioni che verranno prese incideranno sul futuro della ricerca scientifica mondiale, in tutti i campi, non solo sull'informatica.
Il capitolo quarto, lo dedico ad una ricognizione delle dinamiche del Software Libero all'interno del 'Sistema Software', reinterpretandolo attraverso l'iter teorico di North e cogliendo gli elementi che hanno inciso nel cambiamento del mercato, quindi anche il Capitale Sociale quale risorsa per l'azione.

Nel capitolo conclusivo, restando all'interno dell'iter teorico di North, sposto la visione temporale in una prospettiva futura, e rilevo gli elementi in gioco, che andranno ad incidere sulla forma e direzione del cambiamento istituzionale.

****

Userò 'Software Libero' come termine generale che identifica idealmente tutta la comunità, intendendo con questo sia il 'Free Software' sia l'Open Source Software'.

****

1 Fonte: Il Sole-24 Ore - 12 Ottobre 2001.
2 Uso Software Libero come termine generale che comprende sia Free Software sia Open Source Software.
Free Software: Il termine inglese free significa sia gratis che libero. Free in questo contesto è da intendere sempre come libero. Infatti il "Free Software" viene distribuito sia gratuitamente che dietro compenso. Ma a differenza del software gratuito ma proprietario è accompagnato dai sorgenti e da licenze che consentono delle libertà che gli altri copyright non riconoscono.
Open Source: (sorgenti aperti) un software per poter essere definito Open Source deve fare uso di una delle licenze certificate come conformi dalla Open Source Initiative, organizzazione esclusivamente destinata alla gestione della campagna Open Source e della sua certificazione di marchio, che prevedono come fondamentale e comune condizione il rilascio dei codici sorgenti (da qui Open Sources); nelle intenzioni iniziali "Open Source" doveva essere una certificazione (una forma speciale di marchio che potesse applicarsi secondo i termini ai prodotti altrui) registrata ma non è stato possibile ufficializzarlo. Il marchio attualmente in via di registrazione è "OSI Certified"; il termine Open Source è stato introdotto dall'ala moderata della comunità hacker, più permissiva riguardo al mercato per renderlo ad esso accettabile e per evitare il termine free di Free Software usato dagli hackers più radicali con il suo doppio significato (libero e gratuito) e la sua valenza provocatoria e anticommerciale.
3 Il termine ha un significato differente all'esterno ed all'interno della comunità: nel primo caso, pirati disonesti che violano i sistemi in cerca di informazioni riservate, o che addirittura li distruggono, nel secondo, ed è il significato corretto, qualcuno che ami programmare e che lo fa appassionatamente.
Hacker1: "[...]1. Una persona a cui piace esplorare i dettagli dei sistemi programmabili e i modi per estendere le loro caratteristiche, in contrasto con la maggioranza degli utenti che preferisce imparare solo il minimo necessario. 2. Qualcuno che programma appassionatamente (persino ossessivamente) o che ami programmare piuttosto che limitarsi a teorizzare sulla programmazione. 3. Una persona in grado di apprezzare i valori dell'hacking (vedi). 4. Una persona abile a programmare velocemente. 5. Un esperto in un particolare programma, o che lo usa con particolare frequenza o ci lavora su; come in "Unix hacker". (Le definizioni dalla 1. alla 5. sono correlate e le persone che le soddisfano sono assimilabili.) 6. Un esperto o un appassionato di qualunque tipo. Uno può essere un hacker dell'astronomia, per esempio. Raymond "Jargon File" 4.2.2 20/08/2000.
4 Interpretabile dagli esseri umani e non solo dalle macchine. Voce glossario.
5 E.S. Raymond, 1999.
6 A.Pizzorno, "Perché si paga il benzinaio. Nota per una teoria del capitale sociale", in Stato e Mercato n° 57, dicembre 1999, pp. 373-394.
7 History file: cronologia delle versioni di un software che include l'indicazione dei singoli miglioramenti e dei relativi responsabili. Generalmente è allegato ai programmi sotto forma di file.
Credit list: lista di coloro che hanno contribuito alla realizzazione di un progetto. Generalmente è allegato ai programmi sotto forma di file.
8 GNU è il progetto che ha fornito gli strumenti di sviluppo al progetto Linux (kernel del sistema operativo che ne ha preso il nome).
9 Apache è un programma per web server: un software che consente ad un computer di offrire un particolare servizio ad un altro computer sul quale gira il corrispondente client.
10 D.C. North, "Institutions, Institutional Change and Economic Performance", Cambridge, Cambridge University Press, 1990; trad. it. "Istituzioni, cambiamento istituzionale, evoluzione dell'economia", Bologna, Il Mulino, 1994.
11 Teoria della dipendenza dello sviluppo dai punti di partenza - path dependency.

1 La nascita della cultura hacker.

1.1 La storia degli hacker.

"I vincoli informali contano. Per dare migliori risposte si deve conoscere molto di più sulle norme di comportamento derivanti dalla tradizione culturale e su come interagiscono con le regole formali"[1]
La struttura teorica di North indica il 'processo di sviluppo condizionato dai punti di partenza' come la chiave per comprendere le trasformazioni economiche di lungo periodo. Occorre quindi, volgere lo sguardo indietro nel tempo per individuare quegli elementi che andranno poi a condizionare gli sviluppi successivi.
La frase che conclude il saggio di North e che apre questo primo capitolo sta a sottolineare l'importanza dell'individuazione dei vincoli informali, che sono la chiave per la comprensione di un determinato sentiero di sviluppo. Esso è condizionato dai punti di partenza e i vincoli, che rimangono alla base, servono per esaminare la performance di una particolare 'economia' nel lungo periodo.
Qui di seguito, riporto una ricostruzione - necessariamente parziale - della storia degli hacker americani. Il fine è quello di indicare i passaggi della formazione, consolidamento e cristallizzazione dei vincoli informali hacker, che condizionano il sentiero di sviluppo, e che isolerò e descriverò nel capitolo 2.

[La ricostruzione storica è finalizzata all'analisi successiva. I fatti riportati in questo paragrafo sono prevalentemente tratti da Levy, S., 'Hackers', 1984. Principale fonte di informazioni del citato libro è costituita da oltre un centinaio di interviste, condotte dallo stesso Steven Levy, tra il 1982 ed il 1983, e da consistente materiale documentale. Il paragrafo è integrato dalle seguenti altre fonti: Berra, A., Meo A.R., 2001; DiBona, C., Ockman, S., Stone, M. (a cura di), 1997; Hafner, K., Lion, M., 1996; AAVV, Il mondo del computer, 1987; http://www.attivissimo.net di Paolo Attivissimo.]

1.1.1 La genesi della comunità hacker.

La storia degli hacker[2] si intreccia con quella della scienza e dei progressi della tecnologia informatica. Non è possibile stabilire una data precisa della nascita di questa comunità, ma si può scegliere, come inizio di questa storia, la realizzazione dei primi calcolatori programmabili a valvole (detti di prima generazione)[3] con l'avvio dello studio teorico del funzionamento e della programmazione di queste nuove macchine.
I Real programmer, così furono chiamati i primi programmatori, precursori degli hacker, provenivano dai settori dell'ingegneria, della matematica e della fisica. Programmavano in 'linguaggio macchina' ossia con sequenze di O e I, operazione lunga e complicata che richiedeva la conoscenza perfetta dell'architettura del calcolatore, dei codici, di tutte le istruzioni ed il controllo mentale di molti altri elementi[4]. L'attività dei Real programmer esigeva curiosità intellettuale ed inclinazione verso la scoperta attraverso la sperimentazione diretta ed un continuo impegno di ricerca per capire intimamente e profondamente il funzionamento della macchina. Essi trasformano il principio strumentale, dell'accesso alla conoscenza e all'informazione libero ed aperto, in principio ideale.

Nella primavera del 1959 venne inaugurato, presso il Massachusetts institute of technology (MIT) dell'Università di Cambridge, il primo corso di programmazione per computer. L'insegnante era un docente di matematica, John McCarthy: fu lui ad avviare la ricerca sull'Intelligenza Artificiale.
Alla fine degli anni '50 ancora non esisteva ufficialmente la scienza informatica e il corso di McCarthy al MIT dipendeva dall'istituto di ingegneria elettrica.
Tra i suoi studenti si contavano molti appartenenti al Tech Model Railroad Club (Tmrc), un organizzazione studentesca che si dedicava al modellismo ferroviario. Gli studenti, che trascorrevano il loro tempo libero intorno ad un enorme plastico ferroviario nei locali del club, erano divisi in due gruppi. Il primo riproduceva i modellini dei treni e le scenografie. Il secondo girava intorno al Signal and power subcommittee (S&P), dove c'era 'il sistema': una sofisticata attrezzatura a disposizione dei modellisti in gran parte regalata al club dalla compagnia telefonica[5].

Usando quell'attrezzatura i membri dell'S&P approntarono un "impianto mostruosamente ingegnoso" che consentiva il controllo dei singoli treni in qualsiasi punto del plastico. Il sistema era "continuamente testato, riparato, perfezionato e talvolta 'gronked', cioè, nel gergo del club, scassato. I membri dell'S&P erano ossessionati dal modo in cui 'il sistema' funzionava, dalla sua crescente complessità, dal modo in cui avrebbero reagito le altre parti a qualsiasi cambiamento, dal come usare quelle connessioni tra le parti per ottenere il massimo"[6].
Capo dell'S&P era uno studente anziano, Bob Saunders. Nel club, si distinsero studenti del MIT come Alan Kotok, Peter Samson, e molti altri, ma si trovavano anche giovani geniali come Peter Deutsch che, in virtù delle sue conoscenze informatiche, fu accettato alla pari a soli 12 anni.
"I membri anziani stavano al club per ore, discutendo sul da farsi, sviluppando un gergo esclusivo, incomprensibile per gli estranei: un progetto intrapreso o un prodotto costruito non soltanto per adempiere a uno scopo specifico ma che portasse con sé il piacere scatenato della pura partecipazione, era detto 'hack'. Quest'ultima parola proveniva dal vecchio gergo del MIT: il termine 'hack' era stato a lungo usato per indicare gli scherzi elaborati che gli studenti del MIT s'inventavano regolarmente"[7]. Un impresa era definita vero hack se mostrava innovazione, stile e virtuosismo tecnico. "I più produttivi [...] si definivano, con grande orgoglio, 'hacker'"[8].
Gli interessi principali degli hacker erano l'atto stesso della programmazione ed i segreti del funzionamento delle macchine. L'IBM 706 e gli altri computer di prima generazione che lo sostituirono avevano uno spazio esiguo nella piccola memoria, migliorare i programmi riducendo al massimo le istruzioni era un imperativo. E anche se, nel tempo le memorie divennero più capienti, l'esigenza di risparmio di istruzioni rimase: rimase cristallizzata come principio ideale.
Per i discepoli dell'hands-on imperative, caposaldo dell'hacking, era ovvio cercare di entrare in contatto diretto con le macchine per metterci su le mani. Ma, le restrizioni e la proliferazione di regole, pensate proprio per tenerli fisicamente lontani da quelle macchine, era per loro soffocante.
Le cose migliorarono quando al MIT arrivò un nuovo computer, il Tx0, uno dei primi funzionanti a transistor, intorno al quale le restrizioni erano minori.

La comunità hacker si stava consolidando intorno al Tx-0 e gradualmente stava sviluppando lo stile di vita, l'etica, la filosofia ed il nuovo linguaggio, unici e separati che la caratterizzano. "Non ci fu un momento preciso in cui gli hacker del Tx-0 intravidero che votando le loro abilità tecniche nell'informatica, insieme a una dedizione raramente riscontrata al di fuori dei monasteri, sarebbero divenuti l'avanguardia di un'audace simbiosi tra uomo e macchina. [...] Mentre si andavano formando gli elementi di una cultura, cominciavano ad accumularsi leggende [...] Comunque, questo gruppetto di hacker era restio ad ammettere che la loro piccola comunità, in stretta connessione con il Tx-0, avesse lentamente e inavvertitamente costruito un corpo organico di concetti, convinzioni e costumi"[9].
I precetti di questa rivoluzionaria etica hacker non erano scritti né manifesti, ma tacitamente accettati. Ed era il computer stesso ad operare le conversioni.

1.1.2 La prima generazione di hacker negli anni '60

Nel 1961 la Dec (Digital equipment corporation), regalò al MIT una nuova macchina. Il Pdp1 sembrava fatto apposta per gli hacker; di piccole dimensioni rispetto ai predecessori[10], non richiedeva complesse procedure o particolari precauzioni ed accorgimenti per il suo funzionamento, aveva uno schermo più facile da programmare ed il suo costo al dettaglio, di 120.000 dollari, era abbastanza basso da mettere a tacere coloro che si lamentavano dell'uso 'improprio' fatto dagli hacker che consumavano preziosi minuti di tempo operativo. Inoltre alla Dec avevano una mentalità lontana da quella burocratica dell'IBM e sembrava che i suoi ricercatori osservassero lo stesso stile della comunità del Tx-0: informale, interattiva, capace di recepire senza formalismi le nuove idee.
Il set di istruzioni del nuovo 'minicomputer' non era molto diverso da quello del Tx0 per cui cominciarono a 'portare' i loro programmi da una macchina all'altra e a scriverne di nuovi ancor prima che arrivasse.
Il Pdp-1 fu consegnato con un piccolo corredo di software, del tutto inadatto per gli hacker. Per Kotok l'assemblatore[11] era pessimo. Proposero a Dannis di riscriverlo in un week-end e se fossero riusciti sarebbero stati anche pagati. "Quando Jack Dennis arrivò quel lunedì mattina rimase sbalordito nel trovare un assemblatore installato sul Pdp-1 [...] avevano, in un week-end, tirato fuori un programma che sarebbe costato, all'industria del computer, settimane o forse mesi di duro lavoro. Era un progetto che probabilmente quest'ultima non avrebbe intrapreso senza una lunga e tediosa procedura di domande scritte, studi, incontri, esitazioni burocratiche e molto probabilmente anche notevoli compromessi durante l'attuazione".[12]

Gli hacker tenevano i nastri perforati dei programmi in un cassetto in modo che chiunque potesse disporne per cercare di migliorarli, tagliare via un po' di istruzioni inutili o aggiungervi nuove caratteristiche operative.
Samson stava hackerando un programma musicale per il Tx-0. La Dec ne era al corrente, così gli chiese di implementarlo[13] sul Pdp-1 (con migliori capacità audio). "Furono onorati quando la Dec richiese il programma per offrirlo agli altri proprietari di Pdp1, e la questione sui diritti d'autore non fu mai sollevata"[14]. Un buon software era inteso come un dono per il mondo e per la comunità che vi nasceva intorno. Inconcepibile considerarlo una merce da pagare. La Dec in cambio del software che riceveva in dono, concedeva ciò che gli hacker chiedevano per il loro lavoro: pezzi di ricambio, specifiche tecniche, informazioni di ogni genere, senza nessun tipo di formalità.
Altre organizzazioni di hacker, dopo il Tmrc, cominciavano a nascere come l'Higham institute[15]. Per dimostrare, che la programmazione di un computer non era solo mera ricerca tecnica, avviarono la programmazione del primo videogioco della storia: Spacewar.
"Slug Russell sapeva che mostrando una versione approssimativa del gioco e lasciando cadere il nastro di carta nella scatola con i programmi di sistema del Pdp-1, avrebbe invogliato a compiere dei non sollecitati miglioramenti"[16]. Così avvenne e il lavoro di gruppo, fase dopo fase perfezionò il programma. L'evoluzione del gioco avvenne con ordine e senza problemi. Nessuna alterazione era fatta senza ottenere l'assenso degli altri. Questo perché "Le pressioni sociali che facevano rispettare l'etica hacker - che incitava a manipolare per migliorare, non per danneggiare - prevenivano ogni tentazione di combinare guai"[17].
Spacewar divenne popolarissimo[18] e venne distribuito gratuitamente alla Dec ed agli altri proprietari di Pdp-1.

Negli anni successivi, molti dei programmatori lasciarono l'istituto per lavorare presso le industrie o diretti verso altre università, portando l'hackeraggio stile MIT fuori da Cambridge[19]. Alcuni di loro si stavano affermando con mezzi 'tradizionali' come tesi o premi accademici, divenendo noti e stimati presso la comunità scientifica.
Gli accademici, chiamati planner, erano favorevoli a mettere i computer nelle mani del maggior numero possibile di ricercatori, statistici, scienziati e studenti. Ritenevano l'informatica qualcosa di positivo in sé e lavoravano per rendere più facile l'uso dei computer[20], perché se più gente li avesse usati, sarebbero emersi nuovi teorici ed esperti programmatori, e la scienza ne avrebbe tratto beneficio. I computer erano pochi e le liste di attesa per il loro utilizzo sempre piene, occorreva che più persone contemporaneamente potessero usare i computer. I planner promossero un gruppo di studio per realizzare l'utilizzo multiutente, denominato 'time-sharing' (a partizione di tempo), sistema in cui più periferiche fanno capo ad un elaboratore centrale. Il progetto, chiamato Progetto Mac (Multiple access computing: elaborazione ad accesso multiplo) ottenne un finanziamento del ministero della difesa, attraverso l'Arpa (Advanced research projects agency), di tre milioni di dollari l'anno. Dennis ottenne la direzione ma un terzo di quel finanziamento fu affidato a Minsky per il nuovo settore dell'intelligenza artificiale. All'inizio degli anni '60, Marvin Minsky, collega di McCarthy, cominciò ad organizzare il primo laboratorio di intelligenza artificiale al mondo. La realizzazione delle sue idee aveva bisogno di 'geni della programmazione' e per questo incoraggiò l'hackeraggio in ogni modo. Minsky e Dannis misero molti hacker a lavorare, stipendiati, al progetto Mac. Alcuni sulle teorie più astratte sull'intelligenza artificiale ma il difficile era come attuare i programmi che a loro volta avrebbero fatto quelle cose. Minsky lasciò agli hacker del progetto Mac questa incombenza dando loro totale libertà. Si dedicarono a problemi come braccia robotiche, progetti di visione artificiale, enigmi matematici, sistemi time-sharing, superando ogni immaginazione.

Dei nuovi studenti entrati all'inizio degli anni 60 si distinsero Richard Greenblatt e Bill Gosper. Rappresentavano le due visioni dell'hackeraggio del Tmrc e del Pdp-1: il primo, basato sulla costruzione di sistemi pragmatici ed il secondo sull'esplorazione matematica. Ognuno rispettava l'approccio dell'altro e spesso collaboravano insieme ai progetti, sfruttando le rispettive migliori qualità.
Greenblatt e Gosper fornirono grandi contributi alla cultura che stava fiorendo intorno ai computer del MIT, qui l'etica hacker raggiunse i suoi apici. I 'veri hacker' preferivano sempre più il computer del progetto Mac e l'ambiente del Tmrc "dove la gente s'incontrava a qualsiasi ora della notte discutendo di argomenti che, a un estraneo, sarebbero apparsi incredibilmente arcani"[21]. Le discussioni si vivacizzavano, intorno al tentativo di ipotizzare quale fosse la 'cosa giusta' (the right thing) da fare. "Il termine aveva un significato particolare per gli hacker. 'La cosa giusta' implicava che per qualsiasi problema, [...] esisteva una soluzione che era proprio... quella: l'algoritmo perfetto. [...] 'La cosa giusta' spiegò più tardi Gosper, 'molto spesso significava l'unica soluzione, la più corretta ed elegante... quella che soddisfava contemporaneamente tutti i diversi punti di vista e tutti i problemi'"[22].

Quando la Dec costruì il Pdp-6 diede il primo prototipo al progetto Mac. Era un prodotto rivolto all'utenza commerciale ma era anche nato con un notevole apporto al progetto da parte degli hacker per superare le limitazioni dei vecchi modelli. Il set di istruzioni era molto più efficiente e soddisfaceva qualunque richiesta.
Per gli hacker, il Pdp-6 ed il suo nuovo set di istruzioni, voleva anche dire avere un nuovo vocabolario con cui esprimere concetti e sentimenti che prima non potevano essere comunicati se non in modo approssimativo.
Minsky mise subito gli hacker a lavorare, per la stesura di software di sistema per il nuovo computer e i risultati non si fecero attendere.
Greenblatt hackerò un compilatore Lisp per far girare il linguaggio per l'intelligenza artificiale di John McCarthy.
Alcuni, pensavano che il Lisp sarebbe stato una perdita di tempo anche per il Pdp6[23], una delle follie di Minsky. Ma lo sviluppo cominciò ugualmente. Il MacLisp (chiamato così per il progetto Mac) sul Pdp-6, era considerato vitale per il settore dell'intelligenza artificiale. Gli hacker usarono quel linguaggio nei loro programmi per poi essere integrato nelle loro conversazioni[24].
La Dec era interessata al MacLisp, e Kotok concordò con Greenblatt e altri lavorare su un programma, batterlo nel loro codice e debuggarlo. "Faceva parte di un semplice accordo tra MIT e Dec e nessuno aveva niente da ridire. 'La cosa giusta' implicava l'assicurarsi che ogni buon programma ottenesse la diffusione più completa possibile"[25].

Nel '63 Stewart Nelson si aggiunse al gruppo degli hacker, prediligeva l'elettronica e fin da piccolo ne era un abile esploratore. Cominciò da matricola alla radio universitaria, per passare alle linee telefoniche ed al radiotrasmettitore del campus. Appena trovò il Pdp-1 si diede subito da fare; imparò a programmarlo di notte, quando i sorveglianti andavano a dormire, introducendosi di nascosto nel laboratorio.
Con il Pdp-1, per il puro gusto di farlo, 'esplorò' il sistema della compagnia dei telefoni trovando le particolari frequenze usate dalla compagnia per mandare le chiamate interne in giro per il mondo.
La notizia delle sue attività portò Nelson allo status di 'eroe' al Tmrc e al laboratorio del Pdp-1. Gli hacker più 'integralisti' biasimavano il suo comportamento: ritenevano che si fosse spinto troppo in là, ma sapevano anche che non ci sarebbe stato modo di fermarlo.
Con quel modo di fare e con la sua perseveranza, Nelson, amplificò l'etica hacker: "'Se tutti mettessimo in atto la nostra voglia di fare nuove scoperte, scopriremmo di più, produrremmo di più, terremmo sotto controllo molte più cose'. [...] Quando Nelson partiva per queste escursioni elettroniche, si atteneva ai principi non scritti della moralità hacker. 'Puoi chiamare dappertutto, cercare qualsiasi cosa, sperimentare di tutto senza limiti, ma non per lucro'"[26].
L'hackeraggio telefonico non era nuovo ai membri del Tmrc, ma l'etica hacker impediva di trarne profitto. Essi disapprovavano coloro che costruivano apparecchi (chiamati Blue-box) per fare telefonate gratis truffando le compagnie telefoniche.
Erano certi di aiutare le compagnie. Collaudavano le linee principali e, se trovavano dei problemi, li comunicavano alle compagnie fingendosi tecnici della Bell telephone. La compagnia scoprì le centinaia di telefonate 'esplorative' partite dal MIT, ma il tentativo di fermarli risultò inutile[27].
Mettere le mani sull'hardware (ad esempio per inserire un'istruzione per far fare al computer qualcosa di nuovo), per gli hacker era essenziale, come accedere alle macchine per la programmazione, però tassativamente vietato. Ogni modifica aveva bisogno di una lunga trafila burocratica che poteva durare dei mesi durante i quali le attività andavano sospese. Per loro era un inutile complicazione e i divieti dei burocrati erano 'dettagli'. Così, la notte, provvedevano da soli alle modifiche, sicuramente più 'veloce ed istruttivo', poi, verificavano che la macchina funzionasse e cancellavano le tracce del loro passaggio.
Ma, un programma non testato per la premura mandò il computer in crisi. Ci furono delle rimostranze ufficiali contro di loro. Gli effetti finali furono inattesi: "Minsky e gli altri impegnati nel progetto Mac sapevano che le attività notturne degli hacker si erano evolute in un corso pratico post laurea di progettazione logica e dell'hardware [...], la proibizione ufficiale dell'la lab di manomettere l'hardware gradualmente svanì [...]. [Marvin Minsky] sapeva che l'etica hacker era quel che rendeva produttivo il laboratorio, e non aveva alcuna intenzione di modificare uno dei componenti cruciali dell'hackeraggio"[28].

Nelson era il frutto dell'etica hacker, ed il suo comportamento contribuiva alla crescita culturale e scientifica dell'la lab. Scopriva sempre nuovi hack telefonici ed era leader nell'arte del 'lock hacking' (l'hackeraggio delle serrature).
Il lock hacking è l''abile superamento di blocchi fisici'. Questa pratica era una tradizione del MIT, soprattutto al Tmrc. Unito all'etica hacker, divenne più una crociata che un gioco.

"Per un hacker una porta chiusa è un insulto [...]. Proprio come il software dovrebbe essere distribuito senza limitazioni, gli hacker credevano che anche le persone dovessero avere libero accesso ai file o alle apparecchiature che avrebbero potuto promuovere la ricerca degli hacker verso nuove forme di conoscenza del mondo. [...] Le serrature simboleggiavano il potere della burocrazia, un potere che sarebbe stato concretamente impiegato per impedire una piena implementazione dell'etica hacker"[29].
Se avevano bisogno di un pezzo di ricambio o di uno strumento, sotto chiave da qualche parte, facevano escursioni notturne e lo andavano a prendere. Smontavano le serrature, falsificavano le chiavi[30], si infilavano negli uffici calandosi dallo spazio del controsoffitto, poi prendevano i pezzi, ripristinavano i computer, rimettevano tutto a posto con cura e tornavano al lavoro.
Il lock hacking era un incubo per chi amministrava l'la lab. Gli hacker entravano dove volevano. "Erigere barriere avrebbe significato elevare il livello della sfida [...] Così non rimaneva che stabilire un tacito accordo, per cui esisteva 'questa linea, ovviamente immaginaria', [...] se qualcuno violava quei limiti, la violazione sarebbe stata tollerata finché qualcuno non ne fosse venuto a conoscenza"[31]. In questo modo, l'amministrazione poteva mantenere una certa dignità, mentre gli hacker potevano fingere che l'amministrazione non esistesse proprio.

L'la lab del MIT, alla fine degli anni '60, era una 'comunità modello', che basava il suo impegno sulle regole dell'etica hacker. Presupposto era la cooperazione su bese volontaria: sentivano l'hackeraggio come una missione comune.

Nel 1966 arrivò all'la lab David Silver, figlio quattordicenne di uno scienziato del MIT[32]. Con un terminale installato nell'ufficio del padre, connesso al Compatible time-sharing system (Ctss) dell'IBM 7094[33], cominciò a scrivere programmi in Lisp.
Una importante attività dell'la lab, era la robotica; gradita agli hacker perché consentiva di costruire e sperimentare direttamente.
Anche Silver amava la robotica. Costruì un piccolo robot e scrisse in linguaggio macchina un programma per farlo muovere su ruote.
Minsky aveva permesso a Silver di frequentare il Tech Square e, senza badare all'età, gli hacker lo accolsero per il contributo che poteva dare.
"Silver considerava gli hacker come suoi maestri: poteva chieder loro qualsiasi cosa riguardante i computer o le macchine, e loro riversavano su di lui enormi schegge di conoscenza. Gli venivano trasmessi nel gergo colorito degli hacker, carico di strane variazioni [...] sulla lingua inglese"[34].
David Silver voleva che il suo robot usasse la telecamera per 'andare a prendere' oggetti gettati a terra. Silver usò le conoscenze degli hacker, facendosi spiegare singole cose per poi ricomporle, nel suo programma per 'vedere'[35]. Quel che mise insieme consentiva al suo robot di fare veramente quello che lui aveva previsto.
Silver, però si era attirato un mucchio di critiche, soprattutto da coloro che ritenevano il metodo degli hacker come non scientifico.
Minsky difese il ragazzo e lo tenne al laboratorio. In seguito la tensione calò. Ma la contesa con gli hacker restava sullo sfondo. I non hacker (laureandi, ricercatori, ecc.) li giudicavano un gruppo di tecnici sconsiderati ma necessari. Gli hacker pensavano che quelli fossero solo dei presuntuosi.

Parte importante del progetto Mac era il sistema time-sharing. Il primo disponibile era il Ctss, gli hacker lo disprezzavano perché rappresentava l'ideologia burocratica dell'IBM. "'Una delle cose più divertenti del computer è che puoi esercitare un controllo assoluto su di esso. Quando si mette in mezzo la burocrazia, non hai più il controllo della macchina"[36].
Gli hacker usavano il Ctss quando non c'era altro, ma occorreva la password, per loro più odiosa di una porta sbarrata. Approfondirono la conoscenza del Ctss e 'scardinarono' il problema.
Alla luce della tendenza alla guerriglia degli hacker[37], i progettisti dell'la lab dovevano evitare d'entrare in collisione con la mentalità hacker, ma intorno al 1967, vollero convertire il Pdp-6 in una macchina time-sharing.
Il time-sharing era ormai inevitabile. Tra gli hacker da una parte e gli utenti ufficialmente registrati dall'altra, le attese erano sempre più lunghe e il Pdp-6 al centro di aspre contese. Gli hacker non accettavano il time-sharing: più lento, meno potente. I programmi voluminosi non avrebbero potuto più girare. L'idea stessa di non poter controllare l'intera macchina li disturbava.
Si giunse ad un compromesso. La macchina poteva girare in modalità monoutente a tarda notte, così gli hacker avrebbero potuto lanciare i loro giganteschi programmi e il Pdp-6 sarebbe stato a loro completa disposizione.

L'amministrazione approvò il progetto, dando agli hacker piena autorità sulla gestione dell'evoluzione del sistema, e concesse anche nuova memoria per il Pdp-6[38].
Il Kernel[39] fu scritto da Greenblatt e Nelson in linguaggio macchina, per avere maggior controllo e più velocità, altri aggiunsero accorgimenti e nuove funzionalità e tutti si adoperarono per il debugging. Ogni cosa andò benissimo.
Fu chiamato ironicamente 'Incompatible time-sharing system' (Its): era più compatibile di qualsiasi predecessore[40].
Il sistema incorporava l'etica hacker: non usava password e l'accesso a qualsiasi file archiviato su disco era libero, avevano esteso e adattato all'Its l'usanza dei nastri di carta dei programmi nel cassetto, per poter usare e migliorare il lavoro altrui. Chi partecipava, era benvenuto. "La struttura aperta dell'Its incoraggiava gli utenti a guardare questi file [...]. Si poteva viaggiare tra i programmi dei più grandi hacker, cercare delle idee, ammirare il loro codice. [...] L'Its insomma, conservò il senso di comunità che gli hacker provavano quando c'era un solo utente per macchina e le persone si affollavano intorno a lui per guardarlo macinare codice"[41].
Come ogni altro progetto messo in pratica dagli hacker utilizzava un procedimento circolare, in cui il programmatore è anche un utente abituale del sistema o del programma che sta migliorando. Per loro nessuna opera è mai completata, la si può sempre migliorare. "Il sistema Its non è il risultato di un progetto con grandi risorse umane [...] è stato sviluppato progressivamente in maniera pressoché continua [...] si può dire che l'Its sia stato implementato dal progettista e progettato dall'utente. Il problema di progettare del software non realistico si riduce grandemente quando il progettista è un utente. [...] Le caratteristiche introdotte saranno difficilmente di scarsa utilità se gli utenti sono stati i progettisti e difficilmente saranno complicate da usare se progettate dagli utenti"[42].
L'Its era la dimostrazione dell'efficacia dell'hacking, della cooperazione e della condivisione del software, dove i programmi non appartengono all'autore, ma a tutti gli utenti della macchina.

Questo 'periodo d'oro dell'hackeraggio', di isolamento quasi monastico dal 'mondo reale', era destinato a finire di lì a poco.

Alla fine degli anni '60 i computer erano visti da molti giovani come uno strumento malvagio, al servizio dei potenti. La gente non faceva distinzione tra programmatori: li vedeva come 'scienziati pazzi' o cospiratori al servizio di un Grande Fratello orwelliano.
Gran parte degli hacker non badava a come la gente li percepiva, ma col '68 e la guerra in Vietnam, combattuta anche coi computer, al MIT dovettero fare i conti col mondo reale.
Le attività dell'Ia lab erano finanziate dal ministero della difesa attraverso l'Arpa[43]; nessuno aveva mai chiesto loro applicazioni specifiche, ma dalle loro attività venivano ricavate anche applicazioni militari.
"I planner pensavano di lavorare per il progresso della vera scienza. Gli hacker stavano spensieratamente formulando la loro sincera filosofia della nuova era, basata sul libero flusso delle informazioni, il decentramento e la democrazia del computer"[44]. Ebbero un risveglio scioccante quando venne annunciata una manifestazione che doveva terminare con un sit-in dentro il Tech Square. Scossi dalla possibilità di tumulti, accettarono serrature, porte blindate e regole di sicurezza, come gli elenchi restrittivi di coloro che potevano accedere all'edificio.
L'accusa dei pacifisti era di collaborare alla guerra e che il loro idealismo fosse tutto un inganno. La marcia di protesta rimase all'esterno. Ma quando tutto si calmò, le serrature rimasero. Il sistema restava senza segreti anche se protetto fisicamente tenendolo sotto chiave. Gli hacker decisero che la nuova situazione non modificava lo spirito originario.

Alla fine degli anni sessanta, fuori da Cambridge, l'informatica si stava diffondendo presso sempre più università e centri di ricerca pubblici e privati. Le macchine si riducevano di dimensioni e di prezzo, erano sempre più interattive e proliferavano ambienti di programmazione facilitati.
Molti hacker della prima generazione avevano lasciato il lab del MIT esportando in giro per il paese cultura e stile di programmazione, alcuni diretti verso altre istituzioni universitarie, dalla Stanford alla Carnegie-Mellon, altri diretti verso industrie private. Quelli tra loro col senso degli affari ne avevano fondate di proprie e costituivano un forte richiamo per i talenti del Tech Square.
La destinazione più gradita era il centro di calcolo dell'università Stanford, il Sail (Stanford Ia lab), che John McCarthy aveva fondato nel 1962.
Il Sail per molti aspetti somigliava al MIT, ma vi erano anche differenze significative. Alla Stanford la sede, non era un cubo di cemento che dava un senso di claustrofobia, ma un edificio dall'architettura ospitale. Sessantaquattro terminali sparsi nei vari uffici evitavano frustranti attese e 'guerre di religione' tra laureandi e hacker. Questi ultimi, vivevano secondo i principi dell'etica hacker, ma il clima era più rilassato e non si

comportavano da eremiti integralisti[45]. Trovavano il tempo per dedicarsi ad altre attività, oltre all'hacking (comunque intenso).
Il Sail non era inferiore al MIT. I progetti di ricerca, nei campi della robotica, della comprensione della lingua e in quello del linguaggio naturale o della musica digitale, erano molto attivi e aprivano nuove importanti prospettive in quei settori. Come l'Ia lab del MIT era aperto agli esterni, che potevano sedersi alla consolle e cominciare a fare hacking; se fossero sembrati promettenti, McCarthy li avrebbe assunti.

Un altro luogo che ha costituito un punto cardine per l'evoluzione della cultura hacker fu lo Xerox PARC, il famoso centro di ricerche di Palo Alto. Negli anni che corrono tra l'inizio degli anni '70 e la metà degli anni '80, il PARC diede vita ad una notevole quantità di realizzazioni innovative sia hardware sia software. Le attuali interfacce software costituite da mouse, finestre ed icone, le moderne stampanti laser e le prime LAN (Local Area Network)[46], nacquero proprio allo Xerox PARC.

Alla fine del 1969 l'Arpa collegò primi quattro 'nodi' ARPAnet: l'UCLA (University of California Los Angeles), lo Stanford research institute (Sri), l'Università di Santa Barbara (UCSB) e l'Università dello Utah. L'embrione dell'attuale Internet era formato. Presto si collegarono i sistemi di computer dei laboratori di altre università e centri di ricerca.
La 'rete' fu concepita e realizzata con il contributo dei migliori hacker chiamati a partecipare al suo progetto. Nacque con l'etica hacker 'nel sangue'. Valori come quelli per cui i sistemi devono essere decentrati, o per cui deve essere incoraggiata l'esplorazione o sollecitato il libero flusso delle informazioni, vennero confermati e diffusi, passando da un centro di calcolo all'altro. Fu un cambiamento importante per le comunità americane che presto avrebbe coinvolto il resto del mondo.
Gli hacker estesero i loro contatti in tutto il paese, scambiandosi impressionanti quantità di posta elettronica, barattando tecnologia, collaborando a progetti, formando gruppi di amici che non si sarebbero mai incontrati di persona, oppure, mantenendo rapporti con ex colleghi dislocati altrove. Il contatto aiutava a uniformare i comportamenti e l'etica hacker cresceva in rapporto al numero dei suoi aderenti.

La tradizione dell'erede del Pdp-6, il '10', stava per finire. Nel 1969, un hacker dei laboratori Bell, Ken Thompson, inventò il sistema operativo UNIX. Sino ad allora, i sistemi operativi, erano scritti in linguaggio macchina o di basso livello per avere maggior controllo e velocità, ma Thompson insieme a Dennis Ritchie, l'inventore del linguaggio C[47], capirono che l'hardware di nuova generazione e i compilatori disponibili erano evoluti in modo tale da poter usare C per un intero sistema operativo. Unix fu scritto completamente in linguaggio C.
La novità senza precedenti era che poteva essere installato su macchine di diverso tipo quando prima ogni macchina doveva avere sistema operativo e software scritto appositamente. Con Unix gli utenti non dovevano più far riprogettare (e pagare) il software ogni volta che le macchine venivano sostituite perché diventate obsolete e, con la stessa interfaccia e le stesse funzionalità, non dovevano ogni volta riacquisire le capacità di usarlo.
La diffusione dello UNIX in AT&T fu estremamente rapida ed entro il 1980 l'uso dello UNIX era stato tramandato a tutte le università statunitensi.
Fuori da Cambridge l'hackeraggio si sviluppava con principi molto simili a quelli degli hacker del MIT, ma evitando eccessi utopistici che rimasero lì confinati.
Al MIT pensavano che le cose non stessero andando nel modo che ritenevano ideale, ma un altro evento, dopo la manifestazione dei pacifisti, li riportò un po' più verso la realtà: l'Arpa, in conformità ad un emendamento approvato dal Congresso, dovette giustificare ogni progetto che avviava. I finanziamenti senza limite erano scomparsi e non c'era più denaro per stipendiare nuovi hacker. Lo stesso MIT passò ad un tipo di formazione più convenzionale sui computer.
La comunità sembrava sgretolarsi quando arrivò una seconda ondata di hacker.

1.1.3 La seconda generazione di hacker negli anni '70.

La seconda generazione, fondata sulle basi culturali degli hacker del MIT e fedele alla sua etica, era però convinta che fosse necessario divulgare questo spirito al di là di ogni confine. Il modo scelto era di diffondere il più possibile i computer tra la gente. Per farlo, i computer dovevano diventare più piccoli ed economici e prodotti in grande numero, così avrebbero potuto 'cambiare il mondo'.

Il primo tentativo di portare i computer tra la gente (che in quegli anni li guardava come oggetti disumani, inflessibili o come strumenti di guerra), fu realizzato nell'agosto del 1973, attraverso il Community memory

project. Uno dei fondatori era Lee Felsenstein, che apparteneva alla nuova categoria degli hacker dell'hardware.
Il Community memory era un gruppo di persone di Berkeley (California), di fanatici per la tecnologia determinati a "diffondere il sogno e l'etica hacker nelle strade per dar modo alla gente di scoprire il piacere di esplorare e di metterci su le mani"[48]. Installarono, perciò, un terminale connesso con un computer[49] al secondo piano di un grande magazzino, nell'area comune davanti ad un negozio di dischi molto frequentato e vicino alla bacheca dei musicisti: una parete completamente ricoperta di avvisi.
Il Community memory, al vecchio modo di organizzare incontri, ne affiancò uno nuovo: si immetteva l'inserzione e in pochi istanti si trovava la persona che faceva al caso.
Non ci volle molto perché la gente inventasse nuovi usi innovativi per il terminale.
In seguito fu installato un secondo terminale presso una biblioteca pubblica. Le macchine, però, erano ancora inaffidabili e bisognose di continua sorveglianza e riparazione. Il Community Memory, dopo un anno e mezzo, decise di sospendere l'esperimento: nonostante la sua popolarità era a corto di energie umane e di finanziamenti.

Un altro tentativo di diffondere i computer tra la gente della Baia di San Francisco, fu quello di Bob Albrecht: insegnava, con grande successo, informatica ai bambini ed ai ragazzi delle scuole. Fondò una casa editrice, la Dymax, ed ottenuto un Pdp-8 dalla Dec in cambio della stesura di un libro[50], realizzò un laboratorio itinerante (in un furgone) per portare il computer in giro per le scuole.
Nel suo ostinato tentativo di rendere popolari i computer, Albrecht, decise che occorreva una pubblicazione che ne facesse la cronaca e fosse un punto di riferimento di questo neonato movimento. Cominciò a pubblicare un piccolo giornale intitolato 'People's Computer Company' (Pcc). Sulla copertina del primo numero, dell'ottobre 1972, c'era la seguente didascalia:
"I COMPUTER SONO OGGI USATI CONTRO LA GENTE INVECE CHE A VANTAGGIO DELLA GENTE USATI PER CONTROLLARE LE PERSONE ANZICHÉ PER LIBERARLE. È TEMPO DI CAMBIARE TUTTO QUESTO: ABBIAMO BISOGNO DI UNA... PEOPLE'S COMPUTER COMPANY"[51].
Gli hacker dell'hardware scoprirono subito questa rivista e cominciarono ad usarla inviando suggerimenti, segnalando dove e come procurarsi pezzi di hardware e scambiando listati di programmi scritti in Basic [52]. La sede della Dymax e Pcc era situata in un piccolo centro commerciale. A disposizione del pubblico c'era un Pdp-8 con agganciati dei terminali. Uno di questi era collegato tramite linea telefonica ad un computer della Hewlett-Packard, accessibile nei momenti in cui restava inutilizzato. La sede era frequentata dalle persone più disparate e dai talenti informatici di tutta la zona.
Albrecht, inoltre, organizzava cene e pranzi sociali della Pcc, in cui la gente comune si riuniva unicamente per parlare di computer. A quelle cene si potevano incontrare gli esponenti locali della 'cultura alternativa' legata ai computer, come Felsenstein o Ted Nelson. Quest'ultimo aveva scritto e pubblicato in proprio un libro sulla 'controcultura del computer' che ebbe un discreto successo di vendite.

Nel 1974 Felsenstein si trasferì in un monolocale sopra un garage a Berkeley. Voleva realizzare il suo sogno di portare i computer tra la gente. Il progetto era quello di costruire un terminale secondo la concezione del Community memory. Voleva un computer che la gente comune potesse comprendere ed essere attratta dalla voglia di 'metterci su le mani', cambiare pezzi da soli e migliorarlo. Sperava che la diffusione del computer avrebbe portato con se la diffusione dell'etica hacker nella società.
Per sopravvivere trovò lavoro, come ingegnere a contratto, in una piccola azienda, la Systems concepts dove lavoravano molti veterani del MIT. Felsenstein era diffidente nei loro confronti, non sopportava il loro eccesso di purezza e l'avversione a far conoscere la tecnologia mettendola alla portata della gente comune. Dopo un ennesimo 'scontro tecnologico', se ne andò. La recessione rendeva difficile trovare un nuovo lavoro ma, un po' più a sud di Berkeley, stava nascendo la Silicon Valley.
Nella Bay Area quelli come lui, amanti dei circuiti e dei componenti elettronici, erano a centinaia, spesso costruivano in casa apparecchiature con pezzi riciclati, recuperati dai fornitissimi rivenditori di rottami e merci da fallimento della zona o con i nuovi prodotti che via via erano disponibili. Amavano l'atto in sé di costruire. Molti di loro avrebbero voluto costruire un computer, non per farci qualcosa di particolare, solo per provare, per il piacere di farlo e di giocarci.

Nel 1975, i cinquecentomila abbonati a 'Popular Electronics' ricevettero il numero di gennaio che pubblicizzava in copertina un computer in kit di montaggio al prezzo base di 397 dollari. Lo vendeva la Mits (Model instrumentation telemetry systems)[53] di Ed Roberts. I computer, costruiti intorno ai nuovi microprocessori della Intel erano venduti in kit. L'articolo su Popular Electronics descriveva il computer Altair[54] della Mits: 256 byte di memoria, senza input ed output. Unica possibilità di interazione erano degli interruttori sul pannello frontale con cui inserire informazioni nella memoria e le luci intermittenti con

cui 'comunicava'. Nonostante tutta la sua limitatezza, era un computer; per gli hacker dell'hardware era abbastanza per cominciare a darsi da fare con l'ingegno.
La possibilità di costruire un computer con quei chip era di pubblico dominio ma i 'big boys' dell'industria come IBM la consideravano un'assurdità.
Per Roberts, anche solo poche centinaia di acquirenti nel primo anno, avrebbero risollevato la situazione. Appena la rivista fu nelle mani degli hobbisti, fu sommerso da un'inimmaginabile mole di richieste e molti, nell'ordine postale, avevano già incluso il denaro.
La Mits non era pronta e ci volle del tempo per cominciare ad evadere gli ordini, alcuni clienti attesero anche più di un anno, inoltre il kit era solo l'insieme dei pezzi, stava al cliente capire cosa farne e non era facile metterlo insieme[55]. Una volta assemblato non ci si poteva fare molto: "Si poteva immettere un programma solo digitando numeri ottali attraverso quei piccoli interruttori che facevano a brandelli le dita e si poteva intuire la risposta al problema solo attraverso il lampeggiare delle luci led, anch'esse programmate in ottale"[56].
Il vero valore dell'Altair, era la possibilità di avere un computer personale e a basso prezzo
Il primo numero di quell'anno, della rivista della Pcc, conteneva un lungo articolo sull'Altair e rimandava a quello su Popular Electronics. Quando arrivò loro un prototipo già assemblato, fu subito aperto e si cominciò ragionare per tirarci fuori un vero sistema. Agli incontri del club, l'Altair era diventato l'argomento dominante.
Tra i frequentatori del Pcc, Fred Moore, che già insegnava presso il club l'uso dei computer, propose ad Albrecht di avviare un corso di hardware per venire incontro a quelli come lui, con scarsa conoscenza dell'hardware, ma desiderosi di imparare.
Il rifiuto di Albrecht non lo fermò. Fred Moore era un idealista interessato alla capacità dei computer di tenere insieme le persone e credeva profondamente nella cooperazione: bastava che la gente si mettesse insieme, comunicando e condividendo esperienze e conoscenze per risolvere i problemi. Per Fred era arrivato il momento di mettere insieme le persone che amavano i computer e l'Altair, così, insieme a Gordon French, organizzò un nuovo gruppo: l'Homebrew[57] computer group.
All'incontro, nel garage di French, parteciparono 32 persone. Si sarebbero 'bootstrappati'[58] a vicenda all'hacking dell'hardware. Discussero di quello che avrebbero voluto in un club; le parole che ricorrevano di più tra i presenti erano 'cooperazione' e 'condivisione'. Il secondo incontro, si tenne allo Stanford Ai lab (quello di McCarthy). Alle riunioni, partecipavano sempre più persone, tanto che in pochi mesi erano diventate quasi trecento.
I soci portarono agli incontri le loro esperienze e le condivisero tra loro: si aiutarono a vicenda a costruire l'Altair e in seguito a fargli fare qualcosa. Si sentiva il bisogno di ampliare il sistema e di aggiungere delle periferiche.
Bob Marsh, il compagno di garage di Felsenstein, insieme ad un ingegnere di nome Ingram, decise di progettare e costruire delle schede di espansione. L'esigenza più urgente era quella di espandere la memoria dell'Altair. Non avevano denaro, usarono lo stesso sistema della Mits: Marsh annunciò il prodotto e appoggiato dall'entusiasmo (e dal denaro) degli hobbisti raccolse la somma necessaria per avviare l'attività della 'Processor technology'. Le schede che aveva promesso dovevano essere perfette: quelli dell'Homebrew le avrebbero analizzate in ogni dettaglio.
La Processor technology, doveva essere conosciuta per la qualità dei suoi prodotti. Per Marsh era importante per la sua stessa autostima. Quando finalmente fu pronta la scheda, incisa a mano sul fornello di cucina, non aveva un computer su cui provarla. A tarda notte chiamò un amico, un certo Dompier, membro del club, che benché terrorizzato, acconsentì alla prova della nuova scheda. Dompier considerava il suo Altair quasi come un figlio ma, rispettava l'etica hacker, che "subordina la proprietà e l'individualismo al bene comune".

Nonostante fossero geograficamente dispersi in una vasta area tra Sacramento e San José e non avessero una sede stabile, i soci dell'Homebrew formarono una comunità molto unita; era un sistema che valeva molto di più della somma delle sue parti una 'Sinergia'[59].
"Il crescente numero di membri dell'Homebrew che stavano progettando o regalando addirittura nuovi aggeggi, dai joystick ai dispositivi di i/o per gli Altair, usava il club come una fonte di idee, ma vi si rivolgeva anche per le prime ordinazioni o per eseguire dei beta test[60] per i prototipi"[61].
Tutti i membri dell'Homebrew avevano bisogno del contributo degli altri per poter utilizzare, prestandosele a vicenda, le poche attrezzature allora disponibili (gli Altair e tutto quello che erano riusciti a collegare) e testare così ciò che stavano sviluppando. Ma soprattutto avevano bisogno di informazioni, e nel club passavano liberamente. Con il 'Mapping', ognuno raccontava ciò che stava facendo e in base a questo si scambiavano aiuti, consigli specifiche tecniche, senza badare al fatto che tra loro ci fossero dei concorrenti [62] e senza remore nel rivelare notizie riservate ottenute nei modi più disparati dalle industrie

informatiche. In certi casi era vero e proprio spionaggio industriale, ma non lo percepivano assolutamente come reato, era piuttosto una particolare forma di innocuo 'pettegolezzo': tutte le informazioni dovevano essere diffuse, anche le più segrete.
Erano l'avanguardia di una nuova specie di hacker e di una nuova industria, quella dei computer da tavolo, che sarebbe stata profondamente influenzata dall'etica hacker.
Le grandi aziende, nel frattempo, continuavano a snobbare queste persone e questa nicchia di mercato.

Avevano creato un nuovo mercato economicamente molto promettente. Qui però, lo stile hacker, avrebbe lasciato il passo a concezioni meno altruistiche.
Fino a quel momento le informazioni circolavano liberamente, inoltre dal '76 nacquero varie nuove pubblicazioni, che divennero canali di comunicazione tra gli hacker, dove scambiarsi programmi, 'pettegolezzi' e consigli tecnici. Le nuove compagnie in stile hacker le usavano per diffondere specifiche tecniche e schemi dei loro prodotti.
Ma il caso del Basic per l'Altair diede il senso della fragilità delle concezioni dell'etica hacker nel 'mondo reale'.
Nel 1975, la Mits aveva annunciato, una nuova versione del Basic per l'Altair. Per gli hobbisti sarebbe stato un notevole passo avanti nel semplificare l'uso della macchina. Ma non si riusciva ad avere.
Finché, ad un incontro promozionale della Mits tenutosi a Palo Alto[63], dove l'Homebrew computer club si presentò in forze, videro finalmente un Basic 'girare' sull'Altair. Si potevano inserire i comandi ed ottenere subito le risposte: era la versione del Basic per l'Altair di Bill Gates e Paul Allen.

Molti hobbisti ritenevano che il prezzo del Basic fosse eccessivo e che la Mits fosse 'avida di denaro'. Avevano anche sentito dire che Gates e Allen, avevano scritto l'interprete usando un computer che apparteneva a un istituzione finanziata con denaro pubblico e conclusero che il programma apparteneva a tutti i contribuenti[64]. Qualcuno all'incontro aveva raccolto da terra il nastro con il Basic e sembrò giusto copiarlo. Tanto che praticamente era in libera circolazione ancor prima dell'uscita ufficiale.
Paul Allen e Bill Gates[65], che avevano venduto il loro Basic alla Mits in cambio di un guadagno per ogni copia venduta, scrissero agli hobbisti una lettera che venne pubblicata su diverse newsletter. Si intitolava 'Lettera aperta agli hobbisti', e vi si spiegava che questo 'furto' avrebbe allontanato i programmatori dallo scrivere per il mercato hobbistico. "Chi può permettersi di fare del lavoro professionale per nulla? Quale hobbista può mettere tre anni di tempo-uomo nella programmazione, trovando tutti i bug, documentando il suo prodotto e distribuirlo, il tutto gratuitamente?"[66].
La reazione della comunità hacker fu piuttosto vivace, ma il punto più importante era l'emergere del problema dei diritti d'autore. Quando gli hacker del MIT lasciavano nel cassetto il software che scrivevano, non esisteva ancora un mercato. Gli hacker restavano dell'idea, molto salda, che i programmi fossero di tutti. Ma, con i computer da tavolo che si stavano diffondendo, il software poteva essere un'attività assai remunerativa.
La risposta a questa vicenda fu un nuovo 'processo organico' per produrre un Basic di pubblico dominio. Promotore dell'iniziativa fu Bob Albrect aiutato da Dannis Allison, membro onorario del Pcc e professore di informatica alla Stanford. Allison scrisse un articolo in cui gettava le basi per il progetto di un Basic 'Tiny' (minuscolo), che venne pubblicato sulla rivista del Pcc, sotto la definizione di 'progetto partecipativo', apposta per stimolare la partecipazione. Tre settimane dopo la pubblicazione arrivarono le prime risposte e tra queste anche un 'Tiny Basic' già corretto e debuggato contenuto in 2K di memoria[67] scritto da due programmatori del Texas. Fu subito pubblicato su Pcc in formato sorgente e nel giro di poche settimane, cominciarono ad arrivare bug report e proposte per migliorarlo. Il successo dell'iniziativa fu tale che dovettero pubblicare prima un supplemento solo per il Tiny Basic, poi trasformato in una nuova rivista, dedicata solo al "software gratis o molto economico". La rivista venne chiamata 'Dr. Dobbs Journal' (Ddj). Jim Warren, il neo direttore, in una lettera in cui spiegava la ragion d'essere della rivista e l'approccio di pubblico dominio[68] al software diceva: "C'è un'altra via d'uscita ai problemi sollevati da Bill Gates nella sua furente lettera agli hobbisti del computer in merito al 'ladrocinio' di software. Quando sarà gratis, oppure così poco costoso che sarà più facile pagarlo che duplicarlo, non verrà più 'rubato'"[69].
Dato che erano stati messi in commercio altri computer hobbistici, in kit o già assemblati, anche Tom Pittman raccolse la sfida e decise di scrivere un Tiny Basic per questi[70]. Appena pronto lo vendette ad una compagnia, a patto di non esclusività, e decise di spedirlo a chi ne avesse fatto richiesta in cambio di soli 5 dollari, per dimostrare che la gente l'avrebbe comprato e non 'rubato'. "Mandò un annuncio alla rivista 'Byte' e, nel giro di qualche giorno, si ritrovò cinquanta dollari in casella. Alcuni [...] spedirono cinque dollari con un biglietto che diceva di non spedirgli niente, perché l'avevano già copiato"[71].

Al meeting dell'Homebrew tenne un discorso sulla sua esperienza; sottolineò le questioni sul 'free software'[72] e sulla libera circolazione dell'informazione. "Voleva parlare con la gente e mostrare un esempio col quale poter crescere".
Dal lato hardware proseguivano le sperimentazioni di computer da tavolo: l'idea era di riuscire a produrre e distribuire un computer completo, che avesse cioè anche un monitor, una tastiera[73] e una memoria un po' più capiente. Il primo con queste caratteristiche fu il 'Sol', prototipo costruito da un gruppo di membri dell'Homebrew usando parti commercializzate dalla Processor Tecnology di Bob Marsh.
Presto però arrivò il computer progettato da Steve Wozniak, che incarnò 'lo spirito e la sinergia' dell'Homebrew: l'Apple.
Wozniak progettava per puro diletto, voleva una macchina completa per poter giocare, divertirsi e da mostrare con orgoglio agli amici. Quando andava all'Homebrew "non incontrava alcun problema a ottenere dettagli tecnici [...] Era la fertile atmosfera dell'Homebrew che fece da guida a Steve Wozniak [...]. questi incentivi potevano soltanto aumentare il desiderio già intenso che Steve Wozniak aveva: costruire il genere di computer con cui avrebbe voluto giocare"[74].
Nel suo computer inserì la grafica a colori, e dato che era disponibile una scheda progettata da un altro membro del club, non esitò ad utilizzarla, voleva un Basic 'grande', così ne scrisse uno suo[75]. Wozniak progettava cercando di utilizzare meno pezzi possibile, così come gli hacker del MIT avevano l'ossessione della semplificazione del codice, lui cercava di utilizzare il minor numero possibile di risorse.
Steve Jobs, ex compagno dell'università di Berkeley, lo convinse non senza fatica, a mettersi in affari[76]. L'indirizzo ufficiale della Apple Company[77] era una casella postale, ma lavoravano in un garage. Pubblicarono degli annunci[78] sulle consuete riviste amatoriali e cominciarono a vendere l'Apple già completamente assemblato a poco più di 650 dollari. Le vendite da subito furono consistenti e 'Woz' si mise direttamente al lavoro all'espansione della sua scheda, per costruire l'AppleII, dall'architettura 'aperta' che avrebbe facilitato i fabbricanti di schede di espansione. Coerentemente all'etica hacker tutto era senza segreti, documentato e disponibile per chi ne fosse interessato. Come hacker, non erano in grado di gestire un'azienda, che nelle loro mani, pareva più una 'congrega di hippy e liceali', non commisero però l'errore di molti che li avevano preceduti e delegarono tutto ciò che riguardava il management[79].
Per Wozniak lasciare il lavoro e fondare un'azienda fu come 'passare il confine'. Non era più un hacker puro e sentiva molto profondamente la differenza tra progettare semplicemente per il piacere di farlo e progettare per 'far soldi'.
Il 'potere dei computer' si stava diffondendo per merito del mercato. La crescita era andata al di la di qualsiasi previsione e molti altri avviarono aziende per sfruttare il boom dei computer domestici[80].
Gli hacker dell'hardware, di fronte al bivio, scelsero la via degli affari; i pochi che preferirono restare 'puri' rimasero isolati lavorando soli o in piccole comunità.
La speranza che con la diffusione dei computer si sarebbe anche diffuso il sogno hacker, si rivelò una pura utopia, non solo, entrarono in competizione tra loro, e conservare le quote guadagnate sul mercato, imponeva di mantenere il segreto su quello che stavano facendo; anche Wozniak che riteneva fondamentale la trasparenza, in parte riconobbe il segreto come necessario[81]. L'Homebrew perse il suo ruolo di luogo di libero scambio delle informazioni e di condivisione delle tecniche, inizialmente ritenuta 'cosa sacra'.
Un paio di anni dopo l'inizio del boom, quelli che non furono capaci di creare una vera amministrazione in grado di reggere il mercato, si dissolsero sotto il peso della loro stessa crescita.
Le macchine in kit, ideate dagli hacker erano scomparse. Alla 'gente comune' non interessava l'hackeraggio sull'hardware, reso ormai quasi impossibile dal segreto industriale. Ma a questo punto era pronto il terreno per una nuova generazione di hacker del software.

1.1.4 La terza generazione di hacker negli anni '80.

Gli hacker dell'hardware avevano liberato il computer, lo avevano reso personale e avevano generato una nuova industria, con l'Apple in prima fila. Ora che queste nuove macchine interattive cominciavano ad entrare nelle case, la gente cominciava a far fare loro qualche cosa; tra le tante possibilità, furono i giochi la 'Killer Application'[82].
La nuova generazione di hacker si dedicò proprio ai giochi e anche in questo caso alcuni di loro intrapresero la via degli affari.
Caso emblematico fu quello di Ken Williams e di sua moglie Roberta.
Ken era diventato programmatore per caso. Programmare per lui era semplicemente lavoro, ma durante uno degli ultimi incarichi da dipendente entrò in 'relazione' con la macchina: "divenne un tossicomane della programmazione pura. [...] Faceva dei tentativi con tre o quattro linguaggi, [...] affascinato dalle tecniche e

dalle strutture mentali richieste da ogni linguaggio. C'era il mondo intero dentro quel computer... un sistema di pensiero."[83].
Da tipico hacker antiburocratico, Ken Williams, non amava il lavoro da dipendente, così, nel '79 divenne consulente autonomo.
In casa possedeva un terminale collegato con un mainframe dell'IBM, una sera si accorse che dentro c'era installato il gioco Adventure[84] e convinse la moglie a giocarci. Benché lei non amasse né i computer né i giochi, rimase letteralmente rapita "'Non potevo proprio fermarmi. Cominciai a giocare e continuai a farlo. [...] Non volevo essere disturbata. Non volevo fermarmi."[85].
Nel gennaio del 1980 Ken comprò l'Apple. Per lui era un giocattolo e ritenerlo un vero computer gli sembrava 'ridicolo'. L'Apple, però, aveva delle qualità che i 'bestioni' non avevano: era interattivo, abbastanza potente ed avendolo a casa non dovevi condividerlo con nessuno. Roberta aveva acquistato i giochi tipo Adventure disponibili per quel PC, ma rimase talmente insoddisfatta, che decise di scriverne uno lei. Scrisse la sceneggiatura ed il marito tradusse il gioco in codice. A differenza degli altri fu scritto in linguaggio macchina per l'Apple, quindi più leggero e veloce. Ci misero anche le immagini[86] e questo diede al gioco un buon vantaggio.
Fondarono subito la Sierra On-line company per commercializzare il loro Adventure 'Mistery House' (oltre ad alcuni altri applicativi), prima nei negozi andando porta a porta, poi facendo pubblicità sulle riviste di settore. Il solo Adventure vendette decine di migliaia di copie. Nel 1982, la società di giochi On-line contava già settanta dipendenti e stava valutando la prospettiva di diventare una public company[87] che rendeva dieci milioni di dollari all'anno.

Quello dei computer era un settore in crescita nonostante il periodo di recessione. Nel 1982 molte società, nate dalla rivoluzione iniziata dagli hacker dell'hardware, stavano valutando se offrire azioni al pubblico. Gli hacker che le guidavano avevano ammesso il mercato nella loro etica, che per questo stava cambiando, e un'azione mal giudicata da posizioni 'integraliste' era stata poi accettata (anche se non da tutti), come qualcosa di gradevolmente inevitabile; apprezzavano l'etica hacker, ma anche la fama e gli assegni delle royalty.
In quegli anni il software per personal computer era molto poco, chi faceva hackeraggio, scriveva applicazioni utili (per la contabilità o la videoscrittura) ma soprattutto creava 'strumenti per fare strumenti' oppure giochi, poi faceva il giro dei negozi per vendere il proprio software. In seguito aprivano le loro aziende o venivano assunti da chi lo aveva già fatto. I migliori tra loro venivano chiamati 'Software Superstar' e stavano vertice della terza generazione di hacker.
Altri hacker che ebbero successo con le loro aziende, furono ad esempio quelli della Sirius software o della Brøderbund software, che insieme a quelli della On-line erano tra i più competitivi sul mercato per il 'mondo Apple'. Tra loro si conoscevano personalmente e tenevano costanti rapporti. Avevano stretto un tacito accordo di cooperazione, una sorta di 'confraternita'. Si scambiavano continue informazioni, suggerimenti per risolvere problemi di programmazione e si raccontavano cosa stavano facendo in modo da non produrre le stesse cose. I membri dei rispettivi staff, non solo si incontravano alle fiere commerciali, ma organizzavano feste a cui prendevano parte i dipendenti delle tre aziende. Tutto ciò andava anche a vantaggio dei loro clienti che, in questo modo, potevano disporre di una più vasta gamma di prodotti ben realizzati.
Agli inizi degli anni Ottanta, chi possedeva un computer, si imbatteva necessariamente nella mentalità hacker. Comprare software era avvolto in una strana atmosfera artigianale, che poteva quasi apparire illegale: bustine di plastica, grafica primitiva, etichette scritte a macchina e incollate a mano. E utilizzarne uno richiedeva un iniziale processo d'apprendimento, o di trovare qualche 'guru' per farsi dire come fare. Acquisiti i concetti di base, però, non era più fondamentale continuare ad apprendere e solo i 'contagiati' acquistavano riviste specializzate o si iscrivevano ad un club o scrivevano il proprio software se non ne trovavano di già pronto.
Tra i vari personal disponibili sul mercato, l'architettura aperta dell'Apple[88], conforme all'etica hacker, invogliava a 'metterci su le mani' e furono molti i 'contagiati', che si sarebbero presto uniti all'élite. Questa generazione non doveva contendere ad altri il tempo macchina in una scuola o università, il computer - veloce, interattivo, con la grafica e gli effetti sonori - lo avevano a casa. "Potevano mettere le mani sui computer ora, [...] e qualsiasi nozione avessero di hackeraggio, qualsiasi principio dell'etica hacker avessero fatto proprio, sarebbe stato determinato da un processo d'apprendimento che derivava dall'hackeraggio stesso"[89].

John Harris fu tra questi, un compagno di scuola più grande gli lasciava usare il suo computer (un Commodore Pet) per giocare e programmare i giochi. "'Ero ossessionato da ogni genere di giochi', confessa Harris. 'Immagino che fosse un problema mio!'"[90].

John, dopo il diploma, si iscrisse ad ingegneria e cominciò a lavorare come tecnico in una banca. Risparmiato il denaro necessario per un computer tutto suo, comprò un Atari800[91]. L'azienda Atari, però, era stata assorbita dalla Warner communication, la quale aveva sostituito l'impostazione aperta dei fondatori con la chiusura più totale. Le informazioni tecniche sull'800, le ebbe sottobanco dal commesso di un rivenditore. Cominciò a scrivere i giochi in Basic ma, per ottenere un risultato migliore, aveva bisogno di informazioni per usare l'assembly[92] direttamente dal produttore. Quelli dell'Atari respinsero ogni sua richiesta di aiuto. Doveva scoprire quei segreti, così, partendo dal modo di comportarsi del software e della macchina e con l'aiuto di un amico, risalì alle informazioni di cui aveva bisogno. A queste ne aggiunse altre, provenienti da manuali hardware che circolavano illegalmente. John si servì di queste istruzioni per scrivere giochi.
Era la sua unica passione: l'atto in sé di programmare giochi. E la sua ricompensa era il piacere di averli fatti. Si rese conto che, per farlo al meglio, gli occorreva un saldo legame con persone come lui, dei contatti, una comunità. Si associò ad un gruppo di utenti Atari. Rese più belli e veloci i giochi della loro raccolta "Mostrava il suo lavoro agli altri che ci provavano un gusto matto a giocarci; tutto il suo hacking diventava automaticamente di pubblico dominio: la proprietà era un concetto che non l'aveva mai sfiorato". Gli proposero di commercializzarli e "la reazione di Harris fu: 'Sicuro, perché no?' Era come dare via un gioco e farsi pure pagare".
Nell'81 Harris fu presentato a Ken Williams e poco dopo cominciò a lavorare per lui alla On-line[93]. Qui, tra gli altri, scrisse un gioco in stile Pac-Man[94]. Il programma era completamente diverso dall'originale, composto anche con parti elaborate per lavori precedenti, ma era pressoché identico nella grafica. Era un problema e il gioco doveva cambiare grafica per essere commercializzato.
In quel periodo le grandi aziende detentrici di copyright si stavano muovendo per contrastare la 'traduzione non autorizzata' per PC dei giochi a gettone. E l'Atari aveva speso milioni di dollari per i diritti di Pac-Man.
La nuova versione del gioco, chiamato Jawbreaker, per gli avvocati di Ken non avrebbe dovuto creare problemi con l'Atari.
Jawbreaker prima che la revisione fosse ultimata, cominciò a girare tra i gruppi di utenti. "Senza pensare affatto a quelle restrizioni estranee alla cultura hacker come per esempio i segreti industriali", John aveva dato il gioco ad 'alcuni personaggi' di un negozio di computer; "non ci vide nulla di strano nella loro richiesta di prestito di una copia del dischetto".
Una copia arrivò anche all'Atari. Scoperto l'autore, cominciarono a far pressioni perché non fosse commercializzato e per comprarne i diritti lasciando una percentuale sugli utili. Ma John, ancora arrabbiato per il rifiuto dell'assembly, non volle accordi. Non voleva il suo nome sui prodotti Atari[95].
Andarono in causa, e l'Atari perse all'udienza preliminare contro la On-line. Anche se il labirinto era rimasto praticamente identico, i personaggi erano nuovi e il codice di Jawbreaker, scritto da Harris era completamente originale rispetto a quello di Pac-Man. Così il giudice, riconoscendo che le semplici idee non possono essere messe sotto copyright, non bloccò la vendita di Jawbreaker. Comunque, alla fine, On-line e Atari si accordarono prima di giungere al processo[96].

Nel 1981 l'International business machine, introdusse il primo IBM 'PC'[97], il suo computer da tavolo. Molte tra le piccole industrie di computer scelsero la riconversione ancor prima della commercializzazione del 'PC' per non essere messe fuori mercato.
In questa occasione, l'IBM fece una scelta assolutamente diversa rispetto al passato: usò un'architettura del tutto aperta e non proprietaria. IBM, non solo incoraggiò altre aziende a scrivere software, ma ingaggiò società esterne per contribuire al progetto, aziende come la On-line[98] o la MicroSoft, diretta da Bill Gates che scrisse il sistema operativo per l'IBM, l'MS-DOS 1.0[99], che divenne insieme all'architettura del PC IBM quasi immediatamente lo standard di fatto per i decenni successivi. Questo consentì ad altri produttori, di sviluppare un'immensa varietà di accessori, ma anche interi computer 'compatibili'. "Sembrava quasi che l'IBM avesse studiato l'etica hacker e avesse deciso che, in questo caso, applicarla era una buona idea per gli affari. Ma non aveva alcuna intenzione di applicarla troppo. Considerava ancora la segretezza uno stile di vita".

La terza generazione di hacker, con la loro cultura di creativi inventori di giochi, era matura, ma il boom dei giochi e il fiume di denaro che ne seguì trasformò la situazione. Sempre più persone inesperte possedevano computer, limitandosi ad utilizzare il software senza domandarsi come fosse fatto. Circolavano molti validi programmi (non solo giochi) scritti dagli hacker e resi di pubblico dominio, che però la gente non conosceva. Ai negozianti chiedevano quelli visti nelle pubblicità.
In queste aziende divennero più importanti le 'strategie di vendita' della qualità hacker. La 'logica di mercato' cancellò il residuo di etica hacker che era rimasto e la mentalità burocratica prevalse su quella aperta, cooperativa e goliardica dei primi tempi. Molti hacker rimasero disgustati dalla situazione e smisero

di lavorare per loro. Gli hacker vennero sostituiti da programmatori professionali la cui attività fu pianificata dagli addetti al marketing ed incasellata nella gerarchia aziendale. Tra le aziende della 'confraternita' cessò la collaborazione, sostituita dal segreto e dalla volontà di farsi concorrenza. I nomi degli autori non furono più pubblicati sulle scatole, i giochi cominciarono ad essere protetti dalla copiatura e fu eliminata la possibilità di visionare il codice. "Gli editori di software chiamarono questo procedimento 'copy protection' [protezione dalla copia], ma una grossa percentuale dei veri hacker lo chiamerà guerra. Per l'etica hacker, cruciale è il fatto che i computer, per loro natura, non considerano l'informazione come una proprietà."[100]. D'altronde queste aziende avevano investito tutto nella produzione del software e sarebbero fallite senza queste 'misure antiestetiche' per preservare le quote di vendita. Gli hacker, da parte loro, considerarono lo scardinare le protezioni del software come una questione di principio, proprio come quelli del MIT penetravano nei sistemi o facevano 'lock hacking' per procurarsi attrezzi e pezzi di ricambio.

1.1.5 Verso la situazione odierna.

Nel 1980 si potevano distinguere tre culture hacker, molto simili ma costruite su diverse tecnologie: quella ARPAnet/Pdp-10 collegata con il linguaggio LISP, quella legata a UNIX ed al linguaggio C su calcolatori Pdp-11 ed il VAX della DEC e lo straordinario popolo di entusiasti dei neonati computer da tavolo.

Nel 1983 DEC cancellò la sua adesione al progetto PDP-10 per concentrarsi sul VAX e sul PDP-11. Intanto la tecnologia dei microprocessori e della Local Area Network (LAN) iniziarono a fare presa nel mondo hacker. Nel 1982 un gruppo di hacker Unix di Berkeley fondò la Sun Microsystems con l'intento di far girare Unix su sistemi attrezzati con i nuovi processori economici della Motorola.
Nel 1984 la AT&T cominciò ad essere venduta e il suo Unix divenne per la prima volta un prodotto commerciale.
La comunità hacker si divideva sulla rete Internet, o meglio su Usenet[101], in due gruppi: c'erano gli hacker che utilizzavano minicomputer o workstation con il sistema UNIX e hacker che facevano parte del disorganizzato gruppo degli appassionati di computer da tavolo. Nell'ambito della cultura hacker basata su UNIX vi era una grande rivalità tra i simpatizzanti della 'corrente' Berkeley e quelli dell'AT&T.

Alla fine dell'83, il mondo dell'hacking stava per essere investito dall'impresa di Richard Stallman.
Stallman arrivò all'Ia Lab del MIT, assunto come programmatore sistemista, nel '71[102]; era il posto ideale, come egli stesso ricorda: "'non c'erano ostacoli artificiosi, [...] cose come la burocrazia, la segretezza, il rifiuto della condivisione con le altre persone'. [...] Amava stare tra quelle persone per le quali l'hackeraggio era l'essenza della vita. [...] Poteva essere apprezzato per le sue capacità da hacker ed essere parte di una comunità, costruita intorno a uno occupazione magica". Trovò l'etica hacker vicina alle sue convinzioni personali, ne consolidò il rispetto e si impegnò attivamente per l'attuazione dei suoi principi. "Finì per concepire il laboratorio come l'incarnazione di quella filosofia, un anarchismo costruttivo che, come Stallman una volta scrisse in un file, 'non significa giustificare la legge homo homini lupus. La società americana è già improntata a questo principio che le sue leggi mantengono. Noi hacker desideriamo soppiantare quelle regole con un appello alla cooperazione costruttiva'".
Distribuiva i suoi programmi gratis e incoraggiava gli utenti ad aggiungere qualcosa, a migliorarli e personalizzarli senza limite. La sua opera più conosciuta, un programma di editing chiamato Emacs, divenne quasi l'editor di testi standard nei dipartimenti universitari di informatica. Lo cedeva a chiunque accettasse "di rendere disponibili tutte le estensioni apportate, in modo da collaborare al miglioramento di Emacs. 'Ho chiamato quest'accordo 'la Comune di Emacs' [...] Dato che lo condividevo, era loro dovere condividere'".
Nell'83 però, le cose erano profondamente cambiate rispetto al giorno del suo arrivo, gli hacker erano diventati una minoranza. Molti avevano lasciato il MIT per andare a lavorare nelle industrie (accettando i relativi compromessi) e i nuovi arrivati, non vedevano nulla di male nella proprietà dei programmi. Per Stallman era come una 'bestemmia' "'non credo che il software debba avere una proprietà' [...] 'questa pratica mina l'umanità dal profondo. Impedisce alla gente di ottenere il massimo vantaggio dall'esistenza dei programmi'". Anche la sua battaglia contro le password, per l'accesso alle risorse informatiche del laboratorio, era perduta: il ministero della difesa aveva minacciato di scollegare l'Ia Lab dalla rete ARPAnet se non fossero state rispettate le procedure di sicurezza[103].
Il tracollo definitivo si ebbe con la vicenda della macchina Lisp: un computer per hacker progettato apposta per far girare il linguaggio Lisp[104]. Il gruppo che gestiva il progetto, avviato da Greenblatt negli anni 70, era stipendiato dal MIT e lavorava saltuariamente all'Ia Lab per la macchina Lisp. Quest'ultima, però, aveva nuove e interessanti caratteristiche[105], strategiche per la corsa al primato tecnologico contro i

giapponesi, pertanto, alla fine degli anni settanta, l'Arpa cominciò a finanziare il progetto, che divenne autonomo, facendo arrivare sempre più denaro per la progettazione e costruzione.
Ma le caratteristiche della macchina Lisp erano molto attraenti anche dal punto di vista commerciale e Russell Noftsker, ex amministratore dell'Ia Lab, si propose di aiutare gli hacker a fondare una società per commercializzarla. Greenblatt voleva una società in stile hacker e anche il potere necessario perché rimanesse tale. Sul come gestire la società, si creò uno scisma. Da un lato quelli che accettavano un modello tradizionale di attività economica e dall'altro quelli che volevano mantenere il più puro possibile lo spirito hacker. Alla fine furono create due società la Symbolics, commerciale, e la Lmi (Lisp machine incorporated). La Symbolics assunse molti degli hacker del laboratorio, mentre gli altri lavoravano part?time per la Lmi. Questi ultimi, per non danneggiare il laboratorio, si dimisero quando la Symbolics sollevò la questione del conflitto di interesse.

Stallman, su posizioni ancor più radicali, non collaborò alla svolta commerciale e rimase al MIT. Disapprovava quella scelta: per lui il software doveva essere solo libero e distribuito gratuitamente.
Lo scisma era per lui segno del fallimento del laboratorio nel sostenere l'etica hacker. Memorizzò in seguito sul sistema del MIT: "È doloroso per me tornare con la memoria a quei tempi. Le persone rimaste al lab erano i professori, gli studenti e ricercatori non hacker che non sapevano come mantenere il sistema, o l'hardware, né volevano impararlo. Le macchine incominciavano a guastarsi e non venivano più aggiustate, a volte le buttavano addirittura via e non venivano mai fatti gli indispensabili aggiornamenti del software. I non hacker reagivano a tutto questo rivolgendosi ai sistemi commerciali, tirandosi dietro fascismo e contratti di concessione. Mi ero abituato a vagare per il lab, attraverso le stanze ora vuote, quando prima erano piene di gente e pensavo, 'Oh, povero Ia lab! Stai morendo e non posso salvarti'. Tutti si aspettavano che se altri hacker fossero stati educati, la Symbolics li avrebbe assunti e quindi non sembrava valere la pena di farlo... tutta quella cultura era stata cancellata"[106].
Quando poi la Symbolics, invece di condividere le conoscenze ed evitare la dispersione di energie per duplicazione degli sforzi, decise che i suoi nuovi prodotti sarebbero stati coperti da copyright, per Stallman fu 'guerra aperta'. Cominciò a collaborare indirettamente con la Lmi, che in fondo aveva cercato di non danneggiare il laboratorio, perché potesse beneficiare delle innovazioni introdotte dalla concorrente al sistema operativo su cui stava lavorando. La Symbolics, in accordo con il MIT, installava le sue novità sulle macchine del laboratorio. "Stallman poteva ricostruire passo passo ogni nuova caratteristica o correzione di un certo bug, poi rifletteva sulle modalità in cui il cambiamento era stato introdotto, lo analizzava e presentava il lavoro alla Lmi. [...] non poteva semplicemente duplicare i cambiamenti, doveva scovare modi nuovi e diversi per implementarli[107].". Non riteneva immorale copiare il software ma voleva evitare guai alla Lmi.
Stallman sapeva che quanto stava facendo era assolutamente inutile, "non aiutava il mondo a progredire". Decise di aspettare la fine del 1983 per prendere una qualche decisione.

Stallman si considerava ormai un sopravvissuto, l'ultimo 'vero hacker'. La trasformazione dell'Ia lab, "unico esempio che dimostrasse che era possibile avere un'istituzione anarchica e al tempo stesso molto importante", lo aveva disarmato. "Se una volta dicevo alla gente che era possibile non avere sicurezza su un computer senza che altri alla prima occasione cancellassero i file e senza che ci fossero capi a censurare quel che uno voleva fare, almeno potevo indicare l'Ia lab [...] senza quell'esempio nessuno mi crederà. [...] Potevamo considerarci un esempio per il resto del mondo. Ora che tutto questo è svanito, da dove ricominciare? 'Sono l'ultimo sopravvissuto di una civiltà scomparsa'".

Quello stesso anno Stallman si dimise dal laboratorio del MIT per avviare il suo progetto di scrivere una versione libera da qualsiasi copyright del sistema operativo Unix e distribuirlo a chiunque fosse interessato. "Una volta che il mio gruppo si fu sciolto, continuare come prima fu impossibile. Mi trovai di fronte a una difficile scelta morale. La scelta facile sarebbe stata quella di unirsi al mondo del software proprietario, firmando accordi di non-diffusione e promettendo di non aiutare i miei compagni hacker. [...] Ma sapevo che al termine della mia carriera mi sarei voltato a guardare indietro, avrei visto anni spesi a costruire muri per dividere le persone, e avrei compreso di aver contribuito a rendere il mondo peggiore"[108].
Stallman era determinato a tenere in vita, il più puro possibile, lo spirito hacker, ricreando altrove il modo di operare dell'Ia Lab, in modo che potesse essere d'esempio e che stimolasse "innumerevoli piccoli atti come il suo" che dessero vita nel 'mondo reale' all'etica hacker. Nacque così la Free Software Foundation ed il Progetto Gnu (che sta per Gnu's not Unix, Gnu non è Unix), oggi centro gravitazionale di tutto il mondo del Software Libero.

All'inizio stentò a prendere l'avvio, poi, lo sviluppo delle reti e la diffusione capillare di Internet, ha consentito a tutti coloro che la pensavano come lui di mantenersi in contatto e lavorare allo stesso grande progetto. Per scongiurare il pericolo di appropriazione e chiusura che poteva essere fatto col software di pubblico dominio, inventò la licenza GNU/GPL chiamata provocatoriamente 'Copyleft' o 'permesso d'autore', che consiste nel dare a chiunque il permesso di eseguire, copiare, modificare i programmi, e distribuirne versioni modificate, trasmettendo gli stessi diritti che così diventano diritti inalienabili. infine, la 'rivoluzione informatica', che ha portato il computer nelle case di un numero sempre crescente di persone e a costi sempre più bassi, ha dato la spinta decisiva all'affermazione della sua idea.

Il Kernel, elemento fondamentale, per il sistema operativo del progetto Gnu, nacque solo nel '91 ad opera di uno studente finlandese di nome Linus Torvalds, utilizzando un clone di Minix, un FreeUNIX funzionante su macchine Intel 386 e un kit di strumenti di sviluppo messi a disposizione dal progetto Gnu della Free Software Foundation. Fu chiamato Linux. Il rapido successo di Linux attrasse molti hacker della rete Internet verso il progetto Linux che rappresentava la prima versione di UNIX basata su sorgenti interamente liberi e ridistribuibili[109].
Oggi intorno a Linux, direttamente, come gli sviluppatori del kernel, o indirettamente, come ad esempio coloro che sviluppano gli applicativi, lavora volontariamente un numero di programmatori, difficile da stimare, ma che si aggira intorno alle centinaia di migliaia in tutto il mondo. A questi vanno aggiunti i dipendenti delle aziende che hanno deciso di utilizzare il software libero per il loro business, che contribuiscono anch'essi allo sviluppo di questi software e della comunità Linux e del Software Libero in generale.
Attualmente la metodologia degli hacker si sta propagando ad altri settori del sapere soprattutto grazie al Copyleft.

---

1 North, 1990; ed. it. 1994, pagg. 199.
2 La ricostruzione storica è finalizzata all'analisi successiva. I fatti riportati in questo paragrafo sono prevalentemente tratti da Levy, S., 'Hackers', 1984. Principale fonte di informazioni del citato libro è costituita da oltre un centinaio di interviste, condotte dallo stesso Steven Levy, tra il 1982 ed il 1983, e da consistente materiale documentale. Il paragrafo è integrato dalle seguenti altre fonti: Berra, A., Meo A.R., 2001; DiBona, C., Ockman, S., Stone, M. (a cura di), 1997; Hafner, K., Lion, M., 1996; AAVV, Il mondo del computer, 1987; http://www.attivissimo.net di Paolo Attivissimo.
3 A metà degli anni 40 si passa dai calcolatori elettromeccanici a quelli elettronici, a valvole termoioniche. Il primo elaboratore nella storia fu il Colossus nel 1943, prototipo inglese di Turing, Newman e Flowers, che rimase però a lungo segreto militare. Negli USA il primo fu l'ENIAC di Eckert e Mauchly, è circa mille volte più veloce dei suoi predecessori viene completato nel 1945 nel laboratorio dell'Università di Pennsylvania.
4 Il Linguaggio macchina è un linguaggio di programmazione basato su codici numerici, che corrispondono alle singole operazioni elementari eseguite dal computer. L'ideazione dei linguaggi simbolici (più vicini a quello umano) e dei relativi compilatori cioè i 'traduttori' da simbolico a macchina semplificò di molto questa operazione, ma i Real programmer preferivano continuare a programmare in linguaggio macchina.
5 La Western Electric, tramite piano di donazioni alle università.
6 Levy, S., 1984; 3° ed it., 1999, Pag. 22.
7 Levy, op. cit., pag. 23.
8 Ibid.
9 Levy, op. cit., pag. 39
10 Era grande come circa tre frigoriferi, una capacità di elaborazione più o meno come le odierne agende elettroniche ma con meno memoria.
11 Programma di servizio che traduce le istruzioni di un altro programma scritto in linguaggio simbolico (assemblativo) in codici del linguaggio macchina di un determinato processore.
12 Levy, op. cit., pag. 55.
13 Allestire, completare, perfezionare un programma seguendo una procedura che parte dallo studio per arrivare alla definitiva messa in opera.
14 Levy, op. cit., pag. 57.
15 Chiamata scherzosamente così perché si trovava in Higham street a Cambridge.
16 Levy, op. cit., pag. 62.

17 Levy, op. cit., pag. 64.
18 Per giocare meglio Kotok e Saunders inventarono il joystick mettendo insieme pezzi di scarto abbandonati nella sede del Tmrc.
19 Alan Kotok andò a lavorare alla Dec, con un ruolo di primo piano nella progettazione di un nuovo computer, il Pdp-6, versione molto migliorata del Pdp-1, continuando però a frequentare il Tmrc e i laboratori informatici del MIT. Molte decisioni significative per il Pdp-6 vennero prese durante le sedute d'incontro tra hacker a cui Kotok prendeva parte quando il Pdp-6 era in fase di progettazione.
20 Ad esempio con linguaggi di programmazione molto più semplici come il Basic ideato da John Kemeny della Dartmouth.
21 Levy, op. cit.
22 Levy, op. cit.
23 Il giovane Peter Deutsch aveva scritto un Lisp per il Pdp-1, ma non funzionava molto bene, per mancanza di memoria adeguata; il Lisp, che lavora con i simboli e non con i numeri, facilmente convertibili in sistema binario, richiede una notevole quantità di memoria
24 In Lisp è possibile avere informazioni sulla natura di un oggetto aggiungendo una "p" alla fine del nome del tipo dell'oggetto: per esempio, volendo sapere se A è una lista, si scriverà LISTP A; volendo sapere se è un numero si scriverà DIGITP A, e così via. Il risultato di tale gestione dà "t" oppure "NIL"; "T" significa "true' cioè "vero" , mentre "NIL", contrazione del latino "nihil" significa propriamente nulla, vuoto, in questo caso falso". La convenzione Lisp di usare la lettera "p" come un predicato, fu l'ispirazione per una maniera hacker comune di porre una domanda. Quando qualcuno diceva "Cibo-P?" ogni hacker sapeva che gli stavano chiedendo se volesse qualcosa da mangiare. Le parole Lisp "T" e "NIL" finirono per significare, rispettivamente, "si e 'no".
25 Levy, op. cit.
26 Levy, op. cit.
27 "Lo sfortunato distaccamento di Cambridge della compagnia telefonica aveva già avuto a che fare con il MIT prima, e ne avrebbe avuto anche dopo. Un giorno fecero irruzione al nono piano del Tech Square [sede dei laboratori informatici] e chiesero agli hacker di mostrar loro dove fosse la Blue box. Quando gli hacker puntarono il dito verso il Pdp-6, i frustrati funzionari minacciarono di sequestrare l'intera macchina se gli hacker non avessero rimosso e consegnato l'interfaccia telefonica". Levy, S., 1984.
28 Levy, op. cit.
29 Levy, op. cit.
30 Alcuni di loro avevano fatto corsi per corrispondenza per ottenere la qualifica di fabbro per avere l'autorizzazione ad acquistare le chiavi grezze non vendute al pubblico.
31 Levy, op. cit.
32 David Silver era un ragazzo dislessico che andava male a scuola.
33 Implementazione IBM di un ambiente multi-utente operativa dalla metà degli anni Sessanta.
34 Levy, op. cit.
35 Quanto descritto è chiamato 'social engineering', 'ingegneria sociale'.
36Levy, op. cit.
Dall'intervista a Tom Knight.
37 In corso di costruzione e verifica al Tech Square, c'era un sistema time-sharing chiamato Multics. Tormentarono a lungo il sistema con trucchi e crash. Per gli hacker era inaccettabile, per la lentezza, gli eccessivi livelli di sicurezza e il sistema di addebito a tempo che scoraggiava un uso prolungato.
38 Un grosso armadio delle dimensioni di due lavatrici tipo lavanderia a gettone, soprannominato Moby Memory
39 E' il nucleo essenziale di un sistema operativo, responsabile della connessione tra le componenti fisiche di base e tutte le altre parti del sistema. Ha in pratica la funzione di svolgere le operazioni fondamentali, come ad esempio la ripartizione dei carichi di lavoro elaborativo della CPU ('Central Processing Unit', parte del calcolatore dove vengono processate le informazioni) ma gestisce anche le funzioni hardware del sistema.
40 L'Its poteva collegarsi a molte altre cose ed essere esteso all'infinito. Permetteva un miglior uso del monitor ed era dotato, di un sistema di editing che usava l'intero schermo. Gli utenti, non solo potevano lanciare contemporaneamente programmi, ma farlo in una sola volta: tutte funzioni molto avanzate per quei tempi. Inoltre ciascun utente poteva 'vedere' chi altro era presente sul sistema, e trasferirsi sul suo terminale per fare hackeraggio insieme.
41 Levy, op. cit.
42 Levy, op. cit.; dal diario dell'hacker Don Eastlake a cinque anni dall'entrata in funzione dell'Its.
43 Gestita da scienziati, non da militari, interessati soprattutto all'avanzamento della disciplina tanto da arrivare a distogliere fondi da progetti militari per destinarli alla ricerca di base.

44 Levy, op. cit.
45 La differenza si rifletteva anche nel genere di gioco che lo Stanford lab sviluppò dopo Spacewar del MIT. Nell'immaginario della Stanford c'erano elfi, hobbit e maghi della mitologia di Tolkien. Il gioco, chiamato Adventure, scritto da Don Woods, era l'espressione della personalità e del mondo in cui vivevano gli autori. Il genere Adventure, in cui ogni 'episodio' è come un piccolo programma, con un problema logico da risolvere, in un certo senso, era una metafora della programmazione: le esplorazioni dentro quel mondo erano simili a quelle fatte nelle prime macchine dove si vagava hackerando in assembly.
46 Local Area Network. Sistema di cavi, apparati con i relativi software e protocolli che consente lo scambio di informazioni in formato elettronico tra personal computer e server. Il termine indica un sistema che si estende su limitate superfici (tipicamente un palazzo).
47 Il C è un linguaggio di programmazione che utilizza costrutti e strutture ad alto livello, ma che ha capacità di controllo a basso livello tipiche del linguaggio assembly.
48 Levy, op. cit.
49 Un mainframe Xds-940, con un sistema time-sharing, situato nel sotterraneo del magazzino di San Francisco che ospitava Leopold's, il negozio di dischi più all'avanguardia della Baia di San Francisco.
50 Intitolato My Computer Likes Me (piaccio al mio computer) che vendette oltre 250.000 copie.
51 Levy, op. cit.
52 Acronimo di 'Beginners All-purpose Simbolic Instruction Code'. Linguaggio ad alto livello molto semplice da usare. Gli 'hacker puri' (come quelli del MIT) consideravano questo linguaggio 'fascista' perché la sua struttura limitata non consentiva un pieno accesso alle macchine e diminuiva il potere dei programmatori.
53 Molti lo interpretavano come Man in the street (uomo qualunque).
54 Il nome Altair fu preso da una puntata di Star Treck.
55 Roberts ammise in seguito, che sarebbe stato più economico assemblare i pezzi in fabbrica, dato che gli hobbisti frustrati spesso mandavano indietro le loro macchine semicompletate alla Mits, che le finiva gratuitamente.
56 Levy, op. cit.
57 Home brewed (fatto in casa).
58 Bootstrap è il processo con cui il sistema operativo carica se stesso. Quando viene accesa la macchina, una parte del programma carica il codice nella memoria del computer, ovvero fa il boot. Il termine è molto eloquente. Nel gergo di questi hacker indica una persona che partendo da un piccolo stimolo riesce a portare a termine e un'impresa.
59 Termine coniato da Richard Buckminster Fuller (Milton Massachusetts 1913-1993). Divenne famoso per l'invenzione della 'cupola geodesica' (una sfera composta da tetraedi, forma architettonica versatile, mobile economica e resistente) e i concetti di 'sinergia' (l'unione rappresenta più della somma delle parti) e di 'effimerizzazione' (la tendenza ad ottenere rendimenti crescenti con sempre minori investimenti di energia)
60 Periodo di prova di un software nel quale non si garantisce la stabilità e che ha il compito di individuare eventuali bug attraverso l'utilizzo costante degli utenti finali.
61 Levy, op. cit.
62 Non solo Marsh era diventato imprenditore. In queste nuove ditte, l'idea di concorrenza arrivò tardi, perché l'idea era di applicare anche qui il modello hacker, per cui tutti dovevano collaborare insieme.
63 Il pubblico, fuorché qualche curioso, era formato da persone che avevano ordinato l'Altair e che andavano a protestare perché non era ancora arrivato o non si riusciva a farlo funzionare o perché in attesa del Basic che non arrivava.
64 Allen fu assunto dalla Mits, ma Gates all'epoca era ancora studente. Si dovette ritirare da Harvard accusato di aver sfruttato per scopi commerciali personali le attrezzature universitarie.
Fonte: http://www.boston.com/globe/metro/packages/harvard/partone.htm.
65 La diffusione del loro Basic fu al di là di ogni aspettativa tanto da diventare uno standard di fatto e le aziende che entravano in questo nuovo mercato, che volevano usare il Basic, si rivolgevano alla MicroSoft di Gates.
66 Da: "LETTERA APERTA AGLI HOBBISTI" di William Henry Gates III, 3 febbraio 1976.
67 L'Altair aveva solo 4K complessivi; con 2K occupati dal Basic restava anche lo spazio per i programmi scritti dall'utente.
68 Non coperti da Copyright.
69 Levy, op. cit.
70 Usavano un processore diverso da quello dell'Altair, pertanto occorreva un Basic adattato al diverso hardware.

71 Levy, op. cit.
72 Il termine inglese free significa sia gratis che libero. Free in questo contesto è da intendere sempre come libero.
73 Per fare qualcosa con l'Altair, si dovevano girare degli interruttori sul frontale, tanto che agli hacker arrivavano a sanguinare le dita. L'alternativa escogitata in seguito, fu di collegare delle telescriventi (terribilmente rumorose).
74 Levy, op. cit.
75 Diede il codice a chiunque lo volesse e pubblicò anche delle subroutine sul 'Dr. Dobbs Journal'.
76 Wozniak era un hacker puro, non gli interessavano cose diverse dalla programmazione. Lavorava alla Hewlett-Packard e non voleva lasciare un lavoro che gli piaceva, ma quando chiese se erano interessati al progetto di un Apple per loro, lo ritennero invendibile e gli diedero il permesso di venderlo per conto proprio.
77 Chiamata così da Jobs, che una volta aveva lavorato in un frutteto.
78 Uno di questi diceva: 'la nostra filosofia è fornire software per le nostre macchine gratuitamente o a costo minimo'.
79 Assunsero anche un designer industriale.
80 Oltre alla Apple si può ricordare ad esempio la Commodore, la Radio Shack (il suo Trs-80 era il più famoso PC ai tempi), o l'Atari.
81 Meno rigidamente rispetto ai suoi colleghi-concorrenti. La Apple creò una sua comunità interna dove i tecnici si scambiavano le informazioni, a volte ammettevano 'qualche vecchio amico dell'Homebrew'.
82 È chiamata così l'applicazione che da sola induce ad acquistare il personal computer. Per gli utenti domestici furono i giochi, per le piccole aziende fu lo spreadsheet o foglio di calcolo.
83 Levy, op. cit.
84 Quello perfezionato da Don Woods al laboratorio dell'la della Stanford.
85 Levy, op. cit.
86 Il fatto che le immagini fossero in bianco e nero e graficamente elementari era trascurabile, dato che nessuno lo aveva fatto prima
87 Società per azioni quotata in borsa.
88 La macchina, era dotata di una guida di riferimento chiara che indicava dove fosse ogni cosa sui chip e la motherboard.
89 Levy, op. cit. In grassetto il corsivo dell'autore.
90 Levy, op. cit.
91 Non l'AppleII perché lo riteneva una macchina limitata per le sue esigenze. John lo riteneva 'cerebroleso' e lo disprezzava. Per altri, invece, questa limitatezza era un'altra sfida da superare.
92 Linguaggio di programmazione che cambia in base all'hardware, in cui le istruzioni in codice macchina vengono sostituite con codici mnemonici. Da non confondere con il linguaggio macchina dal quale differisce perché le istruzioni sono brevi sigle alfabetiche, di significato più facilmente comprensibile, anziché codici numerici.
93 Senza assunzione, con un fisso mensile più basso rispetto a dove lavorava prima ma con il 30% di diritti d'autore. Fu "la decisione più remunerativa della sua vita".
94 Famoso gioco per macchine a gettone da sala-giochi.
95 "L'Atari non segnalava mai sulla confezione il nome del programmatore del gioco rifiutandosi addirittura di rendere pubblico il nome dell'artista quando lo richiedeva la stampa. Quando qualcuno dei programmatori più importanti se ne lamentò, [...] [i dirigenti] si rivolsero agli hacker chiamandoli 'disegnatori di fazzoletti'. Quegli hacker saranno gli stessi che abbandoneranno l'azienda per formarne altre che distruggeranno la quota di mercato dell'Atari nel settore dei giochi". Cit. in Hackers tratto da John E Hubner e William E Kistner, What Went Wrong at Atari?, articolo ristampato in 'Inforworld', 28 novembre 1983.
96 Anche la On-line aveva i suoi giochi e i suoi copyright, se un giudice le avesse dato ragione sarebbe stato un precedente per la 'copiatura' di software.
97 Basata sul processore Intel 8088, la versione base aveva 16 K di memoria, usava cassette audio per memorizzare i programmi e costava 1.565 dollari. Tutti i modelli avevano una scheda video monocromatica incapace di visualizzare grafica bitmap. Non erano disponibili versioni con disco rigido. Ne furono venduti 13.000 esemplari nei primi tre mesi.
98 Per creare un nuovo tipo di interprete per Adventure e scrivere un word processor di facile uso per il Pcjr. Nel 1983, l'IBM presentò il Pcjr, che doveva essere una versione spartana ed economica dei suoi PC normali ma si rivelò un fallimento clamoroso: caro, sottodimensionato e con una tastiera inutilizzabile.
99 Basato sul DOS della Seattle Computer Products di cui aveva acquistato il copyright.
100 Levy, op. cit.

101 Contrazione di 'USErs-NETwork'. La rete dove vengono inviati i messaggi (articoli) dei Newsgroup. Basata sul protocollo NNTP. Contiene più di 1500 aree di argomenti alle quali l'utente contribuisce unicamente tramite posta elettronica. Possono essere trasportati testi, immagini e file audio.
102 Cominciò a lavorare nel centro elaborazione dati di Harvard, dopo aver preso una laurea magna cum laude in fisica. Era appassionato di informatica sin dalle superiori e prima di entrare in università era già esperto di assembly, sistemi operativi ed editor di testi.
103 In effetti i nuovi utenti, non iniziati all'etica hacker tendevano ad approfittare della situazione ed anche a far danni.
104 Linguaggio molto potente, facile da implementare che permetteva un controllo abbastanza esteso da soddisfare le esigenze degli hacker, ma avido di risorse hardware. Facendo girare il Lisp, il linguaggio dell'intelligenza artificiale, questa sarebbe stata la capostipite di una generazione di macchine capaci di 'imparare' e di dialogare con l'utente.
105 Le caratteristiche, di macchina 'pensante'.
106 Cit. in Hackers tratto da Essay di R. Stallman.
107 Quanto descritto è definito 'reverse engineering'.
108 Da 'Il progetto GNU' di Richard Stallman. Trad. it. disponibile come documento elettronico http://www.gnu.org/gnu/thegnuproject.it.html
109 Il progetto Gnu e Linux saranno trattati diffusamente nei prossimi capitoli.

2 Uno sguardo ai vincoli informali.

2.1 Ideologia ed 'Etica hacker'

Le comunità hacker sono strettamente collegate con il mondo scientifico e accademico ed è qui che affondano le proprie radici.
Esse possono essere definite come una forma particolare di comunità di ricercatori scientifici. Difatti, buona parte delle norme di comportamento individuabili nella loro tradizione culturale corrispondono alle regole tipico ideali della ricerca scientifica.
All'esposizione delle istituzioni, ovvero le 'regole del gioco', applicate dagli hacker (par. 2.2), seguirà un confronto con quelle adottate dalla comunità scientifica ed il 'Metodo scientifico' attraverso studio compiuto da R. Merton in 'La sociologia della scienza'[1] (par. 2.3).

A partire dalla fine degli anni '50, come sottolinea Levy, i componenti della comunità hacker hanno "inavvertitamente costruito un corpo organico di concetti, convinzioni e costumi". Sono stati favoriti da una serie di fattori concomitanti, tra questi, la gestione del laboratorio di intelligenza artificiale del MIT, molto aperta (poteva sedersi ad un computer chiunque passasse dal laboratorio, purché dimostrasse di saper fare qualcosa di valido, senza causare danni), paritetica e poco burocratizzata. Questo è stato un fattore molto importante, perché ha fatto da presupposto e da fertile base per la nascita, lo sviluppo, ed il consolidamento della comunità; riporta Levy parlando di Greenblatt: "Nessuno gli chiedeva di rendere conto delle sue intenzioni. Non aveva il problema di dover notificare la sua decisione ai superiori, [...] non c'erano percorsi burocratici da intraprendere perché [...] in quei primi giorni in cui vedeva la luce l'Ia lab, gli hacker stessi erano i canali di comunicazione. Era l'etica hacker messa in pratica".
Il laboratorio del MIT, 'riserva' protetta dal resto dell'università e dall'ambiente esterno, ha potuto affermare le proprie istituzioni innovative e antiburocratiche, e contrapporle per lungo tempo alle regole del 'mondo reale'.
Pur discostandosi dai metodi ritenuti legittimi dalla 'scienza normale'[2], ha dimostrato la validità dei propri, con il continuo conseguimento di apprezzabili risultati definibili come scientifici.
La seconda e la terza generazione di hacker, pur non partendo direttamente da ambienti accademici, ha sviluppato un corpo di regole molto simile a quello del MIT. Lo stesso per le comunità nate successivamente in ogni parte del mondo, che apparentemente (prima dell'avvento di Internet) non avevano alcun diretto collegamento.
Si può ipotizzare che questo sia avvenuto per tre consistenti punti di contatto, che allontanano l'ipotesi alternativa della soluzione di continuità sia in senso diacronico, tra le generazioni, sia sincronico, tra una comunità e l'altra.
Il primo è naturalmente il computer. La sua architettura, i suoi vincoli, il suo linguaggio, ecc. hanno plasmato il modo di interagire con lui, arrivando in molti casi ad alterare la struttura cognitiva del programmatore e le sue relazioni interpersonali. Come fa notare un hacker intervistato da Levy: "C'era il mondo intero dentro quel computer... un sistema di pensiero".
Il secondo punto di contatto è costituito dagli individui che passano da una comunità all'altra portandone i metodi e i valori, un caso molto significativo riportato da Levy è quello di John McCarthy che fondò nel '59 l'Ia lab del MIT e dopo averne lasciato la direzione fondò nel '62 il Sail (Stanford Ia lab), il centro di calcolo dell'università Stanford, dirigendo entrambi in modo da assecondare l'etica hacker.
Infine, la costante presenza di molteplici canali di comunicazione. In aggiunta alle reti informatiche, che prima dell'avvento di Internet commerciale era a disposizione solo degli istituti universitari, di qualche laboratorio privato, e di pochi altri pionieri che gestivano reti amatoriali. Tra questi canali troviamo ad esempio i club come il TMRC oppure l'Homebrew computer club o ancora gli odierni LUG (Linux User Group), oppure le riviste di settore e quelle dedicate all'underground digitale, che tanta parte hanno avuto nella formazione della comunità di hacker dell'hardware e in seguito del software per computer da tavolo, che si sono moltiplicate fino a coprire oggi ogni nicchia informatica.
L'avvento di Internet commerciale e il miglioramento dell'accessibilità alle macchine ha generato un apertura sistemica[3], così, le comunità possono essere, e sono, in contatto tra loro, ma anche le persone esterne al mondo informatico possono incontrare questa cultura, frequentarla ed integrarsi, scambiando

conoscenze e competenze in modo inter o transdisciplinare. Esempi sono la manualistica informatica e la relativa traduzione fatta anche da non programmatori o iniziative come Nupedia e Wikipedia[4], l'italiano GNUtemberg[5], il progetto OpenLaw del Berkman Center for Internet and Society della Harvard Law School[6], ma se ne potrebbero aggiungere moltissimi altri.

Le singole persone possono essere più o meno coinvolte nella comunità e applicare le regole e le convinzioni in modo più o meno rigoroso. Ma nei casi estremi, e l'Ia Lab del MIT dei primi tempi ne è un ottimo esempio, vi è un immersione totale, totalizzante. Il coinvolgimento descritto da alcuni hacker è comparabile a quello dato da una fede religiosa, paragone emerso spesso anche nel libro di Levy: "votando le loro abilità tecniche nell'informatica", "una dedizione raramente riscontrata al di fuori dei monasteri", "devoti studenti della macchina", "condurre il mondo esterno dentro quel segreto, fare proseliti per l'etica hacker", "dei gesuiti tecnologici", "non c'erano missionari che cercassero di fare adepti. Il computer stesso operava le conversioni".
Queste persone, inoltre, sono accomunate dal fatto di provare piacere nell'atto in sé di mettere le mani sui computer, di programmare, di risolvere problemi, di costruire cose e di aiutare gli altri a fare altrettanto. Come sottolinea Eric S. Raymond: "Diventare un hacker è molto divertente, ma è un tipo di divertimento che necessita di un sacco di fatica. La fatica necessita di motivazione. [...] Per essere un hacker devi provare una forte eccitazione dal risolvere problemi, affinando le tue abilità ed esercitando la tua intelligenza"[7]. Solo coloro che amano la programmazione, vivono - parafrasando Max Weber[8] - per la loro intima 'vocazione' per la scienza della programmazione, immersi ne 'l'esperienza vissuta' di questa scienza, provando la strana ebbrezza, derisa dai non iniziati, e la passione nel seguire la propria via. Solo costoro hanno, quindi, l'incentivo necessario per operare in un ambito fondato su contributi volontari, che si regge fondamentalmente sulla reciprocità e sulla logica del dono.

Alcuni, arrivano a descrivere la loro passione come un piacere quasi carnale o di profonda immedesimazione con la macchina o ancora come lo slancio creativo di un artista nel momento culminante della propria ispirazione, con la relativa alterazione o perdita del senso del tempo, dello spazio e delle cose materiali.

2.2 Le 'regole del gioco' degli hacker.

Le 'regole del gioco' informali, ovvero, i vincoli osservati spontaneamente dagli hacker (convenzioni e codici morali), si sono stabilizzati nel tempo, formando un corpo organico di concetti, principi e costumi. Sono regole non scritte, variamente interiorizzate dai partecipanti e applicate perché sanno che agendo così fanno 'la cosa giusta'. Formano un sistema di punti, strettamente integrati, che possono avere maggiore o minore importanza.

Ecco in dettaglio i punti fondamentali:
Libertà dell'informazione: è un concetto cardine, primo ad essere citato insieme all'accesso alle macchine, e precondizione per gli altri.
Agli esordi della scienza informatica occorreva giungere ad un rapporto 'intimo e profondo' con la macchina per poter programmare nel suo specifico linguaggio. L'accesso libero e aperto all'informazione, è l'unico modo per conoscere l'architettura del calcolatore e quindi per poter programmare senza che quest'attività diventi frustrante.
Per gli hacker l'informazione deve essere libera, per principio, in tutti i settori, non solo nell'informatica, perché senza la possibilità di accedervi non è possibile migliorare e far progredire le cose. Inoltre, maggiore è la complessità di un programma o di un qualsiasi problema da affrontare e maggiore è il beneficio che si trae nell'affrontarlo liberamente e collettivamente. Un sistema aperto e libero, può godere di contributi occasionali e amplificare fenomeni come quello della serendipità[9] e della ricombinazione creativa delle informazioni, entrambe preziose fonti di innovazione.
La tecnologia ed il software sono ritenute risorse che possono incidere sulla vita di tutti, migliorandola, per questo sono da condividere e da mantenere libere da poteri monopolistici.
L'informazione non è considerata come una proprietà, come qualcosa che si possa 'rinchiudere'. La cosa è per loro inconcepibile, assurda oltre che controproducente. Ulteriore motivazione che portano è che la scienza in generale è frutto di una graduale accumulazione di sforzi collettivi, molto spesso di pubblico dominio, rinchiuderne una parte significa non riconoscere il contributo di chi ci ha preceduto, e per merito

dei quali possiamo andare avanti nello stato dell'arte, significa perciò commettere un furto e impedite agli altri di progredire.
Hanno fatto proprio l'aforisma Newtoniano 'se ho visto così lontano è perché stavo sulle spalle di giganti'.
L'informazione libera e pubblica, permette di sapere quali progetti sono in corso, quindi, cosa è già disponibile ed utilizzabile ed eventualmente dove è possibile e necessario cooperare, in modo da evitare che si sciupino energie per rifare qualcosa che già c'è. Consente, inoltre, di sapere chi sta lavorando su un progetto e quale contributo sta dando, pertanto è anche prerequisito per la reputazione dei partecipanti che a sua volta determina lo status all'interno della comunità.
Accesso alle macchine: la possibilità di esplorare i computer, le macchine e tutto ciò che poterebbe insegnare qualcosa deve essere totale. Si impara facendo, osservando il funzionamento delle cose e dei sistemi, rielaborando in modo creativo le informazioni e cercando di produrre e riprodurre ciò che si è osservato: chiunque la pensi ed agisca così è il benvenuto. Le informazioni vanno usate per creare cose nuove o migliorare quelle vecchie. Dato che ciò che è difettoso e che funziona male è sgradevole e molesto, devono intervenire per poterlo migliorare; 'metterci su le mani', è un imperativo categorico. Qualsiasi persona, metodo o legge che sia di ostacolo alla conoscenza e all'uso delle risorse è da avversare o semplicemente da ignorare. Occorre adoperarsi in ogni modo per superare gli ostacoli.
No a burocrazia e formalismi: la burocrazia, qualunque sia la sua origine, universitaria, aziendale o dello stato, è un intralcio che costringe a compromessi e ad inutili attese ed impedisce che si esprima pienamente lo spirito di ricerca e la possibilità di sperimentazione creativa ed innovativa degli hacker. Essi ritengono che i burocrati sfruttino le regole formali solo per fare i propri interessi e per rafforzare il proprio potere.
Qualsiasi organizzazione hacker, deve presentarsi come un sistema libero ed aperto, deve promuovere il libero flusso delle informazioni, essere il più possibile informale, interattiva e capace di recepire senza formalità le nuove idee. Il risultato non è un sistema anarchico ma un organismo, o meglio, un sistema ecologico in cui ogni elemento ha un suo ruolo e ne garantisce l'equilibrio, pena la decadenza e la scomparsa del sistema stesso.
Per questi motivi le iniziative raramente vengono pianificate, semplicemente qualcuno inizia a fare e il resto si struttura intorno a questo Input iniziale che diverrà anche il nodo centrale del network di collaboratori al progetto la cui direzione di sviluppo sarà concordata con gli stessi utenti-sviluppatori.
Semplificazione ordine, riduzione e ottimizzazione delle risorse: sono imperativi da applicare a tutti i livelli.
Ciò che viene scambiato deriva da contributi volontari di numerosissime persone che si organizzano e si coordinano tra loro in modo molto aperto ed elastico non c'è burocrazia e tutto deve essere accessibile senza regole troppo formalizzate perché altrimenti scoraggerebbe la partecipazione. L'eccesso e il disordine sono degli intralci paragonabili alle 'lungaggini' burocratiche. Lo spreco, di tempo, di risorse informatiche, della larghezza di banda della rete, ecc. è condannato in tutte le sue forme.
Una forma particolare di spreco si ha quando si verifica una biforcazione di un progetto (cosa avvenuta ad esempio per UNIX), oppure quando nascono più gruppi che si occupano di cose equivalenti provocando una inutile duplicazione degli sforzi. In questi casi, se possibile, occorre mirare ad una fusione o ricongiungimento.
Sono, infine, da evitare cambiamenti radicali nei progetti, se non indispensabili, perché possono creare biforcazioni per defezione di una parte.
Con un computer puoi creare arte: il codice del programma possiede una bellezza propria. È un'unità organica con una vita indipendente da quella del suo autorepromotore. La parte estetica dello stile di programmazione è incoraggiata e molto apprezzata.
Comunità: è il luogo virtuale-reale in cui tutto si svolge; in realtà si tratta di un reticolo di comunità che ne forma di più grandi, un insieme di network dai confini sfumati che presa nel suo complesso forma, la Comunità del Software Libero. Restando nella metafora ecologica, è l'ambiente naturale dove il sistema ecologico vive e dove avviene lo scambio di materia.
L'appartenenza è sempre intenzionale, si fa parte della comunità solo se si partecipa, si collabora e si coopera volontariamente. In questo modo si acquisisce lo status di hacker ed una posizione, nella gerarchia informale della comunità di appartenenza, che si basa sulla reputazione.
Come sottolinea Raymond in 'How To Become A Hacker', per acquisire lo status di hacker occorre appartenere ad una comunità e per farne parte è necessario condividerne la cultura, aver contribuito a costruire qualcosa e che gli altri lo abbiano riconosciuto:
"There is a community, a shared culture, of expert programmers and networking wizards that traces its history back through decades to the first time-sharing minicomputers and the earliest ARPAnet experiments. The members of this culture originated the term `hacker'. Hackers built the Internet. Hackers made the Unix operating system what it is today. Hackers run Usenet. Hackers make the World Wide Web

work. If you are part of this culture, if you have contributed to it and other people in it know who you are and call you a hacker, you're a hacker.".[10]
Coloro che ne prendono parte, si trovano d'accordo sul fatto che il software in libera distribuzione sia qualcosa di positivo, degno di sforzi collettivi e molti competono per il prestigio di regalare tempo, energie e creatività: è sulla base di tale concordanza che si definisce o meno l'affiliazione a tale cultura. Quello che varia sono le sottoculture di riferimento e le motivazioni individuali con una minore o maggiore adesione personale (fino a rasentare il fanatismo) ed un atteggiamento più o meno avverso al software commerciale. Perseguono obiettivi diversi e comportamenti adattivi e cooperativi graduati in vario modo.
La comunità è il punto di riferimento dove 'incontrarsi'. Può essere in concreto una mailing-list, una chat un sito internet, un luogo geografico come la sede di un club o un laboratorio universitario. Ci si incontra per scambiare i propri 'doni', che consistono in conoscenza, informazioni, consigli, prestazioni, competenza e software (sviluppo, miglioramento, modifica, ecc.), ma che possono anche essere strumenti, beni materiali o la possibilità di un loro utilizzo. In cambio si ricevono 'doni' di questo stesso tipo e, cosa molto importante, si ottengono risorse più strettamente sociali, come approvazione, stima e sostegno.
Ogni comunità si struttura intorno ad una combinazione di una o più piattaforme hardware, sistemi operativi e programmi software e ad un leader (o un ristretto gruppo come avviene, ad esempio, nelle comunità Perl e Apache) che ha dato avvio alla comunità e ai suoi progetti o l'ha ricevuta in 'eredità'.
In questo sistema prevale la subordinazione della proprietà e dell'individualismo al bene comune. E i comportamenti contrari sono fortemente stigmatizzati.
Condivisione: delle informazioni, degli strumenti e delle competenze è un doverevalore. Si dà per quanto è nelle proprie competenze e possibilità ma si riceve, dagli altri partecipanti alla comunità, con le stesse modalità, in un rapporto che risulta asimmetrico a vantaggio del singolo partecipante.
Cooperazione: considerando i punti precedenti la cooperazione diventa quasi un evento naturale che crea una Sinergia[11] positiva che avvantaggia i partecipanti. I cooperanti possono anche essere aziende attraverso i propri dipendenti.
Il free riding a questo livello provoca danni irrilevanti. Il bene di cui si appropria il free rider è informazione già pubblica, che per sua natura non si consuma e distribuita con licenze che ne consentono qualsiasi utilizzo[12] tranne la 'chiusura'. La non partecipazione e la non restituzione è compensata dal fatto che il 'valore' dei programmi cresce all'aumentare del numero degli utenti.
Leadership: i leader delle comunità che gestiscono i progetti sono dei 'primi tra pari'. Il loro stile di leadership, deve necessariamente tenere conto del fatto che il loro ruolo è determinato dai partecipanti che sono dei volontari completamente autonomi, capaci, e competenti, che decidono di attenersi alle direttive del leader perché lo ritengono autorevole, in grado di indirizzare la comunità e i progetti attivi verso l'ottenimento di risultati concreti. Il leader (o il gruppo-leader) deve essere partecipativo-delegante [13] in grado di guidare operativamente il gruppo, senza imporre, e sostenendo la relazione. Comunemente le scelte sono discusse e compiute collettivamente. Scelte non condivise dalla comunità se non giustificate e se protratte nel tempo sono causa di perdita della posizione di leader. Altri prenderanno il suo posto come nodo centrale del network, in caso contrario il rischio è la disgregazione o la scissione della comunità e del suo progetto o addirittura la scomparsa.
Se il progetto è in fase di avvio, una leadership inadatta impedisce il raggiungimento di una massa critica tale da dare vita propria al progetto stesso, provocandone la scomparsa ancor prima di nascere.
Atteggiamento meritocratico: gli hacker non si curano delle caratteristiche superficiali di ciascuno e non basano la propria valutazione sulla base di falsi criteri quali ceto, età, etnia, genere e posizione sociale; operano una sospensione del giudizio e guardano a quello che ciascuno è in grado di offrire di creativo e innovativo. L'accoglienza del dodicenne Peter Deutsch nella comunità degli hacker del MIT è un ottimo esempio. Le 'credenziali appariscenti' non sono prese sul serio se prima non si dimostra quello che si sa veramente fare davanti ad un computer.
Reputazione: il 'gioco della reputazione' determina lo status all'interno della comunità, l'importanza e centralità del nodo che si rappresenta nel network.
Si viene accettati senza pregiudizi e si eleva il proprio status per meriti acquisiti sul campo, per ciò che si è creato e per l'aiuto al progresso dello stato dell'arte, oltre all'aiuto verso gli altri.
La stima quindi è guadagnata con le capacità e/o le buone idee, vi è inoltre un forte grado di apprezzamento per la raffinatezza tecnologica e l'estro digitale.
Uso del gergo hacker: il gergo condiviso dagli hacker è parte integrante della loro cultura sedimentata nel tempo, è un'arma di esclusione dalla comunità, ma anche strumento di inclusione perché rafforza il senso di comunità e fa da collante ideologico. Oltre a derivare inconsciamente dall'ambiente, riflettendolo, (compresa l'architettura interna del computer e i linguaggi di programmazione), è creato in modo cosciente, come sperimentazione, intenzionale, spontanea e spesso giocosa, di nuove espressioni: anche

il linguaggio è qualcosa da esplorare, smontare e rimontare, sperimentare, è, in altre parole, qualcosa da hackerare.
Il suo apprendimento fa parte dell'iter formativo di coloro che aspirano ad entrare in questa comunità.
Tramite Internet è possibile accedere al 'Jargon File', un documento in continua evoluzione, liberamente disponibile per tutti gli utenti della Rete, avviato nel 1975 a Stanford, mantenuto inizialmente da Raphael Finkel, e attualmente da Eric Raymond. Contiene "un manuale completo dello slang degli hacker che chiarisce molti aspetti della tradizione, del folklore, e dello humor 'hackish'"[14]
È un linguaggio volutamente allusivo e di difficile comprensione per gli estranei a metà strada tra il linguaggio tecnico e lo 'slang', come fa notare Raymond, "the line between hacker slang and the vocabulary of technical programming and computer science is fuzzy".
I computer possono cambiare la vita in meglio: il tradizionale punto di vista del 'mondo reale' sull'utilizzo dei computer e delle tecnologie, ritenuto come minimo restrittivo, è stato sovvertito e dilatato dalle idee degli hacker. Essi considerano il computer come un arma politica, in grado di liberare gli individui dall'oppressione, e come mezzo di trasformazione e costruzione della realtà.
Operativamente gli hacker si affidano soprattutto alla propaganda del fatto concreto: il metodo per affermare le loro idee è dare l'esempio, mostrare che il loro modello di approccio alle cose funziona (e farlo sapere).
Fare la cosa giusta: significa, che per qualsiasi problema esiste una soluzione, che è "proprio... quella: l'algoritmo perfetto". È la "linea più corta tra due punti" e non deve esserci la possibilità di farlo meglio. L'unica soluzione, la più corretta, ordinata ed elegante, che soddisfa contemporaneamente tutti i diversi punti di vista e tutti i problemi.
Ciò che è giusto lo si apprende dall'esempio dato dai membri più 'anziani' e frequentando la comunità.
I programmi devono fare la cosa giusta, ma anche le persone: seguire le 'regole del gioco', l'etica, le norme e i valori degli hacker, senza curarsi del giudizio altrui e delle conseguenze personali (ad esempio se si infrange la legge) è la cosa giusta.

L'efficacia delle regole si regge su forti pressioni della comunità al loro rispetto e su sanzioni per la loro inosservanza che dipendono dalla gravità del comportamento. Coloro che violano le regole subiscono la riprovazione collettiva. E dato che anche gli eventi negativi godono di pubblicità ne risente la reputazione. Oltre a questo fatto, in concreto, le sanzioni più gravi prendono la forma della cancellazione del nome dalla credit list, quando ad essere sanzionato è un programmatore, e del boicottaggio, nel caso di organizzazioni, aziende ecc.

2.3 Comunità a confronto.

Comparando le istituzioni della comunità hacker e quelle della comunità scientifica, si riscontrano delle sorprendenti analogie.
La sovrapposizione delle regole è tale da confermare l'origine comune, o meglio, che lo sviluppo di Software Libero può essere senz'altro definito come un tipo particolare e specifico di ricerca scientifica.
La scienza è un processo collettivo di accumulazione di conoscenza che poggia su tre pilastri: pubblicità, controllabilità e ripetibilità. In mancanza di un solo pilastro il processo crolla, oppure, non siamo di fronte ad una attività definibile come scientifica. Il Software Libero è pubblico, controllabile e ripetibile[15].
'La struttura normativa della scienza' descritta da Merton in La sociologia della scienza[16] è utile per individuare le analogie e confrontare le regole istituzionali e le modalità di funzionamento del metodo scientifico con le istituzioni degli hacker nel contesto delle comunità del Software Libero. Merton, quando scrive "La scienza è pubblica e non privata", sottolinea l'aspetto collettivo e pubblico della scienza. gli hacker sostengono lo stesso concetto quando affermano che il software deve essere libero.
Nel suo scritto, Merton considera i 'costumi' (mores) che definiscono i confini del campo dei metodi della scienza.
Le regole di comportamento, proprie della scienza, il suo ethos, spiega Merton, sono un insieme di norme e valori intimamente avvertite come vincolanti dagli scienziati. Si noterà la corrispondenza tra il seguire l'ethos della scienza con il 'fare la cosa giusta' per gli hacker. Esse assumono la forma di prescrizioni, divieti, preferenze e direzioni permesse, sono direttamente insegnate oppure acquisite nel tempo per imitazione. Il loro sommarsi e la maggiore o minore interiorizzazione va a creare la coscienza scientifica del singolo uomo di scienza.

Non si tratta di norme formali istituzionalizzate, ma di usi ed abitudini propri di quest'ambito, deducibili dal consenso intorno ai comportamenti ammessi e dalle sanzioni accompagnate dall'indignazione morale in caso di comportamento deviante.
Sono norme, queste, che vengono a formarsi per mezzo della cristallizzazione dei metodi e delle regole pratiche, funzionali al raggiungimento del fine istituzionale della scienza, vale a dire, l'incremento quantitativo e qualitativo delle conoscenze verificate. Si tratta, quindi, di norme tecniche che rappresentano nel contempo dei principi morali. Sono giustificate dalla coerenza metodologica, ovvero, dal fatto che il loro rispetto consente di raggiungere gli obiettivi perseguiti dalla scienza, ma sono vincolanti anche perché intimamente sentite come 'cose buone e giuste'.
Per descrivere e comprendere l'ethos della scienza moderna, Merton considera quattro insiemi di imperativi istituzionali: Universalismo, 'Comunismo'[17], Disinteresse e Scetticismo sistematico.
Eccoli in dettaglio e a confronto con le istituzioni hacker.

Universalismo: la ricerca scientifica deve avere carattere universale, pertanto, i risultati delle ricerche devono essere valutati solo in base al loro reale valore scientifico, indipendentemente dalle caratteristiche individuali dei ricercatori. Esso viene ben espresso dalla disposizione che esige che il raggiungimento dei risultati sia effettuato attraverso procedure stabilite anteriormente e in accordo con il corpo di osservazioni e procedure fino a quel momento generalmente utilizzate.
Qualsiasi risultato, ottenuto da uno studioso, deve essere accettato o rifiutato indipendentemente dalle sue caratteristiche personali o sociali, che vanno completamente ignorate nel giudicare le sue enunciazioni. "L'oggettività esclude il particolarismo", sottolinea Merton e aggiunge poco dopo, "L'imperativo dell'universalismo è profondamente radicato nel carattere impersonale della scienza"[18], ed è proprio l'oggettività attenendosi al metodo prestabilito, alla routine ufficiale, ad impedire l'imposizione di criteri di interesse esclusivo di un singolo o di un gruppo ristretto.
L'imperativo dell'universalismo si manifesta anche nell'esigenza che le carriere siano accessibili a chiunque dimostri di avere delle adeguate capacità. Escludere una persona capace per motivi estranei alla scienza risulterebbe un comportamento immorale. Ma è al contempo un imperativo funzionale, infatti, questa è un'esigenza che dipende direttamente dal fine della scienza e eventuali limitazioni diverse dalla competenza avrebbero ripercussioni sull'evoluzione della conoscenza scientifica.

L'atteggiamento meritocratico, nel sistema delle 'Regole del gioco degli hacker', corrisponde bene a quello che Merton chiama Universalismo, sia per la valutazione indipendente dalle caratteristiche personali e sociali del partecipante, nella valutazione del suo lavoro, sia per l'esigenza di apertura delle carriere nei confronti di chiunque dimostri di avere delle capacità.

Merton accenna anche alle condizioni 'ambientali', politiche, che rendono possibile il mantenimento dell'universalismo. Dato che anche la scienza dipende dalla struttura sociale nella quale è integrata, solo la democrazia consente di applicare pienamente il principio universalistico, dato che fa parte del suo stesso ethos. In una società democratica aperta, per contro, quando è la scienza a chiudersi dentro a privilegi di casta, l'apparato politico può intervenire per ripristinare le condizioni di uguaglianza delle opportunità.

L'Universalismo, anche per quanto concerne il punto di vista 'politico' e 'ambientale', è un valore fondamentale per gli hacker. Essi ritengono che le condizioni politiche per favorirlo siano preziose. Democrazia, libertà, sono valori costantemente richiamati nei loro scritti 'politici'. E a livello 'ambientale' questo fatto viene espresso con la profonda ostilità che manifestano nei confronti della burocrazia e dei formalismi poiché ostacolano l'ingresso dei dotati e meritevoli e la libera espressione delle proprie potenzialità.

'Comunismo' (o Comunitarismo): è un principio generale di grande importanza in ambito scientifico, secondo il quale ogni conoscenza scientifica non deve essere considerata come proprietà privata del ricercatore, ma deve essere messa a disposizione dell'intera comunità scientifica. "Le scoperte sostanziali della scienza sono un prodotto della collaborazione sociale e sono assegnate alla comunità. Esse costituiscono un'eredità comune, in cui il diritto del produttore individuale è severamente limitato".[19]

Anche se una legge o una teoria prende il nome dello scienziato che l'ha formulata, non significa che essa sia una sua proprietà e neppure che possa farne uso esclusivo, è un'usanza puramente evocativo-commemorativa. L'etica scientifica, infatti, riduce le possibilità di disporre delle proprie scoperte proprio per il fine della scienza stessa. Il diritto di 'proprietà' è, quindi, limitato alla paternità e alla stima e

riconoscimento che ne derivano; essi dovrebbero "essere commisurati alla significatività dell'incremento apportato al fondo comune di conoscenze".[20]
Questo stato di cose genera una cooperazione competitiva, in cui assume un'importanza fondamentale il fatto di raggiungere per primo un risultato originale. La stima e il riconoscimento sono la moneta con cui viene ripagato lo scienziato che vince la competizione, e rendere pubblica la scoperta è l'unico mezzo per dimostrarlo e ricevere il proprio 'compenso'.
La messa in comune, poi, consentirà ad altri di vincere nuove sfide. È il fine stesso della scienza, rivolto ad accrescere il patrimonio comune delle conoscenze, ad imporre la piena comunicazione pubblica delle scoperte. La mancata comunicazione delle proprie scoperte da parte di uno scienziato è una violazione ad una precisa norma istituzionale e l'omissione è biasimata e condannata.
Il carattere comunitario della scienza, si riflette nella consapevolezza degli scienziati del fatto, che la loro attività e l'avanzamento dello stato dell'arte, è una collaborazione tra generazioni, in cui quelle attuali, dipendono dall'eredità di conoscenze che hanno ricevuto dal passato. "L'umiltà del genio scientifico [è] culturalmente appropriata", e Merton sottolinea questo punto, riportando l'aforisma Newtoniano ""Se io ho visto più in là è stato perché stavo sulle spalle di giganti" - [che] esprime al tempo stesso un senso di debito nei confronti dell'eredità comune, ed un riconoscimento della qualità essenzialmente cooperativa e selettivamente cumulativa delle realizzazioni scientifiche"[21].
L'idea di proprietà privata che l'economia capitalistica ha nei confronti della tecnologia è incompatibile con l'ethos della scienza ed in particolare con il principio del comunismo scientifico. La pratica dei brevetti che escludono i più dalla possibilità di usare le scoperte è stata variamente osteggiata da buona parte degli scienziati[22]. Alcuni di essi sono arrivati a brevettare le proprie scoperte per garantire che rimanessero disponibili alla comunità scientifica e all'uso pubblico.

Condivisione, cooperazione, libertà dell'informazione, sostenuti dalla comunità del Software Libero, rientrano nell'imperativo istituzionale che Merton chiama 'Comunismo' (Comunitarismo) e la comunità è il suo punto di riferimento.
Il software free è sempre un lavoro collettivo, l'eponimia quasi non esiste ma il lavoro qualitativamente e quantitativamente rilevante è riportato nella credit list contenuta in tutti i prodotti free (programmi, manuali, articoli, ecc.); essa riporta la lista dei nominativi di tutti coloro che hanno dato un contributo significativo, con l'indicazione di cosa hanno fatto; il più delle volte è anche indicato l'indirizzo e-mail, l'eventuale organizzazione di appartenenza (università, azienda, ecc.) e la provenienza geografica. È il riconoscimento pubblico per coloro che hanno donato il proprio lavoro, attestato all'interno del prodotto free.
Priorità ed originalità anche per gli hacker sono fondamentali. Trattandosi di lavori collettivi assumono una forma differente dovuta allo sforzo di ripartire il lavoro evitando duplicazioni e sovrapposizioni. Nella programmazione software ci sono numerosi modi per ottenere lo stesso risultato. La priorità in quest'ambito non significa chi arriva prima ad una certa scoperta, ma chi riesce a fare meglio una certa cosa. In caso di disputa viene scelto in modo democratico cosa integrare nel progetto.
Come misura difensiva verso la chiusura della conoscenza, il software viene rilasciato con licenze aperte, con la GNU/GPL in testa. Contro i brevetti l'ostilità è totale[23].

Disinteresse: è un elemento istituzionale di controllo sulla gamma di motivazioni possibili da parte dello scienziato; requisito fondamentale dello scienziato è quello di essere disinteressato, cioè il suo lavoro deve contribuire ad aumentare la conoscenza scientifica, senza peraltro perseguire interessi personali o particolari nell'esercizio del proprio ruolo professionale.
Il carattere pubblico e controllabile della scienza è alla base dell'esigenza del disinteresse. L'attività scientifica per sua natura, è costantemente sotto esame da parte dei colleghi propri pari, che per questo sono responsabili gli uni verso gli altri.
Il disinteresse, come gli altri imperativi istituzionali, è radicato nella coscienza scientifica, il timore di sanzioni o di conflitti psicologici, porta gli scienziati a conformarsi a questa norma. Come fa notare Merton, "Culto del proprio gruppo, settarismo informale, pubblicazioni numerose ma poco significative"[24] servono solo all'autostima del singolo ma hanno vita breve nel campo della scienza. Questi comportamenti, incluso il tentativo di 'imbrogliare', superando gli avversari in modo scorretto, è fugato proprio dal controllo dei pari.

Anche per il disinteresse, come per il comunitarismo, ci si può collegare alle regole della condivisione e cooperazione; in questo caso il punto di vista è quello del singolo e non più del collettivo.
L'attività di programmazione a codici aperti consente il controllo da parte degli altri. All'interno dei progetti, l'integrazione di nuove parti di codice viene fatta esclusivamente in base al suo valore qualitativo, non perché fatta da un particolare programmatore o gruppo; così come i tentativi di plagio sono riconoscibili e

prontamente segnalati. Situazioni diverse generano malcontento e rimostranze e mettono in pericolo il buon esito del progetto.
Ad un altro livello di analisi, quando si considera l'attività di programmazione totalmente volontaria, è da ricondurre al rispetto dell'imperativo del disinteresse la difficoltà di individuare degli incentivi diversi dal piacere in sé di 'esplorare' e di 'metterci su le mani' condividendo la propria esperienza, dalla soddisfazione per un lavoro ben fatto riconosciuto dai propri pari o dal desiderio di migliorare la propria reputazione.

Scetticismo sistematico: deve essere un atteggiamento dello scienziato che si appresta a studiare determinati fenomeni della realtà. In particolare si parla di uno scetticismo sistematico, che deve distinguere la pratica scientifica e la metodologia di ricerca. Esso, spiega Merton, "è variamente correlato agli altri elementi dell'ethos scientifico. Si tratta, al tempo stesso, di un mandato metodologico ed istituzionale. La sospensione del giudizio e l'esame distaccato delle credenze secondo criteri logici ed empirici hanno periodicamente coinvolto la scienza in conflitti con altre istituzioni. La scienza, che pone interrogativi di fatto, inclusi gli aspetti potenziali, che riguardano ogni aspetto della natura e della società, può entrare in conflitto con altri atteggiamenti verso gli stessi elementi cristallizzati, e spesso ritualizzati, da altre istituzioni.".
Lo Scetticismo sistematico è un concetto relativo all'atteggiamento che deve tenere l'uomo di scienza.
In un certo senso è come ripartire ogni volta da zero mantenendo la mente aperta e sgombra come quando si è dei principianti. Nessun ostacolo, pregiudizio o interesse devono esserci davanti all'obiettivo da raggiungere.

Per gli hacker la tensione verso la ricerca di nuovi modi per abbattere le barriere che impediscono di conoscere a fondo la tecnologia, arrivando, se occorre, anche a superare i limiti della legalità, è sempre un fatto strumentale all'obiettivo di ricerca ed esplorazione e non avviene mai per interesse personale.
La messa in discussione delle regole, anche entrando in conflitto con quelle degli altri, non è mai pregiudiziale. Altrettanto forte è la tendenza a creare significati alternativi, alle istituzioni formali economiche, giuridiche, già esistenti[25].

2.4 Stile di sviluppo dei progetti.

Per completare la presentazione delle 'regole del gioco' informali della comunità hacker, occorre illustrare il funzionamento dei progetti di sviluppo di Software Libero non sovvenzionati[26].

Eric Raymond in 'La cattedrale e il bazaar' (1997), racconta la sua esperienza con un progetto[27], avviato come verifica sperimentale delle teorie sul Software Libero da lui isolate analizzando le caratteristiche e l'andamento del progetto GNU/Linux (kernel).
Quest'ultimo è considerato 'rivoluzionario'; si discosta solo in parte dalle regole seguite nel passato, ma, i cambiamenti introdotti, si sono rivelati decisivi per far uscire questo 'mondo' dalla marginalità. È il capostipite di una nuova generazione di progetti, per la sua importanza, ampiezza e perché per primo ha sfruttato pienamente tutte le possibilità messe in campo dalle nuove tecnologie, dal crollo dei prezzi dell'hardware, dalla diffusione del personal computer e dalla facilità di comunicazione garantita da Internet.
Raymond mette a confronto gli stili di sviluppo che chiama a 'cattedrale', tipico del software proprietario, e lo stile 'bazaar', adottato per il Software Libero. Quest'ultimo risulta molto variabile da caso a caso, ma in linea di massima, possiede le caratteristiche, sintetizzate da Raymond in 19 aforismi, ed intenzionalmente riprodotte nel suo esperimento.
1. Every good work of software starts by scratching a developer's personal itch.[28]
Si parte sempre dal bisogno di un programmatore che vuole un nuovo strumento utile per fare qualcosa o per risolvere un problema interessante. Non trovando nulla che lo soddisfi pienamente, se lo scrive da sé o modifica un programma già disponibile in rete. Se ritiene di aver fatto un buon lavoro o che possa essere utile ad altri, lo diffonde con il codice sorgente, gratis e senza restrizioni, in modo che possa essere usato dal maggior numero possibile di persone. In pratica a questo punto ha già avviato un progetto.
In molti progetti Open Source, spiega Paul Vixie[29], i programmi vengono semplicemente scritti e distribuiti. Invece, il software proprietario, prima di questo momento attraversa dei passaggi obbligati. Innanzitutto, viene stabilita la fascia di mercato della clientela, l'uso e le caratteristiche del programma, il viene riassunto in un documento, il Marketing Requirements Document (MRD), che passa ai progettisti

[30]. Questi in base all'MRD producono una descrizione del prodotto a livello di 'moduli' e dell'interazione tra di essi, detta 'Progetto a livello di sistema' e in seguito fanno un 'Progetto dettagliato' di ogni modulo. A questo punto, è possibile stimare più precisamente le risorse umane ed economiche richieste dal progetto e, se nulla osta, si passa all'implementazione dei singoli moduli e alla loro integrazione, cioè alla scrittura vera e propria del software, fino ad ottenere qualcosa di funzionante conforme alle specifiche iniziali.
Nel Software Libero tutti questi passaggi si saltano. Sarebbe inutile sprecare risorse economiche e di tempo per progettare sulle esigenze dell'ipotetico 'cliente' dato che è l'utente stesso che scrive da solo (cooperando con gli altri), il prodotto di cui ha bisogno. E comunque i progetti evolvono in modo non pianificabile a priori o pianificabile solo in parte.
La programmazione è fatta per puro piacere e per dimostrare ed esibire la propria competenza tecnica, "il che spiega l'alta qualità media del software originato dalla comunità Linux"[31].
2. Good programmers know what to write. Great ones know what to rewrite (and reuse).
È meglio iniziare da un prodotto non pienamente soddisfacente che da zero. Una delle caratteristiche dei programmatori è il forte senso pragmatico e secondo Raymond una 'sorta di ozio costruttivo'.
Va tenuto presente che occorre radunare un adeguato numero di persone intorno ad un progetto. È impensabile che questo possa accadere senza che vi sia già del materiale su cui lavorare. Anche Linus Torvalds ha cominciato a scrivere Linux partendo dal sistema operativo Minix, un piccolo sistema per uso didattico, simile a Unix. La tradizione degli hacker, legati al sistema operativo Unix, di condividere i codici, favorisce la ricerca ed il recupero di materiale già pronto e disponibile per l'implementazione.
3. "Plan to throw one away; you will, anyhow." (Fred Brooks, The Mythical Man-Month, Chapter 11).
Non sempre la prima soluzione trovata e su cui si sta lavorando è la migliore oppure ci si può trovare in un vicolo cieco e non riuscire a trovare la via d'uscita. La migliore strategia, in questi casi, è di implementare una nuova soluzione ricominciando sul codice di qualcun altro, mettendo a frutto l'esperienza del primo tentativo.
4. If you have the right attitude, interesting problems will find you.
L'atteggiamento giusto è quello di mantenere la mente aperta nei confronti di qualsiasi problema e soluzione possibile in modo da ricombinarli creativamente per uscire dalle difficoltà. Anche a costo di buttare via tutto come al punto 3.
5. When you lose interest in a program, your last duty to it is to hand it off to a competent successor.
Chi avvia e gestisce un programma è responsabile nei confronti della comunità di cosviluppatori e utenti che si viene a creare intorno ad esso. Non si può uscire dal proprio progetto semplicemente disinteressandosi. È un preciso obbligo quello di 'passare il testimone' se si presenta qualcuno che si dimostra interessato al suo mantenimento ed è all'altezza per farlo.
6. Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.
Avviato il progetto e distribuito il software, il programma comincia ad avere degli utenti. Se è stato 'ereditato', un gruppo di utenti già esiste. La loro presenza testimonia l'utilità del lavoro svolto e che si sta rispondendo ad un loro bisogno. Spesso gli utenti sono essi stessi degli hacker, che 'coltivati' nel modo giusto possono trasformarsi in co-sviluppatori. Essendo i sorgenti disponibili senza vincoli, gli utenti competenti saranno in grado di dare una mano a migliorare il codice suggerendo soluzioni o scovando errori e malfunzionamenti. Ogni componente del software richiede nel tempo una costante opera di aggiornamento e miglioramento, che avviene in modo circolare. Pertanto, maggiore è il numero di sviluppatori attivi coinvolti più è rapida l'evoluzione.
Prima dell'avvento di Linux si riteneva, al contrario, che fossero i problemi ad aumentare in modo esponenziale al crescere delle persone coinvolte e della complessità - a maggior ragione per un sistema operativo. La 'Rivoluzione' del metodo di sviluppo, attuata da Linus Torvalds, ha dimostrato che il miglioramento del codice ed il debugging efficace, crescono di pari passo col crescere degli utenti e della complessità del sistema e ha spazzato via i pregiudizi esistenti sulla gestione di progetti complessi. Pregiudizi sorretti anche dai vincoli tecnici presenti fino all'inizio degli anni '90. Come ogni 'rivoluzione' ha dovuto attendere le condizioni necessarie per poter 'esplodere': l'innovazione tecnologica con conseguente crollo dei prezzi dell'hardware, la diffusione capillare del personal computer e la facilità di comunicazione garantita da Internet.
In queste condizioni è più facile raggiungere la 'massa critica' necessaria al decollo di un progetto. Più facile ma non deterministico. Il software libero non è una magia valida per tutti i progetti. La fase iniziale della ricerca di utenti e sviluppatori attivi è la più delicata e non tutti i progetti decollano, anzi molti falliscono e restano immobili.
7. Release early. Release often. And listen to your customers.
Fattore cardine del metodo usato da Torvalds è la rapida e frequente distribuzione delle varie release (versione di un programma). Un altro pregiudizio molto forte nel passato era che fosse inopportuno

distribuire subito perché le prime versioni di un programma sono piene di bug. Inoltre si tendeva a seguire il metodo 'a cattedrale' con un attenta progettazione ed implementazione prima del rilascio della prima versione attentamente riveduta e corretta in modo da far vedere meno bug possibili agli utenti.
Il sistema della rapida diffusione è più efficace ed efficiente ed assicura un rapido sviluppo, ma i pochi che adottarono questo modello prima di Linux non se ne resero pienamente conto. Torvalds in effetti ha portato questa pratica ad un livello estremo di intensità, tanto che nei primi tempi (1991/92), diffondeva le versioni del kernel anche parecchie volte al giorno.
Prestare ascolto ai propri utenti è un altro punto fondamentale. Torvalds trattava gli utenti come co-sviluppatori, tendeva a stimolarli e ricompensarli. Tanto che sono stati loro a coinvolgere altre persone con un passaparola tanto efficace da far raggiungere a Linux la 'massa critica' in brevissimo tempo.
8. Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone. Or, less formally, "Given enough eyeballs, all bugs are shallow." I dub this: "Linus's Law".
Direttamente legato all'aforisma n° 6. La persona che focalizza un problema, osserva Linus, non necessariamente è la stessa che lo risolve, e la cosa più difficile è proprio trovarlo.
Nel modello bazaar i bug diventano in fretta fenomeni marginali se esposti all'attenzione di migliaia di co-sviluppatori. In questo modo aumenta la probabilità che qualcuno trovi la maniera più adatta, tra le molte possibili, di percepire il problema e riesca ad inquadrarlo correttamente. Questo è il motivo della scelta della rapida diffusione: ottenere veloci correzioni e di conseguenza una rapida evoluzione del progetto. Il costante monitoraggio del lavoro degli altri evita che vi siano sovrapposizioni. Si agisce seguendo un processo di sviluppo distribuito che utilizza specifiche tecniche e strumenti di gestione delle conoscenze utili al progetto sempre disponibili e aggiornati per poter cooperare in rete. In primo luogo si usa un insieme di programmi che permette ai partecipanti sparsi in tutto il mondo di essere creatori paralleli di un programma senza ostacolarsi a vicenda; consentono anche di rintracciare tutte le modifiche apportate nel corso del tempo, e chi le ha fatte[32]. Questi programmi gestiscono anche opportuni file che accolgono tutti i maggiori argomenti dove occorre uno scambio di idee con i relativi interventi e, se occorre, le votazioni per decisioni da prendere in gruppo. Altro elemento fondamentale è la presenza di opportuni forum di discussione per gli utenti e per gli sviluppatori, suddivisi per argomenti, in modo da poter selezionare solo ciò che interessa e restare aggiornati sullo sviluppo in corso senza eccedenze di messaggi.
Il processo richiede il coordinamento di uno sviluppatore che amministri le comunicazioni tra coloro che si occupano dell'attività di debugging ma l'attività in sé non necessita di coordinamento. È così che, nonostante la notevole complessità del sistema, non se ne cade preda, e si evita il 'costo' supplementare per gestire il coinvolgimento di nuove persone.
Il costante feedback degli utenti-sviluppatori e l'accorgimento di 'distribuire presto e spesso' minimizza la duplicazione del lavoro di coloro che si occupano del debugging e consente una migliore auto-distribuzione dei compiti evitando la perdita di efficienza che teoricamente dovrebbe sorgere all'aumento della complessità. Come rileva lo stesso Raymond, questa metodologia di lavoro, corrisponde grossomodo all''Effetto Delfi'[33]
10. If you treat your beta-testers as if they're your most valuable resource, they will respond by becoming your most valuable resource.
Non basta avere numerosi utenti, bisogna essere in grado di trattare con loro. Lo stile di leadership risulta qui fondamentale. Trattandosi di volontari non si può comandar loro di fare qualcosa ma bisogna attendere che lo facciano spontaneamente incoraggiando ed indirizzando la partecipazione. Raymond nel suo 'esperimento' ha inserito nella lista dei beta tester[34] chiunque lo abbia contattato riguardo al suo progetto, ha mandato simpatici messaggi all'intera lista in occasione di ogni nuova release, incoraggiando la partecipazione e facendo domande sul design adottato. Il riscontro è stato immediato con numerosi report di bug e ottime soluzioni per risolverli oltre a molte mail piene di critiche costruttive, lodi sperticate, suggerimenti intelligenti. In cambio dopo ogni feedback è stato prodigo di apprezzamenti e lodi.
11. The next best thing to having good ideas is recognizing good ideas from your users. Sometimes the latter is better. 12. Often, the most striking and innovative solutions come from realizing that your concept of the problem was wrong.
Onestà ed autocritica, devono essere costantemente dimostrate da parte del leader del progetto insieme ad una elevata dose di competenza tecnica e diplomazia. Il leader deve essere in grado di riconoscere le buone idee e le soluzioni innovative da chiunque provengano, anche ammettendo che le proprie erano sbagliate o poco funzionali. Conquistando in questo modo sempre maggior fiducia da parte dei co-sviluppatori e più partecipazione; osserva Raymond "il mondo intero ti tratterà come se ogni bit di quell'invenzione fosse opera tua, mentre impari a considerare con sempre maggior modestia il tuo genio innato[35]". Oppure Linus Torvalds: "il problema era capire come dire in modo diplomatico a una persona

che preferivo i cambiamenti suggeriti da qualcun altro. [...] Non ho mai pensato di accettare soluzioni che non mi sembrassero le migliori da un punto di vista tecnico. [...] Era anche un modo per portare la gente a fidarsi di me. [...] Quando la gente si fida di te, accetta i tuoi consigli [36]".
Il leader prende tutte le decisioni finali sull'implementazione, può non essere un bravo progettista ma è fondamentale che sia in grado di riconoscere le migliori idee tra quelle proposte dagli altri. "Come non avevo mai progettato per Linux una vita al di fuori del mio computer, non avevo nemmeno mai deciso di essere il leader. [...] Imparai abbastanza presto che il modo migliore e più efficace per essere un leader consiste nel lasciare che le persone facciano le cose perché le vogliono fare, e non perché tu vuoi che le facciano. I migliori leader sanno quando si sbagliano e [...] mettono gli altri in grado di prendere delle decisioni al posto loro"[37].

Gli aforismi n° 9 e quelli dal n° 13 al n° 19 riguardano disposizioni prettamente tecniche, oppure ripropongono punti già commentati, li riporto in nota con relativa traduzione[38].

Da quanto detto si possono focalizzare tre elementi essenziali che condizionano l'esistenza ed il successo di un moderno progetto di Software Libero: infrastruttura comunicativa (Internet), la leadership e la comunità.
Internet, rispetto ai precedenti sistemi di comunicazione, è capillare, facilmente accessibile, veloce e poco costosa. Ha trasformato il modo di comunicare buona parte dei possessori di computer nel mondo ed ha aperto il 'campo di gioco' delle comunità verso l'esterno - a partire dalla seconda metà degli anni '90 c'è stata un'impennata nel numero di sviluppatori che si sono uniti alla comunità - dando nuova vita ad un metodo di sviluppo che poggia, come si è visto, le radici nel passato. 'Luogo' di incontro e discussione può essere una semplice mailing-list o un newsgroup o altri forum di discussione pertinenti (ospitati ad esempio presso il sito Internet di una organizzazione che gestisce i programmi più grandi e strutturati) dove lo scambio ha un livello minimo di formalità e dove è possibile verificare se qualcuno è disposto a partecipare attivamente.
Il leader è colui che avvia il progetto, coordina i co-sviluppatori e gestisce lo sviluppo prendendo tutte le decisioni riguardo le parti nuove di codice e le modifiche da includere nel progetto e infine si occupa di redistribuire le versioni modificate. Solo in parte 'conduce il gioco' per il resto deve fare da 'arbitro'. Non sempre è l'autore originario del software su cui la comunità di utenti si raduna. Sia Torvalds sia Raymond, per tornare agli esempi presentati, hanno lavorato su materiale già esistente.
Al momento della presentazione pubblica, il codice da lui diffuso deve essere ad un giusto grado di sviluppo. Lo stile 'bazaar' non consente che si possa cominciare da zero, in quanto mancherebbe un oggetto da migliorare o con cui 'giocare'. Può essere anche scarno, incompleto e con molti errori ma deve essere convincente, rappresentare cioè una promessa credibile per gli sviluppi futuri.
Dallo stile di programmazione è possibile verificare le competenze e perfino la personalità del programmatore che lo propone. La nascita della comunità di sviluppatori non è un evento certo e dipende moltissimo dalle caratteristiche personali del leader: competenza, pragmatica, diplomazia, apertura, onestà, autocritica, comunicativa. In mancanza di queste caratteristiche, il progetto non supera la fase critica iniziale e non comincia ad evolvere, oppure si biforca, dando vita a diverse comunità ciascuna con meno risorse e meno possibilità di sopravvivenza.
Il terzo elemento è dato dalle persone che scelgono volontariamente di unirsi e di cooperare, dando vita alla comunità che ruota intorno al progetto ed al suo leader. La domanda, ora, è cosa spinge queste persone a collaborare, ossia, quali sono gli incentivi, le aspettative e l'orientamento degli sviluppatori di Software Libero.
Dai risultati dell'indagine FLOSS[39], emerge che la scelta di unirsi alla comunità è prevalentemente indotta dal desiderio di migliorare le proprie capacità e competenze di programmazione e dalla possibilità di scambiare conoscenze ed informazioni con altri sviluppatori, con la certezza di ricevere più di quanto si è dato
. In secondo luogo dal desiderio di partecipare ad una nuova forma di cooperazione con la possibilità di migliorare i prodotti degli altri sviluppatori. L'attenzione verso gli aspetti materiali (ad esempio migliori opportunità di lavoro) viene maturata con il passare del tempo e la frequentazione della comunità, ma queste motivazioni rimangono secondarie rispetto a quelle di ingresso. Inoltre, l'aspettativa che anche gli altri mettano in condivisione conoscenze e competenze è estremamente elevata. "Potresti chiederti: perché la gente dovrebbe spendere ore del proprio tempo a scrivere software, a metterlo insieme attentamente, e poi darlo tutto via? Le risposte sono varie quanto le persone che contribuiscono. Ad alcune persone piace aiutare gli altri. Molte scrivono programmi per imparare di più sui computer. Sempre più gente cerca vie per evitare il prezzo gonfiato dei software. Una moltitudine in crescita contribuisce come ringraziamento per tutto il software che hanno ricevuto. Nelle università molti creano software free

per ottenere i risultati delle loro ricerche in un utilizzo più ampio. Compagnie commerciali aiutano a mantenere il free software per avere voce su come procede lo sviluppo - non c'è maniera più rapida di ottenere una nuova caratteristica che implementarla da solo! Di sicuro, molti di noi lo trovano semplicemente un grosso divertimento"[40].

2.5 Cultura hacker e Capitale Sociale.

A questo punto, tutti i più importanti elementi e regole informali presenti nel 'campo di gioco' sono stati raccolti. Quindi, è possibile vedere ciò che distingue strutturalmente e qualitativamente la programmazione a 'Bazaar' da quella a 'Cattedrale': la quantità di 'Capitale Sociale' presente nelle comunità di programmatori del Software Libero.
L'idea di Capitale Sociale costituisce una sistemazione teorica elaborata da Coleman[41] a partire dal concetto di 'capitale umano' introdotto da Becker[42] ed altri autori. Il capitale umano si identifica in uno stock di risorse per l'azione, rappresentate dalle abilità e conoscenze acquisite dall'individuo. Quando due o più persone stabiliscono un rapporto anche debole, lo stock di risorse entra nella struttura della loro relazione. Il capitale sociale è quindi quel potenziale di risorse, utili alle proprie strategie d'azione, a disposizione dei singoli partecipanti al network di relazioni in cui sono inseriti. È intimamente connesso alla struttura della relazione.
Un secondo sguardo ai vincoli informali delle comunità del Software Libero, rielaborati attraverso la 'Teoria della società' di Coleman e quelle che egli nomina come forme efficienti di capitale sociale osservabili nei network, permette di valutare il fenomeno da questo punto di vista.
La prima forma è il 'Debito di riconoscenza con aspettativa di restituzione' o 'credit slip'; in sostanza, un credito esigibile non per diritto ma per affidabilità o riconoscenza del debitore.
Le persone che si uniscono alle comunità in questione, possono attingere ad una molteplicità impensabile di software pronto, disponibile, e ricevono un grosso aiuto dagli altri membri per comprenderne a fondo il funzionamento sia come utenti ad ogni livello, sia come programmatori.
Lo scambio di prodotti, consigli, strumenti di sviluppo, ecc., segue, in primo luogo, la logica del dono, con i suoi tre imperativi: obbligo di dare, di ricevere e di ricambiare[43]; ma gli oggetti donati hanno una natura del tutto inconsueta rispetto ad altri beni, perché non sono consumabili, sono riproducibili a costo quasi zero e la diffusione ne aumenta il valore intrinseco, pertanto, ogni singolo dono fa aumentare la 'dote' di tutta la comunità. In secondo luogo assume le caratteristiche del dono moderno[44], vale a dire, che donatori e donatari possono anche essere degli estranei, a differenza di quello arcaico che esigeva un legame sociale stretto (come nella famiglia o nel villaggio).
La convinzione diffusa di aver ricevuto più di quanto si è dato[45] è indice di 'riconoscenza' nei confronti della comunità. Tanto che, nonostante la restituzione permane la sensazione di essere ancora in debito.
Altra forma efficiente di capitale sociale riguarda il 'Potenziale informativo' che è proprio delle relazioni sociali; la sua importanza è data dal fatto che all'aumentare dell'informazione si hanno più elementi a supporto del buon esito dell'azione.
Le comunità hacker sono dei network in cui la maggior parte dell'informazione circola in tempo reale e ad un costo che tende allo zero. Il passaggio delle informazioni non è limitato alle questioni di utilizzo del software e alla programmazione, ma, anche se in minor misura, riguarda qualsiasi ambito di competenza dei partecipanti (di istruzione media molto elevata[46]). Questo, come si è visto nel precedente capitolo, avviene perché la libertà di informazione è elevata a principio cardine che informa il comportamento generale di tutta la comunità.
Coleman prosegue con le 'Regole e sanzioni efficaci'. Una regola che costituisce un importante forma di capitale sociale all'interno di una comunità è quella che impone di rinunciare ai propri interessi per agire nell'interesse della comunità. Essa è più efficace se sorretta da pressioni collettive all'applicazione come approvazione sociale, status, onore ed altre ricompense e da sanzioni di senso contrario in caso di devianza.
Egli fa notare che regole efficaci, in certi ambiti, possono anche ridurre l'innovazione. Questa trappola viene evitata dalle comunità del Software Libero, perché le loro norme, come abbiamo visto, corrispondono a quelle delle comunità scientifiche, che per loro natura si rivolgono spontaneamente verso l'innovazione. La rinuncia ad un tornaconto immediato e personale a favore dell'interesse collettivo dei programmatori Free, è compensata dal riconoscimento pubblico per il lavoro fatto, che può tradursi in vantaggi materiali (ad esempio migliori opportunità lavorative), anche se incerti e posticipati, ma non è questo il primo pensiero dei partecipanti.

La quarta forma riguarda i 'Rapporti di autorità', ossia il trasferimento da un attore ad un altro di diritti di controllo. L'investitura di un leader carismatico è una forma di creazione di capitale sociale ed il leader di un progetto di sviluppo di Software Libero rientra senza dubbio, viste le sue caratteristiche, in questa categoria.
La quinta forma è l'Organizzazione sociale appropriabile'. Con questo si intende la possibilità di dirigere una rete di relazioni, nata con uno scopo, verso altri nuovi obiettivi. Nel caso in oggetto ad esempio troviamo l'impegno per l'allargamento dell'etica hacker ad altri ambiti del sapere; il sostegno della libertà di espressione; le mobilitazioni contro la guerra o la pedofilia su Internet. In questi casi, tra l'altro, tendono a coinvolgere altri gruppi presenti sullo stesso medium e, di conseguenza, ad allargare il capitale sociale.
Infine, come ultima forma, Coleman riporta le 'Organizzazioni intenzionali', intendendo con questo, le organizzazioni in senso stretto, quindi, forme di capitale sociale che sono l'esito diretto dell'investimento degli attori allo scopo di ottenere un qualche risultato. Tra le altre vi sono le associazioni di volontari che producono beni pubblici. È esattamente il caso delle organizzazioni per la gestione dei progetti di sviluppo del Software Libero e delle organizzazioni a sostegno della sua diffusione come la Free Software Foundation o la Open Source Organization[47]

È evidente che il Capitale sociale si distingue nettamente dagli altri beni. Ha le qualità di bene pubblico: è inalienabile, è una risorsa con valore se in uso, non è facilmente sostituibile, non è proprietà delle persone che ne beneficiano.
Vi sono alcuni fattori che ne influenzano la creazione, conservazione e la scomparsa.
Fattori che aiutano la creazione e la conservazione del capitale sociale sono: la chiusura delle relazioni - nel senso di connessioni dirette tra i partecipanti al network -, la stabilità delle relazioni, un ideologia non individualista e incentivante l'aggregazione e la dipendenza reciproca.
L'ambiente del Software Libero è caratterizzato dalla possibilità di connettersi con ogni partecipante il network; dal punto di vista della sistemica è da considerarsi un sistema di tipo aperto (con collegamenti anche verso l'esterno), tanto che ogni partecipante ha rapporti diretti con una piccolissima parte del potenziale a disposizione.
La struttura del network è duratura nel tempo, con relazioni stabili tra individui, con posizioni fungibili al livello delle organizzazioni e con relazioni occasionali con altri membri, infatti, le richieste di aiuto possono essere rivolte pubblicamente alla comunità ricevendo supporto il più delle volte in forma anonima.
L'ideologia insiste fermamente sulla condivisione di conoscenze, capacità e mezzi. Infine, il campo del sapere implicato, è talmente vasto che comunque tutti, anche i più dotati, si trovano ad aver bisogno degli altri per le proprie difficoltà.
La valutazione di queste caratteristiche porta a pensare che sia più probabile la conservazione, consolidamento ed ulteriore espansione del fenomeno, piuttosto che la sua scomparsa.

2.6 Conclusioni

Il quadro delle regole informali si può a questo punto considerare completo, ma, la comprensione delle dinamiche evolutive, così come indicato nella struttura teorica del North, esige che si consideri il gioco del formale e dell'informale nelle strutture d'azione concrete.
Nei prossimi capitoli, di conseguenza, l'esposizione sarà dedicata alle regole formali precedute da un approfondimento delle caratteristiche proprie del software in quanto oggetto delle disposizioni, poi, alle organizzazioni di supporto per passare, infine, all'interazione di tutti gli elementi considerati nel 'campo di gioco' del Mercato e alla loro rilettura attraverso il North.

****

1 Merton, 1973; trad. it., 1981.
2 Kuhn, 1962.
3 Un sistema chiuso risulta isolato dal proprio contesto, mentre in quelli aperti vi è permeabilità con l'ambiente, con scambio di materia (tipico nei sistemi viventi).
4 Nupedia è un progetto per un'Enciclopedia Libera (fuso con il progetto GNUpedia). È un movimento simile al movimento del Software Libero per sviluppare un'enciclopedia universale libera, che copra tutti i campi della conoscenza, ed una biblioteca completa di corsi per la formazione. Licenziato sotto GFDL (GNU Free Documentation License: http://www.gnu.org/copyleft/fdl.html), la Licenza per la libera

documentazione: http://www.nupedia.com/ . Un progetto analogo è Wikipedia, più attivo perché meno strutturato, anch'esso licenziato sotto GFDL: http://www.wikipedia.com/,.
5 Il progetto si propone di ricercare ed indicare tutte quelle strutture (italiane e non) che vendono copie cartacee stampate o fotocopiate di documentazione tutelata da licenze libere
6 Home page del progetto: http://eon.law.harvard.edu/openlaw/
7 Raymond, 1996.
8 Max Weber, 1919.
9 Merton, 1959. Il concetto di Serendipity (o serendipità), fu coniato nel 1754 dallo scrittore inglese Horace Walpole che lo trasse dal titolo di una fiaba (Serendip era l'antico nome dell'isola di Ceylon). La Serendipity descrive quel processo - assai comune anche nella vita quotidiana - che porta a scoperte inaspettate cui si giunge mentre si cercava, si pensava, o si sperimentava, in tutt'altra direzione e con tutt'altri fini.
10 Raymond, 1996, [Esiste una comunità, una cultura condivisa, di programmatori esperti e maghi della rete per trovare le origini della quale dobbiamo andare indietro nei decenni, fino ai tempi dei primi minicomputer funzionanti in time-sharing o dei primi esperimenti con ARPAnet. I membri originari di questa 'civiltà' generarono il termine 'hacker'. Gli hacker costruirono Internet. Gli hacker costruirono il sistema operativo Unix così come lo conosciamo oggi. Sempre grazie agli hacker abbiamo avuto Usenet e il World Wide Web. Se tu sei parte di questa cultura, se hai dato il tuo contributo e altre persone che ne fanno parte ti conoscono e ti chiamano hacker, tu sei un hacker].
11 Una Sinergia è l'effetto complessivo che consegue dall'attività simultanea di vari organi, funzioni, processi; l'effetto è di tipo olistico, per cui la somma delle parti dà un risultato superiore alla somma stessa, ovvero, l'effetto globale è maggiore di quello strettamente addizionale.
12 Danno assai maggiore può causare colui che si appropria di software di pubblico dominio, cioè senza copyright. Questo, con poche modifiche, può essere reimmesso sul mercato con licenza proprietaria e senza sorgenti, privando l'autore del merito per la paternità del prodotto, privando la comunità delle modifiche e dei miglioramenti e creando una biforcazione. Proprio per evitare questo, le comunità del software libero hanno adottato la GNU/GPL.
13 Hersey e Blanchard, 1982; trad. it., 1984.
14 http://www.tuxedo.org/~esr/jargon/jargon.html# 'The on-line hacker Jargon File, version 4.3.1, 29 JUN 2001'
15 Merita una citazione una iniziativa denominata Open Science, raggiungibile al sito Internet www.openscience.org. Si tratta di un gruppo di scienziati che muovendo dalla constatazione che la scienza moderna si basa in misura sempre più larga su simulazioni al computer, su modelli computazionali, e su analisi computazionale di grandi insiemi di dati, invita la comunità scientifica a muoversi per pretendere che i software utilizzati per simulare sistemi complessi vengano resi disponibili nel formato sorgente. I motivi sono connessi con l'applicazione del principio del carattere pubblico della scienza. Le teorie che vengono sviluppate con metodi di simulazione al computer sono basate su assunzioni verificabili in linea di principio; ma la verificabilità pratica dei calcoli non è possibile senza avere il libero accesso al codice sorgente del software utilizzato. Metodo Open Source in questo caso è condizione necessaria per garantire il rispetto da parte di un settore importante della comunità scientifica del principio di pubblica verificabilità della scienza.
16 Merton, 1973; trad. it., 1981. (pagg. 349-359)
17 Termine infelice per via della connotazione politica che immediatamente evoca. Molti di coloro che fanno riferimento agli scritti di Merton usano il termine 'Comunitarismo', certamente più neutrale e forse anche più vicino a ciò che egli intendeva.
18 Op. cit. pag. 352
19 Op. cit. pag. 355
20 Ibid.
21 Op cit. pag. 356.
22 Ma, permangono delle divergenze sull'atteggiamento nei confronti della proprietà intellettuale, in quanto coinvolge il problema del riconoscimento economico degli sforzi per la ricerca.
23 GNU/GPL e questione dei brevetti verranno approfondite nel prossimo capitolo.
24 Op. cit. pag. 358.
25 Emblematica in questo senso è la licenza GNU/GPL che approfondirò nei prossimi capitoli.
26 Quelli sovvenzionati hanno caratteristiche ibride e si discostano dall'idealtipo che cercherò di tratteggiare.
27 Si tratta di Fetchmail, un'utility per "scaricare la posta" da server di posta remoti e renderla disponibile al sistema locale. Sito del progetto: http://www.tuxedo.org/~esr/fetchmail/

28 Questo aforisma è valido anche inteso in senso collettivo. In questo caso il progetto sarà avviato da una comunità per colmare la mancanza di elementi di utilità comune.
29 P. Vixie, in DiBona, Ockman, Stone, 1997.
30 Questo passaggio non è privo di attrito. È nota l'atavica conflittualità tra reparti marketing e R&S, non solo nell'ambito delle aziende informatiche.
31 Raymond, 1997.
32 Il più importante di essi è il CVS (Current Versioning System).
33 Metodo Delfi. Il metodo Delfi prende il nome dalla città greca in cui l'Oracolo di Apollo esprimeva le sue predizioni sul futuro. Infatti la funzione principale di questo metodo è quella di permettere a un gruppo (panel) di esperti o di testimoni privilegiati di formulare ed esplorare scenari di cambiamento sociale, tecnologico e scientifico. E' stato sviluppato dalla Rand Corporatoin negli anni sessanta per la Air Force. Il metodo permette di strutturare la comunicazione tra gli esperti e, per mezzo di un meccanismo di feedback ripetuto, di correggere reciprocamente previsioni, giudizi, valutazioni, fino a far convergere le opinioni su una sintesi finale. La sua finalità è di essere di sussidio alle presa di decisioni ma evidentemente può essere utilizzato anche come strumento per la progettazione della ricerca. Il metodo Delfi è particolarmente adattabile alla CMC (Computer Mediated Communication).
34 Utenti dei programmi beta impegnati nella revisione di un programma.
35 Ibid.
36 Torvalds, Diamond, 2001.
37 ibid.
38 9. Smart data structures and dumb code works a lot better than the other way around. Una struttura dati ben fatta e un codice scadente funzionano molto meglio che non il contrario. 13. "Perfection (in design) is achieved not when there is nothing more to add, but rather when there is nothing more to take away." "La perfezione (nel design) si ottiene non quando non c'è nient'altro da aggiungere, bensì quando non c'è più niente da togliere."[Questione tecnica che richiama da vicino la regola hacker 'Semplificazione ordine, riduzione e ottimizzazione delle risorse'. La frase riportata come aforisma n° 13 è di Antoine de Saint-Exupéry (aviatore e designer di aerei, quando non scriveva libri per bambini)]. 14. Any tool should be useful in the expected way, but a truly great tool lends itself to uses you never expected. Ogni strumento dovrebbe rivelarsi utile nella maniera che ci si attende, ma uno strumento davvero ben fatto si presta ad utilizzi che non ci si aspetterebbe mai. [È un richiamo a fenomeni come quello della serendipità e della ricombinazione creativa delle conoscenze già visti nel capitolo precedente]. 15. When writing gateway software of any kind, take pains to disturb the data stream as little as possible and never throw away information unless the recipient forces you to! Quando si scrive del software per qualunque tipo di gateway, ci si assicuri di disturbare il meno possibile il flusso dei dati - e *mai* buttar via alcun dato a meno che il destinatario non ti ci costringa! 16. When your language is nowhere near Turing-complete, syntactic sugar can be your friend. Quando il linguaggio usato non è affatto vicino alla completezza di Turing, un po' di zucchero sintattico può esserti d'aiuto 17. A security system is only as secure as its secret. Beware of pseudo-secrets. Un sistema di sicurezza è sicuro soltanto finché è segreto. Meglio diffidare degli pseudo-segreti. 18. To solve an interesting problem, start by finding a problem that is interesting to you. Per risolvere un problema interessante, comincia a trovare un problema che risvegli il tuo interesse. 19. Provided the development coordinator has a communications medium at least as good as the Internet, and knows how to lead without coercion, many heads are inevitably better than one. Stabilito che il coordinatore dello sviluppo abbia a disposizione un medium almeno altrettanto affidabile di Internet, e che sappia come svolgere il ruolo di leader senza costrizione, molte teste funzionano inevitabilmente meglio di una sola.
39 FLOSS: 'Free/Libre and Open Source Software: Survey and Study'; prima inchiesta su vasta scala commissionata dalla Comunità Europea con lo scopo di fare luce su tutti gli aspetti di questo fenomeno.
40 http://www.debian.org/intro/about
41 Coleman, 1990. (pp. 300-321)
42 Becker, 1964.
43 Mauss, 1950.
44 Godbout, 1992.
45 Survey FLOSS, 2002.
46 Il 70% degli sviluppatori di Software Libero possiede una laurea o un titolo superiore (master - phd). Survey FLOSS, 2002.
47 Che tratterò in un prossimo capitolo.

3 Il Software Libero e le istituzioni formali.

3.1 La natura del software.

Prima di introdurre le regole formali che lo riguardano direttamente - copyright o licenze d'uso, brevetti, leggi di tutela a livello nazionale e accordi internazionali - occorre conoscere meglio l'effettiva 'natura' del software.

Uno dei caratteri essenziali nelle società moderne e 'occidentali', generalmente accettato anche da sociologi ed economisti, è l'importanza sempre più strategica dell'informazione, intesa come risorsa, ovvero, della capacità di produrla, elaborarla e distribuirla. E il software è informazione.
Il termine informazione definisce una risorsa tipicamente astratta, immateriale, che riguarda la manifestazione delle idee e come le altre risorse è un mezzo capace di soddisfare dei bisogni. È costituita da conoscenze e notizie prodotte e trasmesse in modi e con supporti eterogenei. Può, prendere forma nei modi più svariati come libri, immagini, musica, film, programmi televisivi, formule matematiche, e ovviamente anche software[1].
Tra i tanti appellativi, la nostra società, è stata denominata 'società dell'informazione'. Questo sottolinea ulteriormente il ruolo prezioso di questa risorsa nella nostra società, e lo è anche dal punto di vista economico. L'informazione ha sempre avuto una posizione importante, ma oggi, è divenuta centrale e strategica rispetto a quella delle altre risorse economiche.
Da qui l'esigenza di tutelarla e regolamentarla meglio, compito non facile, a causa del criterio seguito che si basa sulla territorialità[2]. Ulteriore complicazione è data dal fatto che, il più delle volte, i sistemi di tutela degli stati nazionali, sono delle riletture e adattamenti di vecchi strumenti, nati in epoca priva di tecnologie digitali in cui l'informazione era incorporata in supporti la cui produzione era costosa e la diffusione circoscritta.
La tecnologia digitale è completamente nuova rispetto ai riferimenti dei legislatori, da qui la difficoltà di trovare delle normative adatte a 'prodotti' che possono essere dislocati in ogni punto del globo terrestre, in tempo reale, a costo irrisorio (escludendo l'attrezzatura) e in copie che risultano identiche all'originale.

Prima di affrontare più in dettaglio il tema delle normative, vale a dire delle istituzioni formali, occorre focalizzare un altro problema che complica ulteriormente il quadro della situazione: i costi di produzione (implementazione) del software.

3.2 Costi di produzione del software.

Il software è un prodotto costituito da un insieme di istruzioni digitali. Come gli altri prodotti classificabili come informazione è immateriale e per essere utilizzato, prodotto e accumulato deve essere incorporato in un supporto. Tra supporto e prodotto c'è una stretta integrazione, ma vanno tenuti logicamente distinti sia per comprendere i problemi inerenti alla sua produzione sia per quelli che sorgono per la sua tutela giuridica.
La struttura dei costi di produzione del software è decisamente singolare.
A differenza dei prodotti industriali 'classici', esso è affetto da una diseconomia di scala dei costi di sviluppo rispetto alla dimensione del prodotto, in altre parole, all'aumentare della dimensione del programma, cresce anche la sua complessità, e lo fa in modo esponenziale, quindi anche il suo costo cresce allo stesso modo. Se ad esempio si raddoppiano le dimensioni di un programma, la sua complessità va praticamente elevata al quadrato, ma, quel che è peggio, è l'incremento della 'difettosità', più che proporzionale rispetto alla complessità, quindi, aumentano allo stesso modo anche i tempi di verifica (test) e di correzione di bug o altre imperfezioni[3].
I costi di progettazione e di sviluppo marciano paralleli a questi aumenti e non sono comunque facilmente stimabili in anticipo soprattutto per i bug, difficili da individuare oltre che da correggere. Inoltre, non esiste uno strumento automatico per verificare la correttezza di un programma, ne consegue, che l'attività di

programmazione è, e rimane, di tipo creativo-artigianale e non può essere industrializzata. La produttività media dei programmatori non è incrementabile con strumenti automatici,[4] in questo senso, non si può considerare il loro prodotto come veramente industriale. In più il software invecchia molto rapidamente, quindi, produrlo è una costante corsa contro il tempo.
Conseguenza della "natura non industriale dell'industria del software" è che, un'azienda che implementa direttamente il software, dovrà investire in proporzione ai fattori sopra esposti ma dovrà anche avere, per il suo prodotto, un bacino di utenza sufficientemente ampio da assorbirne i costi. E non è una certezza che ciò avvenga.
Infine, come già accennato, il software può essere riprodotto e/o trasferito ad un costo irrisorio, che facilita le fasi successive dei software industriali ma anche la pratica, particolarmente diffusa in tutto il mondo, della duplicazione abusiva.
Considerando queste premesse non stupisce che le aziende produttrici, che seguono la filosofia 'Closed Source', abbiano cercato di difendere i propri prodotti, con tutti gli strumenti a disposizione. Innanzitutto informatici, come le tecnologie anticopia o l'occultamento del codice sorgente (quale segreto industriale), ora anche con dispositivi hardware. In secondo luogo, affidandosi agli strumenti giuridici: il copyright e se possibile con i brevetti.

3.3 Tipologia del software in base alle licenze.

Il software è classificato (quasi ovunque) come opera dell'ingegno, pertanto, l'autore è titolare del 'Diritto d'autore' o 'Copyright', che prevede una serie di diritti sulla sua opera, in parte cedibili ed in parte no.
Il Diritto d'autore distingue tra un diritto di proprietà immateriale (corpus mysticum), che spetta esclusivamente all'autore e quello del possesso materiale del bene (corpus mechanicum), ossia il diritto di chi possiede 'materialmente' e a qualsiasi titolo - acquisto, noleggio, prestito - quell'opera.
L'opera d'ingegno è legata indissolubilmente a colui che l'ha creata e tale vincolo persiste indipendentemente dalle vicende dell'opera o delle copie possedute da terzi.
In capo all'autore sono riconosciuti una serie di diritti, morali e patrimoniali, che possiede per il solo fatto di essere l'autore. L'acquisizione dei diritti è data dal fatto in sé della creazione dell'opera, senza formalità alcuna, come la pubblicazione, il deposito o la registrazione dell'opera.
Il diritto morale è inalienabile e riconosce all'autore la facoltà di rivendicare la paternità dell'opera e la sua integrità, cioè, di opporsi a deformazioni, danni o modifiche. L'autore è anche l'unico che può decidere se pubblicare o meno la sua opera (diritto di inedito).
Il diritto patrimoniale, invece, può essere ceduto; si concretizza con lo sfruttamento economico dell'opera in qualsiasi forma e con qualsiasi mezzo consentiti dalla legge e col godimento dei benefici connessi, ma anche, con la rinuncia agli stessi.

Il primo livello tra gli strumenti giuridici a tutela dei diritti sopra esposti, è il 'patto' che intercorre tra l'autore e l'utente.
Nel caso del Software, il 'patto' con l'utente è il 'Contratto di licenza d'uso'. L'utente, infatti, diventa proprietario solo del supporto e nemmeno di quello se entra in possesso del software tramite download[5], ha poi, facoltà di utilizzarlo, solo lui, secondo le modalità previste dalla licenza d'uso. Questa, infatti, contiene le volontà dell'autore per quanto riguarda la possibilità o meno per l'utente di eseguire il software per i propri scopi (escludendo eventualmente alcuni tipi di utilizzo oppure particolari soggetti) e per quanto tempo, di cederlo e a quale titolo (vendita, noleggio, prestito), di copiarlo (per cederlo o per conservare una copia di backup[6]), di studiarlo, ricostruirlo, adattarlo, correggerlo, modificarlo o tradurlo in altra lingua e ridistribuirne le versioni modificate, di affiancarlo o includerlo in altri software, ecc. Queste volontà sono tutelate dalle leggi sul Copyright o 'Diritto d'autore'.
In base a ciò che le licenze vietano o consentono, il software può essere classificato ed inserito nelle categorie Closed, Open, Free e Public domain Software o sottogruppi minori.

3.3.1 Public Domain Software.

È il software che circola senza la copertura di alcun tipo di licenza.
Agli albori dell'informatica erano molti i software a circolare in questo modo.

Con l'avvento del mercato del software si è presentato il problema della cooptazione di questi programmi di dominio pubblico da parte di singoli o aziende. Questi venivano presi, eventualmente modificati, e ridistribuiti coperti da copyright, spesso senza il codice sorgente e senza citare l'autore originario.
Gli autori di Free Software hanno adottato l'uso di licenze, proprio, per i problemi che il 'Pubblico Dominio' creava: cooptazione e chiusura dei sorgenti, possibile occultamento della paternità dell'opera, biforcazione dei progetti (come avvenuto con UNIX, anche tra Free e non-Free).

3.3.2 Free Software.[7]

Come definito nelle pagine di documentazione della Free Software Foundation,[8] si può parlare di 'Free Software' solo se rispetta precise caratteristiche.
Il 'Free Software', dove Free va inteso come libertà e non come gratuità,[9] riconosce all'utente quattro tipi di libertà[10]:
- di eseguire il programma per qualsiasi scopo;
- di studiare come funziona il programma e adattarlo alle proprie necessità;
- di copiare e distribuire le copie per aiutare il prossimo;
- di migliorare il programma e distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio;

solo se sono garantite tutte queste libertà si tratta di Free Software. Per usufruirne pienamente, vanno rispettate alcune condizioni:
avere a disposizione il codice sorgente, che deve essere incluso o reso disponibile, ma si è liberi di distribuire le copie del software, modificate o meno, nella forma che si ritiene più opportuna.
L'uso del programma non deve subire restrizioni di sorta nei confronti di persone o organizzazioni, su qualsiasi sistema informatico e per qualsiasi attività.
Le modifiche devono potersi fare senza formalità alcuna e lo stesso per la pubblicazione della propria versione.
Le libertà concesse devono essere irrevocabili, salvo abusi da parte dell'utente, in caso contrario il software non è Free.
Per l'uso, qualsiasi esso sia, non devono essere richiesti comunicazioni o permessi, inoltre, non va pagata la licenza d'uso. Questo non significa che non si possa chiedere un compenso per il supporto fisico e per le varie forme possibili di servizi o consulenza. Anzi, è proprio su questo che si sostiene il 'mercato' del Software Libero.
Le versioni modificate in ogni caso devono indicare l'avvenuta modifica, per proteggere la reputazione degli autori originari in caso di malfunzionamento.
Ulteriori regole sulle modalità di distribuzione sono ammissibili, purché non entrino in conflitto con le libertà principali. Ad esempio il 'Copyleft' o 'Permesso d'autore'[11], ovvero, la regola che, in caso di redistribuzione del programma originale o modificato, vieta di aggiungere restrizioni per negare ad altre persone le libertà principali ed obbliga al rilascio sotto la medesima licenza.
La proprietà di trasmettersi da un programma all'altro è la ragione per cui la licenza Copyleft GNU/GPL (la più diffusa ed importante del progetto GNU) è stata definita ad effetto autopoietico, virale.
Il Free Software è perfettamente compatibile con un utilizzo commerciale e non è insolito che venga usato, sviluppato e distribuito anche in ambito commerciale, al contrario, lo viluppo in quest'ambito è diventato una voce importante delle comunità del Software Libero.
È ammesso, inoltre, che vengano poste regole sulle modalità di fare un pacchetto (unire più software in un'unica distribuzione) purché non ostacolino la diffusione di versioni modificate.
Esiste Free Software privo di Copyleft ma nel progetto GNU della Free Software Foundation è fortemente sconsigliato: "Crediamo che ci siano importanti ragioni per cui sia meglio usare il permesso d'autore, ma se un programma è software libero senza permesso d'autore, possiamo comunque utilizzarlo"[12].

3.3.3 Open Source Software.

Il software Open Source, per essere definito tale, deve rispettare una serie di criteri[13] stabiliti nella Open Source Definition - una carta dei diritti dell'utente di software - della Open Source Initiative[14].
La redistribuzione deve essere libera [free], pertanto, le licenze devono consentire di vendere o donare i programmi anche aggregati ad altri software con licenze meno permissive o chiuse. Per le licenze d'uso non potranno essere richiesti diritti d'autore o altri pagamenti.
Il Codice Sorgente deve essere allegato, qualora non lo fosse deve essere comunque disponibile senza restrizioni e deve essere ben pubblicizzata la possibilità di ottenerlo gratuitamente.

Deve essere consentita la modifica e la realizzazione di prodotti derivati oltre alla loro redistribuzione alle stesse condizioni del software originale.
Il Codice Sorgente dell'Autore originario deve rimanere integro. Questo significa che la licenza potrà impedire la redistribuzione del codice sorgente modificato solo se quella stessa licenza consente la distribuzione di "patch files" (modifiche, aggiunte o correzioni contenute in appositi file non fusi in modo inscindibile con il codice originario), che modificano il programma al momento dell'installazione. Essa dovrà consentire esplicitamente la distribuzione di programmi costruiti a partire da codice sorgente modificato. Così, le modifiche 'non ufficiali' sono disponibili ma, anche, distinte dal codice originario. L'autore può pretendere che le versioni derivate siano distribuite con un altro nome.
La licenza non deve contenere restrizioni per persone, gruppi o campi d'applicazione.
I diritti inclusi in queste licenze devono trasmettersi automaticamente a coloro cui è ceduto il programma senza bisogno di licenze aggiuntive.
Se il programma è aggregato ad altri sotto un'unica licenza, in caso di estrazione e diffusione del singolo programma, questo mantiene la licenza originaria dell'aggregato.
Viceversa la licenza deve consentire che il programma venga aggregato (non 'confuso') ad altri senza porre vincoli sull'altro software, ad esempio, imponendo la stessa licenza.
Le differenze con le licenze Free Software non sono molte e infatti sono pochissime le licenze non comuni alle due definizioni, quelle Open Source, permettono una maggiore 'promiscuità' con il software proprietario e ne agevolano la distribuzione aggregata.

Il Software Libero in genere, è ceduto senza alcuna garanzia. La ragione dell'assenza di garanzia è dovuta al fatto che l'autore originario rischia di essere chiamato in cause legali anche per modifiche non apportate da lui. Garantire questo tipo di software diventerebbe di ostacolo alla sua diffusione ed esporrebbe l'autore ad insopportabili rischi finanziari.
È concessa garanzia, ma limitata - come avviene per il software proprietario - solo in caso di diffusione commerciale. In questo caso è il 'Commerciante' che si assume la responsabilità. La limitazione scelta nella maggior parte dei casi, fa si che si debba solo risarcire il prezzo del pacchetto software, oppure, obbliga alla sostituzione del supporto danneggiato. Mai in nessun caso per il danno causato per la perdita di dati o altri danni causati da malfunzionamento[15].

3.3.4 Software proprietario.

Si tratta di tutto il software per cui l'uso, la distribuzione e la modifica, sono vietate, oppure richiedono un esplicito permesso[16]. Anche il software proprietario è distribuito sia gratis sia a pagamento, in quest'ultimo caso si definisce con l'aggettivo commerciale. Le licenze sono diverse da un produttore all'altro e spesso le aziende differenziano le licenze per ogni software che producono e distribuiscono.

Il software proprietario si qualifica in vari modi, a seconda delle clausole della licenza, in base alle modalità di distribuzione e al fatto che venga o meno, e in che momento, richiesta una qualche forma di pagamento. Le licenze devono essere accettate (dopo averle lette! - ma è una pratica poco diffusa) durante l'installazione.
Le forme più comuni sono:

- Commerciale proprietario: prevede l'acquisto della licenza per poter entrare in possesso del supporto e/o per poter cominciare l'utilizzo. Alcune volte, il solo fatto di aprire la confezione esterna, in plastica trasparente del packaging, equivale ad accettazione della licenza - leggibile all'esterno della scatola, senza attendere l'accettazione esplicita durante l'installazione del prodotto[17].

- AD-Ware: contenente banner pubblicitari. Alcune case produttrici di software hanno deciso di rendere gratuito all'utenza il loro software. La licenza è offerta da coloro che usufruiscono del servizio di pubblicità inserito nel programma. Spesso è possibile liberarsi dalla pubblicità pagando la licenza per l'uso libero dai banner.

- Demo: hanno alcune opzioni non attive per cui il programma risulta incompleto ma utilizzabile (spesso usata per i video giochi), oppure, sono integri, ma, cessano di funzionare dopo un periodo di tempo concesso per provarlo (questo secondo tipo è detto anche Shareware).

- Shareware: possono essere ottenuti gratuitamente e provati per un certo periodo di tempo. Se il software è ritenuto soddisfacente per le proprie esigenze, allo scadere del periodo di prova, occorre pagare una certa somma all'autore altrimenti il programma cessa di funzionare.

- Freeware: ceduti gratuitamente, ma con le limitazioni all'uso previste dai software proprietari. Sono spesso usati dai produttori di software proprietario commerciale per inserirsi o guadagnare posizioni in una nicchia di mercato o per far affermare uno standard proprietario. In alcuni casi viene fatta una distinzione tra clienti domestici e 'Business' per cui, il medesimo programma, gratuito o a pagamento a seconda del tipo di utente che ne fa uso.

La distribuzione gratuita del software è, pertanto, una strategia attuata sia in ambito proprietario che Libero, ciò che le distingue è la finalità sottostante. In ambito proprietario si mira alla massima diffusione per conquistare fette di mercato legando l'utenza ai propri prodotti con i propri standard, che generalmente sono chiusi, (il che comporta, ad esempio, l'impossibilità di visualizzare i documenti o di visualizzarli esattamente come sono stati creati, se si utilizzano programmi di altri produttori) e per 'abitudine' (migrare da un software ad un altro, anche se svolge lo stesso compito, è costoso, quantomeno, per il tempo da dedicare all'apprendimento) e anche, per spingerli per 'gratitudine' ad acquistare altro software o la versione 'più evoluta' di quel produttore.
In ambito Libero si ha cessione gratuita perché la modalità di allocazione delle risorse è la reciprocità: la comunità dona il software all'utente e l'utente fa aumentare il valore intrinseco del software. Se il donatario è un utente attivo, arricchisce il patrimonio di conoscenza, il capitale sociale che sta dietro il progetto di sviluppo di quel software e arricchisce, nel loro complesso, le comunità del Software Libero.

3.4 Le principali Organizzazioni a sostegno del Software Libero.

Prima di passare al nuovo capitolo, sul terzo garante, che illustra le problematiche inerenti alla tutela giuridica del software, occorre accennare alle due più importanti organizzazioni senza scopo di lucro che sostengono e promuovono il Software Libero. Esse hanno dato vita a due distinti movimenti, con un'ampia base storica e filosofica in comune. Hanno confini 'fuzzy'[18], a volte indistinguibili, ciascuno con finalità, leader carismatici ed autocoscienza propri.
La Free Software Foundation [FSF] e la Open Source Initiative [OSI], sono sorte per la promozione di questo tipo di software, del suo modello di sviluppo, della filosofia e per incentivare l'utilizzo e la difesa del copyright specifico studiato per la tutela dell'ideale sottostante attraverso i propri 'prodotti'.

3.4.1 La Free Software Foundation e il "Copyleft".

All'inizio degli anni '80, il mercato si era imposto con le sue regole anche sul prodotto software, che veniva, così, diffuso a sorgenti chiusi per proteggere il vantaggio competitivo guadagnato sulla concorrenza.
I vincoli posti dai produttori impedivano la soluzione di problemi, anche banali, che in luoghi gremiti di persone competenti per risolverli, come il laboratorio del MIT, erano barriere profondamente frustranti, che ostacolavano lo svolgimento fluido delle attività quotidiane oltre a quelle di ricerca vera e propria. Ricorda Stallman: "Avevo già sperimentato cosa significasse un accordo di non diffusione per chi lo firmava, quando qualcuno rifiutò a me e al laboratorio AI del MIT il codice sorgente del programma di controllo della nostra stampante; l'assenza di alcune funzionalità nel programma rendeva oltremodo frustrante l'uso della stampante"[19].
La segretezza, ormai accettata dai più come normale esigenza di mercato, stava rapidamente contagiando gli ambienti della ricerca e delle università, e minava nel profondo l'uso mantenuto fino ad allora di cooperare, condividere e scambiare le conoscenze, bloccando di fatto l'innovazione.
Nel 1983, gran parte della comunità di hacker si era dispersa e Richard Stallman, decise di lasciare il suo incarico di sistemista al laboratorio del MIT "Una volta che il mio gruppo si fu sciolto, continuare come prima fu impossible. Mi trovai di fronte ad una difficile scelta morale. La scelta facile sarebbe stata quella di unirsi al mondo del software proprietario, firmando accordi di non-diffusione e promettendo di non aiutare i miei compagni hacker. [...] In questo modo avrei potuto guadagnare, e forse mi sarei divertito a programmare. Ma sapevo che al termine della mia carriera mi sarei voltato a guardare indietro, avrei visto anni spesi a costruire muri per dividere le persone, e avrei compreso di aver contribuito a rendere il mondo peggiore"[20].

Hacker puro, inflessibile, coerente con l'etica hacker, era profondamente convinto della dannosità pratica di non diffondere il codice sorgente, del fatto che il software non dovesse avere un 'padrone' e, comunque, che nessuno mai avrebbe dovuto pagare per poterlo usare.
Con la ferma volontà di ricreare una comunità libera per tutti, come quella del MIT delle origini, tra la fine del 1983 e l'inizio dell'84, Richard Stallman, mise in attività la Free Software Foundation (FSF), organizzazione non profit basata su lavoro e contribuzioni volontarie, ed il Progetto GNU[21] ad essa strettamente legato.
L'obiettivo del progetto GNU era di sviluppare un sistema operativo completo, compatibile con Unix, composto totalmente da software libero, il 'Sistema GNU', rigenerando, a tal fine, il progetto etico ed il modo di lavorare degli hacker. Un'altra delle ragioni era, ed è, quella di mantenere un saldo legame tra produttori ed utenti del software proprio perché, come già detto in precedenza, sono parte integrante del suo valore, che è un valore d'uso.
La FSF è lo sponsor primario del Progetto GNU, "la missione della FSF [è quella di] conservare, proteggere e promuovere la libertà di utilizzare, studiare, copiare, modificare e ridistribuire software per computer, e [di] difendere i diritti degli utenti del software libero"[22]. Essa si finanzia attraverso le donazioni, in denaro o attrezzature, da parte di privati[23]. aziende ed altre fondazioni. Ulteriore fonte di finanziamento proviene dalla vendita del proprio software, anche in versioni personalizzate per l'utente, e di altro materiale, dai manuali ai gadget. In questo modo è riuscita a ingaggiare un gruppo di programmatori professionisti per lo sviluppo di Free Software a tempo pieno e per l'assistenza ai propri 'clienti'.
La FSF è diventata un punto di riferimento per tutti i programmatori volontari che contribuiscono all'evoluzione del progetto GNU. Nella Free Software Directory, sono ospitati i collegamenti a 1743 progetti con licenza GNU/GPL o compatibili con il Free Software[24]. Il software ospitato gode della reputazione della FSF che richiede un livello di qualità del prodotto basato su standard molto elevati.
La versione del Sistema GNU al momento più diffusa è nota come "Linux". Al momento in cui fu scritto Linux, il Sistema GNU era quasi finito. Linux è solo il kernel, parte piccolissima ma fondamentale del sistema, il cui nome corretto è GNU/Linux[25]. Va rilevato, però, che proprio per merito di Linus Torvalds e del suo progetto "Linux" tutto il Software Libero, incluso il progetto GNU, ha preso nuovo vigore e sta uscendo dalla marginalità ed in alcuni particolari settori sta raggiungendo una posizione centrale[26].
Nel progetto GNU, La Free Software Foundation richiede che tutto il software, se possibile, venga diffuso sotto la licenza GNU/GPL.

Ma perché la licenza GNU/GPL è così importante?
Il progetto GNU è nato allo scopo di offrire la più ampia libertà possibile agli utenti e nel modo più capillare possibile. La soluzione già 'pronta' a questo scopo era il Pubblico dominio, ovvero, senza copyright.
Diffondere il software senza gravarlo di alcuna regola ha il vantaggio di garantire la massima propagazione possibile, ma, anche l'inconveniente, non trascurabile, di consentire la libertà di farne una versione modificata proprietaria a 'codice chiuso', privando, tra l'altro, l'autore e gli utenti del godimento della conoscenza delle migliorie e provocando inutili biforcazioni. In questo modo lo scopo del progetto GNU di offrire la massima libertà e la massima diffusione, e di garantire la possibilità di cooperazione, sarebbe stato vanificato.
La soluzione fu brillantemente trovata nel 'Copyleft' o 'Permesso d'autore'. Esso usa, in modo creativo e profondamente hacker, le leggi sul diritto d'autore, per consentire le libertà sostenute dal progetto GNU dandogli anche un solido fondamento giuridico. "Nel 1984 o 1985, Don Hopkins, persona molto creativa, mi mandò una lettera. Sulla busta aveva scritto diverse frasi argute, fra cui questa: "Permesso d'autore [Copyleft] -tutti i diritti rovesciati". Utilizzai l'espressione "permesso d'autore" per battezzare il concetto"[27].
Il Copyleft, scardina e ribalta l'uso ed il significato corrente del Copyright. Ciò che viene usualmente vietato nel copyright, qui, è permesso e difeso con i medesimi strumenti; è un impiego alternativo delle leggi sul diritto d'autore, con il risultato di ottenere un'adeguata protezione giuridica nonostante l'obiettivo inverso rispetto a quello del software proprietario. Il Copyleft consente al software GNU di non essere 'snaturato', di mantenere, cioè, la propria natura di bene pubblico puro.

Per ottenere il suo scopo, il Permesso d'autore, dopo le clausole espresse di consenso all'uso libero, cioè, di eseguire, copiare, modificare e redistribuire senza restrizioni appone un'altra clausola, che rende il copyleft così originale ed efficace, che vieta di aggiungere restrizioni ed obbliga ad ogni passaggio di trasmettere esattamente le stesse libertà che si erano ricevute. Le libertà date dal Permesso d'autore, diventano, così, dei diritti inalienabili.

Anche le versioni modificate, chiunque sia a farle, privato o azienda, devono essere libere: "Ciò assicura che ogni lavoro basato sul nostro sia reso disponibile per la nostra comunità, se pubblicato. Quando dei programmatori professionisti lavorano su software GNU come volontari, è il permesso d'autore che impedisce ai loro datori di lavoro di dire: 'non puoi distribuire quei cambiamenti, perché abbiamo intenzione di usarli per creare la nostra versione proprietaria del programma'"[28].
Il copyleft, però, consente solo l'aggregazione su un unico supporto o altro mezzo di distribuzione con programmi licenziati diversamente, questi programmi possono mantenere la loro licenza. Ma se vengono fusi, integrati o combinati con programmi protetti dal Permesso d'autore, i derivati, a loro volta, devono essere liberi e protetti da Permesso d'autore.
La più importante concretizzazione del Permesso d'autore è la GNU General Public License (Licenza Pubblica Generica GNU), abbreviata in GNU/GPL[29], usata per la maggior parte del Software GNU[30].
Il contenuto 'dispositivo' della licenza è quello descritto per il Free Software.
Non si tratta, però, di uno strumento contrattuale ordinario o di una normale licenza d'uso per software, infatti, il preambolo, e gran parte degli articoli della licenza, sono un vero e proprio 'manifesto politico', inteso a spiegare le motivazioni teoriche filosofiche e pratiche che le stanno dietro.
Come sottolineano Berra e Meo :" il modello del copyleft ha permesso di dare un fondamento giuridico a un mercato costruito sulla non mera appropriazione privata della proprietà intellettuale. Inoltre, la soluzione del copyleft fornisce uno stimolo a contribuire alla crescita del software libero, in quanto costituisce una garanzia di fiducia sulla stabilità di un patto di libera circolazione del software, e anche un modo per consentire un meccanismo di creazione di risorse finanziarie"[31].
Sta proprio in questo insieme di caratteristiche l'importanza cruciale della GNU/GPL che la rendono un vero e proprio motore del cambiamento istituzionale in grado di mostrare un modo nuovo di creare e distribuire il software che, come con il Copyright, scardina e rivolta le vecchie concezioni.
Il software 'Gipiellato', in quanto bene pubblico puro - che non solo non si consuma ma aumenta di valore con l'uso -, è lì da raccogliere dal 'Calderone magico'[32] e da usare; l'utilizzo non è mai appropriazione esclusiva come invece può accadere con il software di Pubblico Dominio.
La GNU/GPL suggerisce, per chi vuole accogliere il suggerimento, che esiste un modo nuovo di 'fare business' nel comparto del software e che è già lì 'a portata di mano'. Il Software Libero, infatti, è acquisibile liberamente e gratuitamente, ciononostante viene commercializzato come quello Proprietario. La differenza sostanziale rispetto al modello Proprietario sta nell'oggetto della compravendita: quello Proprietario ha come oggetto la licenza d'uso quello Libero, che la licenza d'uso l'ha già trasferita agli utenti, ha come oggetto il risparmio di tempo e di lavoro per acquisire il software.
Il modello proposto da Stallman e la forza del Copyleft hanno dato alla Free Software Foundation la forma di un vero e proprio movimento sociale che è riuscito anche a travalicare i confini dell'informatica e a 'contagiare', come già in precedenza accennato, altri ambiti del sapere.
Purtroppo l'ideologia radicale e intransigente della FSF, l'enfasi sulla parola Free, con il suo doppio significato di libero (quello corretto), ma anche di gratuito (elemento secondario e conseguenza del fatto di essere libero), e la proprietà virale della GNU/GPL sono malviste e/o male interpretate da coloro che agiscono facendo attenzione alle 'leggi del mercato'. Per questo per dirigenti o utenti commerciali e per le case produttrici di software proprietario, è abbastanza problematico prenderlo in considerazione.

3.4.2 La Open Source Initiative e le licenze OSI Certified.

Nel '97, un gruppo di leader della comunità Free Software, decise che occorreva trovare un modo per promuovere le idee del software libero tra coloro che rifiutavano il concetto.
L'intenzione era di ampliare il pubblico di GNU/Linux e del Free Software, ancora circoscritto a pochi addetti ai lavori, ad alcuni ambienti accademici e ad un certo numero di appassionati. Occorreva trovare un modo, di far uscire il Software Libero dalla marginalità e aumentarne il valore d'uso.
L'esperienza fatta con Linux dimostrava la validità del modello, e l'esperimento di Raymond descritto nel saggio 'La cattedrale e il Bazaar' [Cat B], forniva la prova che Linux non era semplicemente un caso fortunato. Il metodo di sviluppo era di sicuro interesse per i produttori di software ma l'ostacolo ideologico li teneva lontani.
Cat B circolava liberamente dal '97. Lo stesso anno, Netscape era alla ricerca di una strategia[33] per contrastare l'avanzata di Internet Explorer, il browser di Microsoft, ai danni del proprio Navigator[34].
Le argomentazioni di Cat B convinsero i dirigenti Netscape che il modello Linux poteva essere una buona idea, considerato anche il fatto che molti dei loro programmatori conoscevano e partecipavano a progetti di Software Libero e che già alcuni moduli del loro software erano Liberi. Così, il 23 gennaio '98 annunciarono pubblicamente l'intenzione di rilasciare al più presto i sorgenti del browser e di tutta la suite 'Communicator' come software libero: un milione e mezzo di righe di codice!

La notizia sorprese tutti, non solo gli operatori del settore, ma anche, la nascente comunità Open Source; era la prima volta nella storia, che una grande azienda software liberava il proprio codice proprietario. Ricorda Raymond: "Poco dopo che la notizia mi fu giunta, il giorno seguente, appresi che il CEO Jim Barksdale, parlando ai reporter dei mass media nazionali, aveva definito la mia opera come "l'ispirazione fondamentale" da cui era scaturita questa decisione.
Questo fu l'evento che, nella stampa commerciale specializzata in informatica, fu dai commentatori definito "il colpo udito in tutto il mondo". [...] Per la prima volta nella storia della cultura hacker una società inclusa fra le 500 di Fortune, quotata a Wall Street, aveva puntato il proprio futuro sulla convinzione che la nostra strada fosse quella giusta e, più specificamente, sull'analisi che io avevo formulato e secondo cui la "nostra via" era giusta. [...] Il fatto che CatB avesse modificato l'immagine della cultura hacker non sorprendeva; in fondo era l'obiettivo che avevo da tempo perseguito [...] a questo punto, la comunità hacker non poteva far altro che assistere la Netscape nella sua battaglia"[35].
Il prodotto della Netscape, era composto anche da software di proprietà di terzi[36] e nasceva il problema della licenza; non era possibile il semplice rilascio dei sorgenti e l'utilizzo immediato della GPL, perché incompatibile con le altre licenze.
Raymond insieme ad altri leader della comunità Open Source ed all'ufficio legale di Netscape si consultarono per affrontare il problema e trovarono una soluzione soddisfacente sia per gli sviluppatori sia per Netscape.
Il progetto di sviluppo del browser fu chiamato 'Mozilla'[37], lasciando l'uso di 'Navigator' in esclusiva alla Netscape. Furono inventate due diverse licenze: la 'Netscape Public License' (NPL), che garantisce il diritto di Netscape sul codice originario e derivato, e la 'Mozilla Public License' (MPL), che agisce all'interno della NPL, e garantisce come Free Software il codice nuovo. Organizzarono anche 'Mozilla.org' per ospitare e gestire il progetto in autonomia.
Dopo una frenetica operazione di ripulitura del codice dalle parti proprietarie non utilizzabili, con una grande festa nella notte tra il primo ed il due aprile 1998, fu ufficialmente liberato il codice[38].

L'idea di 'Open Source' venne forgiata nel febbraio '98 ad un incontro tra leader del Free Software. Si riunirono presso la VA Research a Mountain View coinvolgendo anche alcuni dirigenti dell'industria Linux [39] che muoveva allora i primi passi nel mercato. Il nuovo termine, avrebbe sostituito quello di Free Software evitando la connotazione anti-business di quest'ultimo. La decisione fu ufficializzata il mese seguente.
La notizia della scelta di Netscape arrivò proprio sul punto della svolta verso il nuovo modello Open Source e ne divenne portavoce e strumento di promozione.
Il gruppo di leader stabilì, anche, che occorreva una vera e propria campagna di marketing per convincere l'ambiente degli affari della validità del metodo di sviluppo e della possibilità di costruire con il Free Software una struttura in grado di reggere le leggi del mercato.
Per la campagna di marketing, rielaborarono i temi pragmatici esposti in Cat B e li usarono per ribaltare gli stereotipi negativi. Occorreva associare al Free Software ed all'Open Source Software delle immagini positive e attraenti per dirigenti ed investitori delle aziende di software proprietario, in modo da costruire una buona 'reputazione del prodotto' legata alla maggiore affidabilità, qualità e bassi costi. La correttezza del loro messaggio sarebbe stata 'facilmente' verificabile con l'accesso libero al codice sorgente. Elaborarono poi delle precise tattiche cui affidarsi[40]:
Dimenticare la strategia "bottom-up"; puntare sulla strategia "top-down".
La strategia di diffondere i concetti presso i tecnici si era dimostrata infruttuosa. Occorreva, invece, convincere direttamente i dirigenti perché imponessero le decisioni dall'alto.
Linux è il nostro caso più rappresentativo.
In quanto tale andava usato come esempio per attirare consensi.
Catturare le società "Fortune 500".
Ottenere l'attenzione delle società Fortune 500 significava catturarne i capitali e attirare le piccole e medie aziende per imitazione.
Cooptare i mass media di prestigio che si rivolgono alle società Fortune 500.
Per avere l'attenzione delle società Fortune 500 occorreva l'attenzione dei media che tradizionalmente si rivolgevano ai loro manager ed investitori: New York Times, Wall Street Journal, Economist, Forbes e Barron's Magazine.
Istruire gli hacker in tattiche di guerriglia marketing.
Tutti gli hacker, non solo pochi ambasciatori scelti.
Utilizzare il marchio di certificazione Open Source come garanzia di genuinità.
Registrarlo come marchio di certificazione collegandolo ad un significato preciso ed inequivocabile, per poterlo difendere da eventuali abusi e dai tentativi di stravolgerne il significato.

Occorrevano subito delle 'linee guida' per descrivere in modo dettagliato ed incontrovertibile ciò che rientrava o meno nel concetto di 'Open Source'.
Gran parte della Open Source Definition (il testo delle linee guida) si deve a Bruce Perens. Egli era alla guida del progetto 'Debian', una distribuzione GNU/Linux che si era impegnata ad utilizzare solo software conforme ai principi e alla filosofia del progetto GNU.
All'avvio del progetto Debian, era disponibile molto software gratuito, anche provvisto di sorgenti, ma sotto le più disparate licenze. Occorreva stabilire cosa era utilizzabile in quanto fedele alla filosofia del progetto e cosa no, e in base a quali criteri. Allo scopo furono scritti il 'Contratto Sociale Debian'[41] e la 'Guida Debian del Free Software'. Attraverso una conferenza via e-mail durata un mese, gli sviluppatori Debian ne discussero e perfezionarono i contenuti.
La Guida rendeva agevole decidere quale software, pur disponibile, fosse utilizzabile o no semplicemente confrontandola con la relativa licenza.
Raymond e gli altri, ritennero la Guida Debian molto adatta a descrivere il software Open Source. Da una versione modificata della Guida venne, quindi, ricavata la Open Source Definition[42].
La nuova 'Definizione' consente una più ampia promiscuità tra software libero e proprietario. La GPL entra a pieno titolo nella definizione ma è affiancata da molte altre licenze[43] più morbide e non 'virali', che però la Free Software Foundation considera inaccettabilmente restrittive per gli utenti.
Sempre nel '98 venne registrato un marchio di certificazione, 'OSI Certified', per poter garantire che il software licenziato sotto il termine Open Source fosse alla sua corretta interpretazione.
Inizialmente il marchio da registrare doveva essere direttamente 'Open Source' in modo da evitare il più possibile utilizzi e associazioni scorrette, ma il tentativo non andò a buon fine[44]. Per occuparsi della certificazione di marchio e della gestione della campagna di sensibilizzazione nell'ambiente degli affari fu creata una organizzazione: la 'Open Source Initiative'.

Al momento del lancio la campagna Open Source incontrò resistenze e forti critiche da parte della comunità Free Software con in testa un agguerrito Richard Stallman. Le critiche erano soprattutto rivolte contro l'abbandono del termine Free. Stallman e i suoi ritenevano che il tentativo di risolvere l'ambiguità semantica avrebbe avuto come conseguenza l'annullamento della percezione immediata del retroterra filosofico del movimento. "Mentre il software libero chiamato in qualunque altro modo offrirebbe le stesse libertà, fa una grande differenza quale nome utilizziamo: parole differenti hanno significati differenti [...] Il termine "open source" è stato rapidamente associato ad un approccio diverso, una filosofia diversa, valori diversi e perfino un criterio diverso in base al quale le licenze diventano accettabili [...] Il risultato è che la maggior parte delle persone fraintende quello che quei sostenitori sostengono. [...] La spiegazione di "software libero" è semplice: chi ha capito il concetto di "libertà di parola, non birra gratis" non sbaglierà più. [NdT: in inglese, "free speech, not free beer" mette sinteticamente in contrasto i due significati della parola "free"] Non c'è un modo più breve per spiegare il significato di "open source" e indicare chiaramente perché la definizione ovvia è quella sbagliata "[45].
Un'altra serie di critiche erano rivolte contro la promiscuità con il software proprietario, che avrebbe creato seri problemi, traendo in inganno utenti e programmatori. Il rischio stava nella possibilità di ritenere libero il software che in realtà non lo era, mettendo in pericolo gli sviluppi successivi di quel software e vanificando i contributi dei programmatori volontari. Infatti, se viene integrato del software proprietario insieme a quello libero, il destino del programma passa di fatto nelle mani del detentore della licenza proprietaria a meno di riscrivere ex novo tutta la parte non libera o di abbandonare il progetto.
Nonostante le critiche, dimostrate dai fatti[46], rimaneva il problema che l'intransigenza di Stallman ed il modello ideologico del movimento, poteva apparire ad uno sguardo superficiale o indotto con strategie FUD[47], come 'anarchico', 'comunista', o peggio, ispiratore della pirateria informatica; l'Open Source poteva essere il modo di attirare l'interesse delle case produttrici di software proprietario. È quello che poi è accaduto.

Dopo i feroci attriti iniziali, le relazioni tra i due movimenti si sono tranquillizzate. La situazione è simile a quella dei 'separati in casa'. Come precisa lo stesso Stallman: "Per il movimento Open Source, il fatto che il software debba essere Open Source o meno è un problema pratico, non un problema etico. Come si è espresso qualcuno, "l'Open Source è una metodologia di sviluppo; il Software Libero è un movimento di carattere sociale." Per il movimento Open Source, il software non libero è una soluzione non ottimale. Per il movimento del Software Libero, il software non libero è un problema sociale e il software libero è la soluzione [...] Siamo in disaccordo sui principi di base, ma siamo più o meno d'accordo sugli aspetti pratici. Perciò possiamo lavorare ed in effetti lavoriamo assieme su molti progetti specifici. Non vediamo il movimento Open Source come un nemico. Il nemico è il software proprietario. [...] Riconosciamo che hanno contribuito alla nostra comunità, ma noi abbiamo creato questa comunità e vogliamo che si sappia.

Vogliamo che quello che abbiamo realizzato sia associato con i nostri valori e la nostra filosofia, non con i loro. Vogliamo che ci sentano, non vogliamo sparire dietro ad un gruppo con punti di vista diversi"[48].

Rimangono le differenze su alcuni valori e sulla visione del mondo ma i loro rapporti vanno avanti evitando sterili spaccature.

3.5 Il modello Open Source come strategia di mercato.

Il Software Libero impone di trovare un modo diverso di 'fare business' nel comparto del software[49].
La soluzione escogitata col modello Open Source consente alle aziende di spostare il loro core business dalla produzione di codice sorgente, estremamente onerosa e carica di rischio per l'impresa, verso attività di fornitura servizi, di packaging e di edizione di questo software.
La produzione interna di codice non cessa, ma viene parzialmente esternalizzata e reindirizzata verso le comunità di sviluppo come contributo collaborativo volontario. Il contributo non è 'a fondo perduto' perché i propri tecnici, collaborando, aumentano le proprie competenze e conoscenze che poi riportano all'interno dell'azienda come capitale sociale.
L'elemento importante nel modello Open Source che molti riconoscono come premiante è, quindi, quello del supporto, di portare sul mercato sotto forma di business non il valore dell'acquisto del programma e del sorgente ma quello di 'servizi di supporto pacchettizzati' e non. Per sfruttare in pieno le tecnologie, per i clienti e per i mercati, è fondamentale avere qualcuno che supporti sotto l'aspetto dei servizi.
Coniugando in questo modo l'Open Source con il mercato, si evita, tra l'altro, il cortocircuito tra gli sviluppatori e gli utenti finali dai quali si ottengono importanti feedback utili all'orientamento del business.

GNU/Linux è il prodotto intorno al quale gravita, attualmente, la maggior parte dell'Open Source. Di fatto è usato soprattutto in aree legate all'infrastruttura della Rete: al Web serving[50], alle e-mail, o macchine a tema come firewall. Dopo il rilascio del nuovo kernel, all'inizio del 2001, ha avuto un'accelerazione inimmaginabile dovuta alle nuove potenzialità tecniche, che sono alla base della decisione se adottare oppure no questa nuova soluzione, e ha cominciato ad espandersi anche per altri usi come il comparto desktop.
GNU/Linux, e in generale il Software Libero, è vantaggioso da diversi punti di vista. Innanzitutto la flessibilità, perché ha la possibilità di essere una risorsa disponibile per aziende ed enti di qualsiasi livello, dal programmatore free alla multinazionale, e per prodotti di qualsiasi dimensione, dai dispositivi embedded ai mainframe.
In secondo luogo, si basa su principi pluralistici, garantendo la neutralità tecnologica, con formati e standard aperti. In modo che nel momento in cui si comunica tra entità con piattaforme diverse non vi siano ostacoli.
In più, è un opportunità oggettiva per tantissime aree e nel rispetto delle diverse velocità dei singoli mercati. In certi mercati come quelli emergenti risulta più facile l'affermazione di GNU/Linux perché si parte da zero direttamente con questa soluzione. Ma anche in realtà europee o italiane è una opportunità potenziale importante ad esempio nelle Pubbliche Amministrazioni[51], nella scuola o nel mondo del settore no profit dove è fondamentale il risparmio.
La situazione dei paesi emergenti è quella di disporre di poco denaro, tecnologie obsolete[52] o modeste, ma una notevole abbondanza di risorse umane e intellettuali.
Molti di questi paesi stanno adottano soluzioni libere per non dirottare i loro capitali verso altri paesi e per poter investire su energie locali, che fanno da volano per ulteriore sviluppo. In Sudamerica ed in Estremo Oriente l'uso di GNU/Linux è in rapidissima crescita.
Infine, c'è l'aspetto della durata della soluzione adottata. Ci sono i sorgenti e se c'è bisogno di funzionalità aggiuntive è possibile svilupparle internamente o farle sviluppare. In ambiente proprietario si è legati indissolubilmente al fornitore del software per cui si è costretti a cambiare il prodotto se non si viene più supportati, se il fornitore scompare o se la propria versione non possiede le funzionalità necessarie con l'ulteriore aggravio di dover convertire tutti i dati dal formato vecchio a quello nuovo altrimenti irrimediabilmente illeggibili (sempre che questo sia possibile).
In sintesi è una grossa opportunità di sviluppo che ritorna al mercato, alle aziende ai clienti e agli utenti, proprio là dove si forma bisogno.

3.5.1 Le distribuzioni.

Le distribuzioni[53] sono l'esempio più significativo di attività da parte di organizzazioni che si dedicano al software Libero. Sono molto numerose, di diversi tipi, dimensioni e prezzo, sia commerciali che non commerciali; utilizzano prevalentemente software proveniente dal Progetto GNU. La maggior parte si basa sul kernel[54] GNU/Linux. Non è l'unico ma è il più usato per ragioni 'tecniche' oggi.
La distribuzione si occupa di assemblare le varie parti del sistema (kernel e software di supporto), di sviluppare quello che manca o che vuole migliorare, di mettere insieme una collezione di software applicativo e di verificare che il tutto funzioni senza problemi. Generalmente differiscono tra loro per varie caratteristiche come i metodi di installazione, gli strumenti per l'amministrazione del sistema ed altro. Ognuna è orientata ad una o più nicchie per cui si hanno distribuzioni dedicate alle aziende (Red Hat, che è anche la più diffusa tra le commerciali, oppure la tedesca S.u.S.E.), al Desktop (Mandrake o Debian con un progetto avviato ma non ancora pienamente sviluppato), o ad una singola popolazione come la Red Flag[55], distribuzione cinese o ancora a settori particolari come i progetti avviati da Debian dedicati ai bambini tra i 7 e i 12 anni (Debian Jr.), alla pratica medica e di ricerca (Debian Med), al settore educativo (Debian-Edu), solo per fare qualche esempio.
Le distribuzioni partecipano attivamente ai Progetti Open Source collegati al loro prodotto o sono esse stesse dei progetti.
Le organizzazioni commerciali, le aziende, che seguono questo modello devono offrire alla propria clientela del valore aggiunto. Non si possono appoggiare alla semplice cessione della licenza d'uso come nel modello proprietario. Molte riescono ad ottenere buoni utili provenienti da più fonti: la vendita di copie, la vendita di manualistica e materiale di supporto, lo sfruttamento del marchio, la garanzia, la vendita di training, consulenze, sviluppo e supporto tecnico post-vendita e in piccola parte anche dai gadget. Le versioni 'in scatola' vendute nei negozi includono copie del sistema e degli applicativi, manualistica e 'servizi pacchettizzati' come l'assistenza (telefonica, on-line oppure on-site) o la garanzia, il supporto tecnico ed eventualmente software proprietario di cui rivendono le licenze. La vendita si compone di due categorie generali di prodotto: beni fisici e servizi. Le versioni 'downloadabili'[56] sono gratuite e Free ma non danno servizi extra.

Va fatto un discorso a parte per la distribuzione Debian GNU/Linux. Debian vive solo sui canali si comunicazione: è un sistema operativo completamente libero, sviluppato in modo cooperativo da un migliaio circa di volontari sparsi in tutto il mondo, che collaborano via Internet. "La dedizione al software libero di Debian, la sua natura non-profit e il suo modello aperto di sviluppo la rendono unica tra le altre distribuzioni Linux. Punto di forza del Progetto Debian sono la base di volontariato, la sua dedizione al Contratto Sociale Debian e alla sua volontà di fornire il miglior sistema operativo possibile"[57].
Attualmente non è il sistema più semplice ma molto versatile: Debian GNU/Linux, versione 3.0., supporta undici diverse architetture[58], funziona su macchine che vanno dai palmtop ai supercomputer, ed è tradotta in molte lingue. Viene distribuita con oltre 8710 pacchetti[59], programmi tra cui scegliere in base alle proprie esigenze, già compilati e impacchettati per agevolare l'installazione.
Debian è supportata da donazioni monetarie, di materiale hardware o di banda passante su Internet. Non segue i tempi di rilascio delle nuove versioni dettati dal mercato, ma i tempi 'naturali' dello sviluppo cooperativo. Nonostante ciò i ritardi si possono considerare ragionevoli (pochi mesi rispetto alle altre distribuzioni), dato che il prodotto finale è considerato eccellente.

3.5.2 Le imprese che forniscono servizi e supporto.

Intorno al modello Open Source sono sorte numerose imprese individuali o aziende che dedicano la propria attività, ricavando profitto, alla fornitura di servizi (supporto, ricerca, sviluppo, localizzazione, personalizzazione, formazione, ecc.) come Linuxcare[60] o Prosa[61], oppure 'accessori', ad es. i manuali sul software Open Source della O'Reilly & Associates[62] o delle italiane Hops[63] e Apogeo[64] pubblicati con licenze che consentono la riproduzione dei testi come la GNU/GPL o la GNU/FDL[65].

I consulenti autonomi.
La disponibilità dei sorgenti e la modularità che rende estremamente flessibile il Software Libero, consentono di adattare il comportamento del computer alle diverse necessità. È una possibilità sconosciuta all'utente di Software proprietario che viceversa deve adattare i propri bisogni alle funzionalità del prodotto che ha acquistato; può personalizzarlo solo per la parte concessagli dal produttore, oppure, si ritrova un carico di funzioni che non utilizzerà mai e che appesantiscono inutilmente il suo sistema.
L'utente con particolari esigenze può rivolgersi a consulenti autonomi in grado di fornire assistenza tecnica per personalizzare ed estendere il software con le caratteristiche di cui ha bisogno.

Come racconta Alessandro Rubini[66]: "Per esempio, ho scritto, insieme con altri, un programma per un laboratorio di Fisiologia [...]. Nei due anni di utilizzo, i medici hanno trovato talmente tanti modi per estendere il programma che ora questo software è considerato migliore delle soluzioni proprietarie. Se consideriamo il totale di quanto hanno pagato in questi anni, il programma risulta alla fine più costoso di alcune alternative proprietarie. Questo non è determinante per i miei clienti, poiché hanno ottenuto esattamente ciò che volevano [...]. Ovviamente il programma è software libero ed altri centri hanno dimostrato interesse ad averne una copia".
L'assistenza al software proprietario è spesso fornita soltanto da consulenti autorizzati, le cui quantità e qualità sono gestite in modo centralizzato; il Software Libero mette, invece, la conoscenza tecnica a disposizione di chiunque voglia imparare, chiunque può costruire la propria professionalità ovunque si trovi nel mondo (purché conosca un po' di inglese, abbia un computer ed un telefono), e ad un costo relativamente contenuto (quello della telefonata per collegarsi ad Internet) su hardware obsoleto o molto economico.
L'offerta di consulenti a supporto del Software Libero può, senza dubbio, adattarsi alla domanda e la competenza evolvere parallelamente al software.

Le società di assistenza tecnica.
Quando il bisogno del cliente non può essere coperto da una singola persona o da una piccola impresa, ci si può rivolgere a società specializzate nell'assistenza tecnica che si occupano di quelle situazioni in cui i sistemi informatici sono molto complessi o utilizzati in modo intensivo e vanno gestiti da un pool di tecnici, ad esempio, nelle grandi imprese o nelle banche, dove l'assistenza deve garantire reperibilità 24 ore su 24 e solo una società più strutturata può assicurare una tale copertura.
In situazioni di questo tipo, come già accennato, l'uso di software proprietario vincola l'utente al servizio offerto dal fornitore annullando ogni possibilità di scelta salvo disponga di adeguata forza contrattuale. Se il servizio non soddisfa o se il fornitore scompare dal mercato, non resta che la 'migrazione' verso altre soluzioni con i costi aggiuntivi, anche ingenti, che essa comporta. Il Software Libero, invece consente la scelta del fornitore del servizio che può essere lo stesso del software (tutte le distribuzioni ad esempio lo offrono, 'pacchettizzato' o personalizzato) oppure da società indipendenti, che 'giocano' in un sistema di mercato con pari opportunità.

Le società di formazione e certificazione.
Sono società indipendenti che formano i tecnici e/o attestano la loro preparazione; sono già presenti ma non in quantità sufficiente e spesso con attestati non riconosciuti universalmente. Le distribuzioni forniscono servizi di questo tipo ma ovviamente calibrati solo sui propri prodotti.
Per quanto riguarda in generale il Software Libero, stanno già nascendo società in grado di fornire attestati e garanzie di funzionamento, compatibilità, standardizzazione.

Lo sviluppo di queste nicchie di mercato a servizio del Software Libero, dipenderà dall'evoluzione complessiva del mercato del software, questa, a sua volta dipenderà dalle conseguenze delle scelte passate, presenti e future, riguardo la struttura dei diritti e delle regole politiche, economiche e contrattuali, che lo coinvolgono direttamente.

3.6 Il problema del Terzo garante.

Per concludere la panoramica sulle istituzioni formali occorre fare un salto indietro nel tempo.
La moderna legislazione sul Copyright nasce, e si sviluppa, tra il 15° ed il 16° secolo, poco dopo l'invenzione da parte di Gutenberg della stampa a caratteri mobili, come insieme di privilegi e prerogative concessi ad autori ed editori.
La ragione della creazione di questi nuovi diritti era quella di incentivare la produzione culturale a vantaggio di tutta la società. La concessione agli autori di un monopolio temporaneo di alcuni diritti serviva a garantire loro la possibilità di sfruttare economicamente il proprio lavoro. Per gli editori, la possibilità di stampa in esclusiva, consentiva di annullare il rischio dato dalla concorrenza e di rientrare più agevolmente degli investimenti fatti. Si reputò che abbassando il rischio connesso alla pubblicazione ci sarebbero state, a beneficio dei lettori, molte pubblicazioni in più[67].
Nel passato, la possibilità economica di riproduzione delle opere, era privilegio di pochi e la legislazione garantiva in modo ragionevolmente equo tutte le parti in causa.

Oggi le cose sono molto cambiate.
Il recente rapidissimo avvicendarsi di nuove tecnologie come registratori a cassette, fotocopiatrici, videoregistratori, ecc., l'avvento del digitale e dei computer domestici, l'espansione straordinaria delle reti e di Internet, ha cambiato lo scenario di riferimento e ha costretto a numerose quanto affrettate modifiche legislative del copyright, tuttora in corso.
Per molto tempo Internet, la novità che ha letteralmente rivoluzionato lo scenario, è rimasta zona franca per il diritto e la Rete è un meta-territorio dove ognuno è dovunque in ogni momento. Questo fatto non è ancora sufficientemente compreso da chi si occupa di legiferare in proposito, infatti, il software e gli altri prodotti digitali, continuano ad essere protetti su base territoriale attraverso il diritto d'autore e contemporaneamente dal segreto industriale - qualora distribuito senza codice sorgente. In ogni caso, la facilità di riproduzione si è accompagnata per molto tempo ad un'ampia tolleranza da parte degli apparati politici, economici ed educativi e questo ha contribuito a plasmare gli attuali comportamenti.
Gli utenti che riproducevano abusivamente le opere protette dal diritto d'autore non erano quasi sanzionati. Molte aziende tolleravano la duplicazione abusiva - che però condannavano pubblicamente - perché forma indiretta di promozione e diffusione dei propri prodotti e dei propri standard chiusi. La conseguenza è stata anche la non percezione come reato della duplicazione abusiva[68] del software e dei prodotti digitali in genere da parte dell'utente.

Va ora sottolineato il fatto che i differenti copyright, posti dai produttori a tutela del software, assecondano oppure sono in contrasto con la natura immateriale del prodotto digitale e le abitudini dell'utenza; permettere ciò che altri vietano - copia, modifica, redistribuzione - cambia totalmente la garanzia di applicazione dei contratti, l'attività di controllo da parte del terzo garante e la struttura dei costi totali che includono accertamento e censura delle violazioni. Il software Libero è avvantaggiato, da questo punto di vista, perché il controllo è rivolto quasi solo verso i produttori: ad esempio, per la redistribuzione di Software Libero con licenze proprietarie o per il plagio di parti di Software Libero incluso in prodotti proprietari e diffuso senza sorgenti (in questo caso è più difficile la prova per via del segreto industriale).
Gli abusi nei confronti dei prodotti proprietari sono invece posti in essere anche dai singoli utenti in uno stillicidio di piccole violazioni difficili da individuare e da reprimere, tant'è che si è preferito fare estemporanee campagne punitive anti-pirateria che hanno colpito pesantemente solo pochi malcapitati[69] senza minimamente intaccare il problema.

Dopo l'introduzione di ulteriori nuove tecnologie come masterizzatori per CD e software peer-to-peer[70] è cominciata una forte attività di lobbying da parte delle maggiori aziende editoriali, cinematografiche, musicali e del software perché venissero approvate normative più severe. In Italia, ad esempio, oltre ad obblighi e controlli aggiuntivi per distributori e rivenditori (onerosi e senz'altro odiosi, non solo per chi redistribuisce il Software Libero) hanno raggiunto un ulteriore risultato: quello della 'socializzazione' del costo dell'accertamento, dove le violazioni del contratto di licenza d'uso, sono state trasformate in illeciti penali perseguibili d'ufficio.
Nonostante ciò la normativa si è dimostrata inefficace a proteggere gli interessi della parte che l'ha promossa e lesiva per tutti gli altri, dando ragione a chi invece la critica e invita a pensare a nuovi modelli scollegati dai vecchi schemi ormai difficili da sostenere.
Una buona legislazione dovrebbe mediare tra tutti gli interessi in gioco in modo il più possibile equilibrato, ma per il software, si stanno accumulando squilibri e anomalie per via dell'asimmetria di potere contrattuale sia economico che politico che c'è tra gli attori.
I produttori di software proprietario in molti casi hanno approfittato di posizioni favorevoli nel mercato per imporre i propri standard e raggiungere possibilmente posizioni di monopolio ricorrendo anche a sistemi ai confini del lecito come l'acquisto dei concorrenti per cancellarli dal mercato o a mezzi illeciti come 'forzare' i produttori di computer alla vendita abbinata hardware-software, vietata, ad esempio, in Europa[71] senza che l'autorità a garanzia della concorrenza intervenisse tempestivamente e/o efficacemente a difesa di coloro che hanno presentato ricorso.
Oltre a quanto già detto che, come già rilevato, può cambiare in modo più o meno significativo da un continente all'altro e da paese a paese, è da segnalare un'altra caratteristica (anomala) nello status legale del software: esso sfrutta gli stessi privilegi di cui si giovano le opere d'arte.
Nessuno garantisce che il prodotto assolverà ad alcuna funzione, compresa quella per cui lo si è acquistato. È un'immunità comprensibile per un libro, un quadro o per una canzone, ma non per un bene strumentale come il software.
Legalmente non si può chiamare in giudizio il produttore del software, nemmeno se si scoprono palesi errori di programmazione, o peggio, se si è subìto un danno (come la perdita di dati) causato dal suo provato malfunzionamento. Conseguenza diretta di questo stato di cose è che il produttore di software non

è tenuto a correggere errori segnalati o riconosciuti e che, se ne ha interesse - e spesso avviene -, può introdurre volontariamente degli errori che sistemerà successivamente, a pagamento, con un "aggiornamento".
Non tutti i produttori possono avvantaggiarsi di queste specificità giuridiche ma solo quelli con grande potere di contrattazione, per cui, il piccolo sarà comunque tenuto, ad accollarsi i costi relativi all'obbligo di risultato ed alle clausole di garanzia, per poter lavorare e per soddisfare le commesse importanti.
L'utente 'medio' ha potere contrattuale pari quasi allo zero. Ad esempio, quando le software house ritirano dal commercio i vecchi programmi (spesso lo fanno per spingere quelli nuovi), non hanno l'obbligo di depositare i sorgenti e/o le specifiche tecniche in archivi pubblici; nessuno più potrà supportare quell'utente, pertanto, tutti i dati, di proprietà dell'utente, scritti in quel formato diventeranno con il passare del tempo e con l'adozione di nuovi programmi e formati da parte degli altri utenti, irrimediabilmente illeggibili da questi e lui stesso avrà problemi di conversione se vorrà adottare prodotti di altri fornitori. Oppure, se un consumatore italiano vuole comprare in un grande magazzino un computer economico, ma lo vuole 'naked' [nudo], cioè privo di sistema operativo, deve armarsi di infinita pazienza ed arrivare molto ben preparato al momento dell'acquisto per controbattere le obiezioni del venditore e per ottenere il rimborso del sistema operativo che non desidera, spesso senza riuscirci, nonostante la legge europea antitrust vieti questi comportamenti[72].

I maggiori produttori di software (insieme agli editori) stanno anche invocando una maggiore omogeneizzazione delle legislazioni di tutela a livello mondiale, dato che, spesso, questa risulta contraddittoria o inapplicabile con alti costi di misurazione ed impossibilità di ricorrere all'esecuzione coattiva. In linea di principio è una richiesta più che legittima ed auspicabile, proprio per ridurre le incertezze e favorire il settore indipendentemente dal tipo di software prodotto, ma le proposte di armonizzazione appoggiate dalle Major rischiano di introdurre nuovi squilibri ed incertezze.
Negli Stati uniti è stato introdotto nel 1998 il Digital Millenium Copyright Act[73] che ha già prodotto i suoi effetti; in Europa è stata emanata la direttiva 2001/29/CE meglio conosciuta come European Union Copyright Directive (EUCD)[74] riguardante "l'armonizzazione di taluni aspetti del diritto d'autore e dei diritti connessi nella società dell'informazione", non ancora attiva ma che entro il 22 dicembre 2002 dovrebbe venire integrata nelle legislazioni degli stati membri.
Queste normative introducono forti ostacoli ad una vasta gamma di attività, finora svolte senza vincoli (quali la ricerca sulla crittografia, la libera diffusione di informazioni ed il libero sviluppo di software), perché potenzialmente in grado di agevolare violazioni al diritto d'autore, quindi virtualmente pericolose.
L'introduzione di tali vincoli non tiene conto delle conseguenze negative, che vanno ad intaccare attività non solo oggi lecite, ma anche, indispensabili per il buon andamento di attività utili all'evoluzione dello stato dell'arte e per l'intera società.

Queste normative compromettono la libertà di ricerca e di espressione, il diritto al 'fair use' dell'opera da parte degli utenti, la possibilità di sviluppare nuovo software (libero e proprietario), impediscono la nascita di un mercato del materiale digitale "di seconda mano" e lo sviluppo di attività di conservazione e/o diffusione di materiale digitale con rilevanza storica e documentaristica, infine, indeboliscono ancora di più la possibilità per gli utenti di difendere nei tribunali le proprie libertà.
Le nuove leggi, però, non cambiano il fatto che l'introduzione e l'applicazione avvengono a livello locale pertanto lo scopo dichiarato di armonizzare le legislazioni viene decisamente meno.
Negli Stati uniti, il DMCA è in vigore dal '98 e le conseguenze negative si sono già fatte sentire. In alcuni ambienti (ricerca scientifica, università, aziende), si comincia a farlo presente e a protestare per le conseguenze inattese del DMCA; alcuni paesi sono giunti a raccomandare ai propri programmatori di non mettere piede sul suolo americano per non incorrere in sanzioni per attività lecite svolte nel proprio paese, ma, che in America, sono ritenute sanzionabili[75].
È ciò che è accaduto al programmatore russo Dmitry Sklyarov[76] autore nel suo paese di un programma alternativo a quello di Adobe per la lettura dei suoi e-book (regolarmente acquistati), in grado di aggirare le protezioni e utilizzato tra l'altro per permetterne, alle persone cieche, la lettura attraverso un sintetizzatore vocale. Sklyarov, negli USA per un convegno, venne arrestato. L'accusa non riguardava violazioni del copyright ma del DMCA per aver creato un programma in grado di aggirare protezioni tecnologiche, quindi, potenzialmente in grado di farlo. Il giovane rimase in carcere diverse settimane e fu scarcerato a seguito del ritiro di parte delle accuse formulate da Adobe indotto dalle pesanti proteste su Internet e di fronte alla minaccia di boicottaggio da parte dei suoi stessi utenti.

Vi è un'ultima, ma non meno importante, questione di tipo normativo che riguarda la materia dei brevetti [77].

Lo strumento giuridico del brevetto nasce per proteggere soluzioni originali a problemi tecnici, il fine sociale del brevetto è quello di mettere a disposizione del pubblico un'innovazione, sia pure garantendo all'inventore un certo periodo di tempo durante il quale ne sarà il monopolista (20 anni in tutti i paesi dove è presente questa istituzione). Infatti i brevetti sono sottoposti a due clausole limitative: quella temporale e quella che obbliga alla pubblicazione completa in un archivio pubblico in modo da svelare l'idea sottostante e il modo in cui è stata realizzata. La pubblicazione ha lo scopo di consentire ad altri di non dover reinventare la stessa cosa agevolando così la ricerca di quel settore. Quindi, anche se limita il libero uso delle invenzioni non dovrebbe 'danneggiare' il progresso e la società, che può giovarsi dei miglioramenti delle tecnologie.
La pratica brevettuale ha come scopo primario, secondo i suoi sostenitori, di stimolare la ricerca garantendo gli investimenti, facendo così da incentivo economico ai ricercatori e alle aziende che investono nella ricerca. Molti ricercatori, però, contestano la pratica brevettuale, la ritengono dannosa in sé e di ostacolo all'innovazione e pensano che senza brevetti andrebbe anche più speditamente. La questione è, e da tempo rimane, complicata ed aperta soprattutto per il risvolto economico che comporta ogni modifica allo stato delle cose.

Tutte le legislazioni vietano di brevettare le idee a sé stanti. Il brevetto infatti riguarda l'applicazione industriale di una certa idea. Di rigore, il software in quanto tale, insieme alle idee, teorie scientifiche e metodi matematici, non è brevettabile, come sancisce esplicitamente la legge, e la sua protezione è affidata al diritto d'autore, ma, la legge viene aggirata brevettando il software in quanto strumentale al funzionamento e all'utilizzazione di dispositivi industriali, che possiedano caratteristiche di novità di non ovvietà e di utilità.
Se tutte le idee alla base dei programmi fossero brevettate nessuno potrebbe più sviluppare software a parte poche grandi aziende e lo sviluppo si fermerebbe[78]. Ma è proprio quello che sta accadendo negli Stai Uniti dove la pratica ha introdotto di fatto la brevettabilità del software aggirando il divieto[79]. In più, i produttori di software, hanno ottenuto un ulteriore (atipico) privilegio: quello di brevettare le loro invenzioni, depositando solo la descrizione in parole ed in diagrammi dei software usati, senza rendere pubblicamente visibile il codice sorgente[80]. In questo modo, se il brevetto viene registrato, nessuno può più dirigersi in quella direzione nello sviluppo o nella ricerca, senza pagare o essere portato in tribunale. Il rischio peggiore risiede nel fatto che un programmatore può scoprire di aver violato uno o più brevetti dopo aver rilasciato il prodotto. I costi legali per sostenere le proprie ragioni rischia di mandare in fallimento anche aziende di medie dimensioni. Oltretutto, evitare il rischio è praticamente impossibile perché le ricerche tra i brevetti, spesso assegnati con troppa leggerezza, sono poco affidabili e comunque troppo costose per i singoli e le PMI. Non è difficile constatare la distorsione della concorrenza e il fatto che ci si è allontanati di molto dallo spirito che fu alla base dell'introduzione dei brevetti (e del copyright).
In Europa il fenomeno è già presente ma in misura ridotta rispetto agli USA[81]; non mancano però proposte volte all'introduzione legale dei brevetti astratti motivate dal bisogno di "uniformare il mercato europeo a quello americano", per 'aiutare' il mercato europeo.
I brevetti astratti, registrati dalle grosse società (IBM, Apple, Microsoft, ecc.), sono usati come merce di scambio e per 'tenere sotto controllo' la concorrenza, infatti, i detentori dei brevetti non hanno obblighi particolari per cui possono concedere o no una licenza su un brevetto, alle condizioni che vogliono, anche facendo discriminazioni tra un soggetto e l'altro. Spesso il controllo dei brevetti è gestito attraverso la creazione di società specializzate, la cui sola attività è quella di sfruttare le licenze d'uso, senza svolgere attività produttive o di ricerca.
Negli USA le piccole imprese e i singoli sviluppatori non sono economicamente in grado di sostenere questo stato di cose che li imprigiona in un insieme confuso di norme o direttamente di processi giudiziari tale da aver frenato e quasi bloccato le capacità di crescita e di innovazione di una larga parte di questo settore.
In Europa il 90% delle piccole e medie imprese si è espressa contro la brevettabilità del software ritenendola una minaccia alla libera concorrenza e alle piccole e medie imprese finché l'Unione Europea manterrà il divieto, si gioverà di condizioni più favorevoli allo sviluppo e alla concorrenza tra gli attori, e continuerà a ridurre il gap tecnologico che ha con gli americani.

Il quadro delle maggiori istituzioni formali che interessano il mercato del software è questo. Si può comprendere quanto sia difficile nel mondo del digitale controllare efficacemente il rispetto delle regole economiche, dei diritti di proprietà e dei contratti. È una situazione che crea incertezza e difficoltà decisamente negativa, specialmente se si pensa al fatto che ci si riferisce ad una risorsa strategica per l'evoluzione dei mercati.

Le risposte politico-giuridiche poste in essere per far fronte al cambiamento sembrano non comprendere né la sua natura del né quella della risorsa intorno alla quale si sta strutturando la 'nuova economia'. L'evoluzione fino ad ora è stata così rapida anche per via dell'assenza di quelle regole formali che si vogliono ora introdurre.

Il movimento che sostiene il Software Libero è comunque molto attento e propositivo, l'opera di informazione che stanno svolgendo molto attivamente serve a contrastare l'asimmetria informativa attualmente presente sul mercato politico, sono state presentate molte proposte di legge a tutti i livelli e molte realtà stanno già cogliendo le potenzialità di questo 'nuovo' modello.

---

1 Gambaro, Ricciardi, 1997.
2 Previsto dalla Convenzione di Berna del 1886. Ma, seguito anche dai paesi non firmatari.
3 Berra, Meo, 2001, pagg. 25, 65, 126-7.
4 Ibid.
5 Prelievo di un file da un computer remoto.
6 Copia di sicurezza o di riserva.
7 Come nei precedenti capitoli, uso Free Software per distinguerlo dall'Open Source e Software Libero se mi riferisco ad entrambi.
8 Organizzazione senza scopo di lucro fondata da Richard Stallman nel 1984 per patrocinare il Free Software. per una descrizione approfondita rimando al prossimo paragrafo.
9 L'ambiguità del termine inglese Free provoca questo fraintendimento.
10 http://www.fsf.org/philosophy/free-sw.it.html
11 È un gioco di parole: copyright (diritto d'autore) è formato dalle parola 'copy' (copia) e 'right' (diritto, ma anche destra), opposto di 'left' (sinistra, ma anche lasciato), da qui 'Permesso d'autore'.
12 'Cos'è il Software Libero?' http://www.fsf.org/philosophy/free-sw.it.html
13 Versione 1.9; http://www.opensource.org/docs/osd-italian.php.
14 Organizzazione senza scopo di lucro che vede tra i fondatori Bruce Perens. Egli scrisse la prima bozza come linee guida cui dovevano attenersi gli sviluppatori della distribuzione Debian GNU/Linux, "The Debian Free Software Guidelines" e l'ha perfezionata usando i commenti degli sviluppatori stessi attraverso una conferenza via e-mail che durò tutto il mese giugno del 1997. Egli riformulò le linee-guida della Debian per creare la "Open Source Definition". Per ulteriori dettagli rimando al paragrafo 3.4.2.
15 Spesso sono clausole in contrasto con le legislazioni nazionali che servono a disincentivare l'avvio di azioni di risarcimento danni anche se legittime.
16 Il più delle volte a titolo oneroso.
17 Licenza 'a strappo'.
18 Confusi, sfumati. La 'Logica Fuzzy' (non quella aristotelica) consente di comprendere meglio la distinzione dei due movimenti.
19 http://www.gnu.org/gnu/thegnuproject.it.html
20 Ibid.
21 GNU è un acronimo ricorsivo per "GNU's Not Unix" (GNU Non è Unix) e si pronuncia gh-nu (con la g dura).
22 http://www.gnu.org/home.it.html.
23 Nel 2001, più del 67% dei fondi operativi della FSF è stato inviato da donatori privati.
24 Fonte: http://www.gnu.org/directory/all/ ; ottobre 2002.
25 "Un distributore di CD-ROM trovò che nella sua "distribuzione Linux", il software GNU costituiva il contributo maggiore, circa il 28% del totale del codice sorgente, inclusi alcuni dei principali componenti essenziali senza i quali non potrebbe esserci nessun sistema. Linux propriamente detto costituiva circa il 3%". Tratto da: http://www.gnu.org/gnu/linux-and-gnu.it.html
26 Ad esempio, i film di animazione e sempre di più gli effetti speciali cinematografici sono fatti usando strumenti basati su Linux. (Titanic, Planet of the Apes, Harry Potter, Toy Story, A Bug's life, Monsters & C. , The ice age, per fare solo qualche esempio)
27 R. Stallman: http://www.gnu.org/gnu/thegnuproject.it.html
28 R. Stallman: http://www.gnu.org/gnu/thegnuproject.it.html
29 Il testo integrale della GNU/GPL è consultabile sul sito http://www.gnu.org.

30 Altri tipi di Permesso d'autore sono utilizzati in circostanze specifiche. I manuali GNU sono protetti da Permesso d'autore, in una versione molto semplificata. Le altre licenze fanno perdere qualche libertà, pertanto, sono usate nel progetto GNU solo se indispensabili.
31 Berra, Meo, 2001, pag. 97.
32 E.S. Raymond, 1999.
33 I tempi della giustizia non sarebbero stati d'aiuto a Netscape.
34 Microsoft si stava muovendo con mezzi illeciti approfittando della posizione di quasi monopolio. La 'condanna' dell'antitrust americano è arrivata nel novembre 2002, blanda e tardiva.
35 Raymond in Di Bona, Ockman e Stone, 1999, pag. 226-7.
36 Ogni azienda proprietaria dei moduli inclusi in Communicator dovette scegliere se allearsi a Netscape, rilasciando i sorgenti con licenza open, o non farlo, con la prospettiva, però, di essere estromessa non appena rimpiazzato il suo codice.
37 Dal nome confidenziale che il team di sviluppo dava al nucleo del codice sorgente di Communicator.
38 L'occasione fu un 'party selvaggio' in uno dei night-club più grandi di San Francisco. Vicenda narrata in Di Bona, Ockman e Stone, 1999, pag. 213-22.
39 Del gruppo di promotori facevano parte, tra gli altri, Tim O'Reilly, Todd Anderson, Chris Peterson (del Foresight Institute), John "Maddog" Hall e Larry Augustin (entrambi di Linux International), Sam Ockman (del Silicon Valley Linux User's Group), Bruce Perens ed Eric Raymond. Nei giorni successivi aderirono all'iniziativa molti altri, compreso Linus Torvalds.
40 Raymond in Di Bona, Ockman e Stone, 1999, pag. 229-30.
41 Testo disponibile c/o: http://www.debian.org/social_contract.it.html.
42 Testo integrale commentato da Bruce Perens disponibile c/o: http://www.opensource.org/docs/definition.php.
43 Attualmente sono 36, l'elenco completo e le rispettive caratteristiche sono disponibili alla pagina http://www.opensource.org/licenses/.
44 Lo status giuridico di 'Open Source', come di 'Free Software', è di parola generica e nessuna restrizione legale può essere obiettata nel suo utilizzo.
45 Perché "Software Libero" è meglio di "Open Source" di Richard Stallman disponibile all'indirizzo: http://www.gnu.org/philosophy/free-software-for-freedom.it.html
46 Queste critiche non erano infondate e il caso KDE lo ha provato.
KDE, è il primo progetto Open Source di un desktop grafico per Linux, dipendeva in parte da Qt, una libreria proprietaria (la libreria è una parte di software che svolge delle operazioni ripetitive, routine, ed è riutilizzabile per programmi diversi). La libreria grafica Qt, di Troll Tech aveva una licenza che ne proibiva la modifica o l'uso con qualunque display software che non fosse X Window System. Ogni uso diverso richiedeva allo sviluppatore una licenza del costo di 1500 dollari. Ma, la possibilità di un desktop grafico per Linux era così allettante che molti utenti accettarono la situazione. Molti altri, però, lo ritennero insopportabile e decisero di avviare un progetto indipendente di desktop grafico, GNOME, e un altro gruppo avviò il progetto Harmony per produrre un clone Free con licenza LGPL (Licenza del progetto GNU apposita per librerie che consente l'uso delle stesse anche per lo sviluppo di software proprietario) di Qt.
Nel momento in cui GNOME divenne utilizzabile, Qt, per evitare l'abbandono del suo prodotto in ambiente Linux e per smorzare le ostilità, fu rilasciato sotto una nuova licenza, la QPL (classificabile come software libero non compatibile con la GPL e come conforme OSI) che rientrava - di poco - nella Open Source Definition (KDE, Gnome ed Harmony sono progetti tuttora molto attivi).
47 Argomenti poi usati da alcuni 'concorrenti' proprietari che si affidano alle tecniche di vendita FUD (Fear Uncertainty Doubt) che consiste nello stimolare - anche con la menzogna - sensazioni di paura, incertezza e dubbio nel cliente in modo da indurlo a preferire certi prodotti, i propri o di aziende 'amiche', a danno della concorrenza.
48 http://www.gnu.org/philosophy/free-software-for-freedom.it.html, cit.
49 È nuovo per il software ma non nel mercato preso nella sua complessità che conosce già da tempo queste forme di commercio. Dimostrazione concreta più immediata è quella dell'industria alimentare, che si basa su ricette di cucina che sono di pubblico dominio per realizzare i propri prodotti che poi confeziona e distribuisce ma nulla impedisce di cucinare le stesse cose 'fatte in casa'.
Lo stesso è per il Software Libero dove troviamo le 'Distribuzioni' che forniscono un prodotto confezionato pronto all'uso, ma il Software Libero è estremamente versatile perché composto, come un puzzle, da tanti piccoli pezzi e la disponibilità dei sorgenti consente di ricomporli per le esigenze più disparate.
50 Sul mercato questa soluzione Linux era già molto diffusa. I dati di IDC (http://www.idc.com/) o di altri consulenti alla fine del 2001 indicavano che Linux era stato installato sul 26.9% dei server con trend di crescita estremamente significativi.

51 Molti paesi stanno valutando la possibilità o hanno già approvato leggi per l'adozione del Software Libero nella pubblica amministrazione e nella scuola.
52 Linux nasce per l'Intel386 e funziona tutt'oggi per quell'architettura.
53 Con il termine 'Distribuzione' si indica sia il prodotto che l'azienda che lo genera.
54 Nucleo, nocciolo. È la parte del sistema operativo che permette al computer di avviarsi, di comandare tutti i dispositivi e di eseguire programmi.
Per avere un Sistema Operativo, al kernel va aggiunta una 'collezione' di altri programmi che consentono di far funzionare il computer, ad esempio, le interfacce utente, con cui l'utente 'dialoga' col computer in modo testuale (usando un linguaggio codificato che va appreso) oppure con l'interfaccia grafica (GUI, o Graphical User Interface) che consente di dialogare in modo più intuitivo selezionando delle icone.
55 http://www.redflag-linux.com/eindex.html
56 Che si prelevano da Internet.
57 http://www.debian.org/News/2002/20020719
58 Alpha, ARM, HP PA-RISC, Intel x86, Intel IA-64, Motorola 680x0, MIPS, MIPS (DEC), PowerPC, IBM S/390, SPARC.
59 Elenco dei pacchetti suddiviso in categorie: http://packages.debian.org/stable/
60 http://www.linuxcare.com/
61 http://www.prosa.it/
62 http://www.oreilly.com/
63 http://www.hopslibri.it/
64 http://www.apogeonline.com/Openpress
65 GNU Free Documentation License; dettagli all'indirizzo: http://www.gnu.org/licenses/licenses.html
66 Da un articolo Alessandro Rubini che "scrive software libero per vivere e sostiene il software libero come missione" (http://www.it.gnu.org/philosophy/software-libre-commercial-viability.it.html)
67 Dopo il vaglio della censura.
68 Cioè vietata dalla licenza.
69 Per una trattazione completa dei fatti accennati rimando a Gubitosa, 1999. E-book disponibile gratuitamente alla pagina: http://www.apogeonline.com/ebook/90017/scheda
70 Lett. Da pari a pari. Si tratta di software che permette di unirsi ad una 'rete paritetica' dove non c'è una gerarchia ed ogni utente è sia server sia client senza dipendere da un nodo centralizzato. Il sistema consente lo scambio tra utenti che mettono in condivisione le proprie risorse digitali come musica, film, testi, programmi, ecc.
71 Artt. 85 e 86 del Trattato CE, che vietano rispettivamente gli accordi tra imprese che restringano il gioco della concorrenza e l'abuso di posizione dominante.
72 http://www.attivissimo.net/rimborso_windows/istruzioni.htm
73 Il testo del Digital Millennium Copyright Act statunitense è disponibile in inglese all'indirizzo:
http://www.eff.org/IP/DMCA/hr2281_dmca_law_19981020_pl105-304.html
74 Il testo della direttiva 2001/29/CE (EUCD) è consultabile all'indirizzo: http://europa.eu.int/smartapi/cgi/sga_doc?smartapi!celexapi!prod!CELEXnumdoc&lg=it&numdoc=32001L0029&model=guichett.
75 Molti casi confermano i danni agli utenti ed ai ricercatori di queste normative.
Per un confronto approfondito tra EUCD e DMCA ed un approfondimento della questione 'danni', si vedano gli indirizzi:
http://www.softwarelibero.it/progetti/eucd/analisi.html
http://www.eff.org/Legal/recent_legal.html
http://www.sims.berkeley.edu/~pam/papers/Samuelson_IP_dig_eco_htm.htm
76 http://www.freesklyarov.org/
77 Per un approfondimento della questione, si vedano gli indirizzi:
http://www.softwarelibero.it/GNU/nemici/brevetti.shtml
http://lpf.ai.mit.edu/Patents/
78 "Non è pensabile che un ufficio brevetti possa valutare lo stato dell'arte, quando l'arte in questione copre tutto lo scibile umano (in quanto ogni concetto astratto può essere messo nella forma brevettabile di "programma per elaboratore")": http://www.linux.it/GNU/nemici/brevetti.shtml.
79 grazie ad un trucco linguistico: "Se volete brevettare un programma dovrete formulare la domanda in maniera un po' più tortuosa; non dovete scrivere che avete escogitato un algoritmo matematico per ordinare gli elementi di una matrice in ordine crescente (esiste e si chiama algoritmo "bubble sort"), ma che avete inventato un apparato hardware che permette di inserire una serie di numeri casualmente da un dispositivo di input e che li trasforma in una sequenza ordinata su un rotolino di carta". In Carlini 2002, pag. 108.

80 Questo è in contrasto con la ratio della norma sul brevetto che ricambia con lo sfruttamento esclusivo il fatto di svelare e descrivere l'invenzione in modo che possa essere immediatamente acquisita al patrimonio tecnologico di tutti.
81 l'Ufficio Brevetti Europeo ha già approvato più di 20.000 di tali brevetti, (dal brevetto sul formato grafico JPEG alla diagnosi automatica (qualunque diagnosi), dal confronto della pronuncia dell'allievo con la pronuncia dell'insegnante, al ridimensionamento di una finestra grafica quando e` oscurata da un'altra finestra, includendo la conversione di nomi da una convenzione ad un'altra per rappresentarli) arrivando al punto di piegare le normative, diramando direttive per gli esaminatori in diretto contrasto con la legislazione vigente. La "galleria degli orrori" raccolta da Foundation for a Free Information Infrastructure (FFII) include sia esempi di brevetti malfatti (http://swpat.ffii.org/vreji/pikta/mupli/index.en.html), sia esempi di impedimenti allo sviluppo software a causa di brevetti (http://swpat.ffii.org/vreji/pikta/index.en.html).

4 Il Software Libero riletto attraverso le teorie di North.

In questo capitolo conclusivo, utilizzo lo schema teorico del North, per collegare tra loro tutti gli elementi fin qui esposti ed analizzare la performance di lungo periodo del 'Sistema di allocazione del Software'.

4.1 Il 'Sistema software'.

Volendo richiamare un'immagine per illustrare la situazione attuale, si può adattare la stessa metafora del gioco usata dal North.
Il 'luogo' dove tutto si svolge, il 'campo di gioco' dove avviene produzione e scambio di software, è sia un territorio reale (regioni, nazioni, continenti, ecc.) sia un metaterritorio virtuale, rappresentato dalle Reti, di cui Internet fa parte. Territorio reale e virtuale sono collegati ma sono due entità da tenere logicamente separate.
Qui gli attori o 'giocatori', per mantenere la metafora, sono idealmente suddivisi in due grandi squadre che possiamo chiamare 'Proprietaria' e 'Libera'; ciascuna segue le 'regole di gioco' per la produzione (lavoro dipendente retribuito e/o cooperazione volontaria), per la distribuzione e lo scambio (vendita e/o dono) delle proprie risorse. Una parte delle regole è usata esclusivamente dalla propria squadra, un'altra parte è comune. Le regole (istituzioni) formali sono comuni a tutte le squadre. Alcune 'vincono la partita' giocando lealmente, nell'osservanza delle regole, magari interpretandole in modo 'creativo' pur restandone all'interno, altre "squadre vincono [...] perché trasgrediscono sistematicamente le regole"[1].
Un'altra squadra presente in campo è quella dell'Utenza, che segue regole proprie ed influenza il gioco delle altre due.
Le strategie dei giocatori, messe in atto per vincere la partita, possono produrre effetti anche inattesi e in punti lontani del campo rispetto a quello in cui si sta svolgendo l'azione.
La correttezza del gioco "dipende dalla efficacia dei controlli e dalla severità delle punizioni"[2]. Le istituzioni formali, però, valgono contemporaneamente per tutti i giocatori, solo per singole porzioni di territorio, per questo, gli arbitri (terzi garanti) le possono far valere solo nella loro porzione di campo. In ogni caso, anche in presenza di accordi tra organizzazioni territoriali, parte del campo è senza controllo. Inoltre, le regole formali possono cambiare anche rapidamente sotto la pressione delle diverse squadre.
Il 'gioco', molto complesso, è un vero e proprio sistema, e il campo ha confini che mutano e che possono presentarsi sia rigidi, sia quasi inesistenti, nello stesso momento a seconda del punto di vista considerato, infine, i singoli giocatori o gli arbitri possono 'cambiare maglia' passando da una all'altra ciclicamente (un programmatore che lavora in un'azienda Proprietaria oppure un Utente può fare Free Software la sera per hobby) o una sola volta (un'azienda Proprietaria rilascia i sorgenti del proprio software sotto GNU/GPL).

Per comprendere l'evoluzione del 'Sistema software', occorre considerare i vari frammenti esposti nei capitoli precedenti ai diversi livelli dello svolgimento del 'gioco' e come si sono influenzati reciprocamente nel tempo.
Le attività di produzione, distribuzione e scambio del bene Software o come direbbe Polanyi le "forme di integrazione"[3] dell'economia del software, si articolano sia nell'ambito dello scambio di mercato, sia in quello della reciprocità[4]

In una prima fase, che corrisponde al periodo della nascita e del primo sviluppo della scienza informatica, era prevalente il sistema di regole della comunità scientifica, poi trasfuso nelle comunità hacker, basato in gran parte sulla reciprocità come sistema di allocazione delle risorse, e sulla cooperazione e collaborazione volontaria per la produzione delle stesse.
All'interno delle comunità di ricerca, allora appartenenti ad apparati educativi (università) e in minor misura politici (centri di ricerca della difesa) ed economici (laboratori di ricerca interni alle imprese), era normale un'ampia condivisione degli strumenti e delle conoscenze. Gli scambi tra comunità diverse erano presenti ma disagevoli senza una rete informatica come quella attuale e senza la possibilità di un contatto costante tra le persone.

I computer e gli altri strumenti informatici erano costosi e assai ingombranti. Poche realtà erano in grado di acquistarli e gestirli, ogni macchina aveva il suo sistema, hardware e software erano strettamente legati, pertanto, ancora non poteva esistere un mercato di massa per il software.
Lo scambio di mercato è diventato la forma di integrazione preminente nel sistema del software solo dopo numerose innovazioni ed eventi: sono diminuite le dimensioni e il prezzo dei componenti, gli hacker dell'hardware hanno creato i computer personali e vecchie e nuove aziende ne hanno facilitato la diffusione nelle PMI e a livello domestico, e infine, sono stati implementati i sistemi 'portabili' (non legati ad un solo hardware).
Il mercato di massa, però, si rivolgeva prevalentemente ad un'utenza diversa rispetto a quella interessata dalla forma della reciprocità, che ha continuato a mantenere le proprie regole di produzione cooperativa e di distribuzione basata sul dono (dall'84 anche grazie al movimento di Stallman), come illustrato nei precedenti capitoli. Continuando parallela a quella di mercato.
Dato che la produzione delle comunità hacker, si è concentrata principalmente sugli strumenti di cui gli stessi programmatori hanno bisogno e dato che le comunità hanno tra i bisogni fondamentali, per la loro stessa esistenza quello di comunicare, una delle attività più avanzate è stata proprio quella dello sviluppo dell'infrastruttura delle Reti mondiali (World Wide Web) per questo scopo.
Anche l'idea della comunicazione tra computer, al di là delle esigenze interne delle organizzazioni, è stata cooptata dalle aziende e spinta nell'area dello scambio di mercato con lo sviluppo di Internet commerciale.
L'ulteriore diminuzione dei costi dell'hardware e la 'semplificazione' dell'uso del computer[5], ha consentito di abbassare il livello di competenza tecnica necessario per il suo utilizzo a livello desktop e ha fatto in modo che Internet si potesse diffondere enormemente tra il pubblico, tanto da indurre molti ad acquistare un PC solo perché attratti da Internet.
Questi fatti hanno provocato un 'effetto collaterale' non trascurabile: il collegamento tra aree del campo e giocatori che prima difficilmente interagivano, quindi, anche tra le forme di integrazione dello scambio di mercato e della reciprocità. L'apertura sistemica ha permesso il contatto di parte dell'utenza (tra quelli con adeguate competenze) dell'area del mercato con i prodotti e le informazioni provenienti dall'area della reciprocità.

La Free Software Foundation, è nata per ricreare l'ambiente inclusivo del MIT e il suo modo di lavorare, collaborativo e libero, impostato sui valori della reciprocità, che rischiava di essere compromesso dall'avanzamento del metodo di sviluppo esclusivo tipico del mercato. In tal senso non era 'culturalmente' indicata a gestire il rapporto tra Software Libero e Mercato. A questo scopo, ma solo dopo l'affermazione di GNU/Linux che completa il Sistema GNU e che lo rende un prodotto 'maturo' per essere offerto sul mercato di massa, dodici anni dopo, è intervenuta la dell'Open Source Initiative.
L'intento di questa organizzazione è di dare un supporto nel governare ed incentivare il successo del Software Libero e del suo metodo di sviluppo anche in area commerciale, situazione prevista ma gestita dalla FSF in modo filosofico-ideologico piuttosto che pragmatico, fatto che teneva lontani molti operatori economici. Il movimento Open Source si è proposto come una sorta di mediatore culturale a cavallo tra due mondi.
La nascita dell'Open Source Initiative, criticata dagli appartenenti più radicali del movimento Free Software e ritenuta da molti come un inopportuno 'fork' è in realtà un adattamento ad una nuova situazione che nel frattempo era maturata.
Il movimento Open Source ha aiutato la congiunzione tra il modello di produzione Libero legato alla reciprocità e il modello di produzione e distribuzione di mercato, in questo si distingue qualitativamente dal movimento Free Software.

Oggi, sempre più aziende impostano la propria strategia parzialmente o completamente sul Software Libero; l'azione delle distribuzioni di sistemi GNU/Linux commerciali, tendente alla semplificazione dell'uso per avvicinarlo ad un pubblico sempre più vasto e privo di conoscenze tecniche, ha aperto la strada alla prospettiva del Software Libero di massa.
Mentre in passato c'erano due mondi relativamente chiusi nei propri confini, con scambi relativamente ridotti da uno all'altro, i cambiamenti tecnologici appena ricordati, hanno virtualmente annullato l'esistenza dei confini portando i due mondi a mescolarsi in vario modo; ognuno mantiene un nucleo più o meno grande di regole esclusive, da rispettare 'quando si gioca nel campo dell'avversario', ma il Software Libero Commerciale le combina e le adopera entrambe.
Ora, dato che è proprio dall'attività nella reciprocità che è partito il cambiamento, ne è stata sostanzialmente il motore e, dato che, un modello aperto come quello del Software Libero, consente un più rapido adattamento alle situazioni che mutano nel tempo e nello spazio, il campo della reciprocità non ha subito contraccolpi improvvisi.

La parte di campo, invece, dove si segue il modello di scambio di mercato, è stata irrimediabilmente alterata dall'introduzione del Software Libero Commerciale, con risorse alternative di qualità ed il suo modello produttivo.
Se il Software Libero, forte anche del fatto di essere disponibile gratuitamente e liberamente per programmatori, organizzazioni economiche e utenti, dovesse diffondersi massicciamente nel mercato[6], - come i tassi di crescita fanno pensare - gli operatori che agiscono in regime 'Proprietario', dovranno inevitabilmente rivedere le proprie strategie alla luce del cambiamento, operazione non priva di attriti, dubbi e perplessità, per il fatto che occorre conciliare due sistemi di allocazione diversi in cui modelli soggettivi degli attori, logica dell'azione individuale, capacità di rapportarsi all'ambiente, motivazioni, incentivi e utilità attesa, sono differenti.

La situazione attuale del 'Sistema Software', nella combinazione della coppia produzione-distribuzione, si può riassumere nella sottostante "tabella" che combina, sotto quest'aspetto, i modelli del Software Libero e Proprietario:

Il 'Sistema Software' nella combinazione della coppia produzione-distribuzione:

A - Produzione chiusa (non condivisa)
B - Produzione Libera (condivisa)
C - Distribuzione a pagamento(di beni e/o servizi)
D - Distribuzione gratuita (di beni e/o servizi)

-(quadrante 1: incrocio tra A e C)
Software Proprietario [Mercato]

-(quadrante 2: incrocio tra B e C)
Software Libero Commerciale [Mercato]

-(quadrante 3: incrocio tra A e D)
Freeware e simili [Mercato]

-(quadrante 4: incrocio tra B e D)
Software Libero [Reciprocità]

Il Software Proprietario (quadrante 1) segue una produzione di tipo chiuso, con segretezza del codice sorgente, prevalentemente interna alle imprese in modo da preservare il segreto; quando si avvalgono di collaborazioni esterne, esse sono sottoposte alle medesime condizioni di segretezza. Il software è orientato al mercato, ossia, è pianificato in anticipo in base alle esigenze che si ritiene abbiano i propri utenti (es. software pacchettizzato per il mercato di massa) oppure è realizzato su richiesta esplicita dei propri clienti (es. soluzioni per l'impresa). La forma di allocazione rientra nel modello dello scambio di mercato e l'oggetto dello scambio sono le licenze d'uso del software ed eventuali servizi accessori personalizzati o standardizzati e pacchettizzati e conteggiati a parte. Tra le strategie adottate dalle imprese che producono software Proprietario c'è anche quella di regalare alcuni dei propri prodotti con forme e modalità che cambiano a seconda dei casi. Il Freeware e similari (quadrante 3) si può in tal senso considerare un sottogruppo del software Proprietario, che a parte il prezzo segue le stesse forme e dinamiche.
Il Software Libero (quadrante 4) segue una produzione di tipo totalmente aperto consentendo a chiunque desideri collaborare e cooperare di farlo senza alcun ostacolo che non sia quello della competenza nella programmazione, dei vincoli tecnici e della struttura collaborativa di ciascun progetto. La produzione è orientata al gruppo e nasce da reali esigenze della comunità. La totale apertura comporta che non vi siano ostacoli nemmeno di natura economica; il Software Libero è un bene pubblico puro e la forma di allocazione rientra nel modello della reciprocità (con scambio di doni moderno, prevalentemente indiretto e impersonale).
Il Software Libero Commerciale[7] (quadrante 2), infine, attinge dal 'calderone magico' del Software Libero e adatta il prodotto da offrire alla propria clientela o ne sviluppa di proprio, seguendo in parte lo stesso modello collaborativo (con programmatori stipendiati che collaborano ai progetti Free e che sviluppano soluzioni di diretto interesse per l'impresa). Offrendo un bene acquisibile liberamente, il vero oggetto dello scambio è l'insieme dei beni e servizi che vanno al di la del software in sé: scelta, aggregazione e

adattamento del software, trasferimento, supporto, manuali, confezione, garanzie, personalizzazione, assistenza, consulenza, formazione, ecc. la forma di allocazione di questi beni è quella dello scambio di mercato.

Va ricordato comunque che nella realtà operano pochi attori che si mantengono puri su un versante o sull'altro; per cui si potranno avere attori singoli o organizzazioni economiche Proprietarie che offrono anche Software Libero e viceversa, in proporzioni variabili, incontaminate solo agli estremi.
Questo può dipendere dalle dimensioni aziendali, che vanno dal programmatore singolo alla multinazionale, dal tipo di clientela/utenza, dal prodotto offerto (pacchettizzato per il mercato di massa o soluzioni per l'impresa) o dal mercato di riferimento (globale/locale), dalla forza contrattuale degli attori e così via.

4.2 Cambiamento e sviluppo nel mercato del software.

Il Software Libero sta avendo ultimamente una crescita quantitativa e qualitativa eccezionale e il suo modello sta conquistando posizioni in un mercato dominato da altre soluzioni, tecnologie e standard ben affermati e che traggono beneficio da tre dei quattro meccanismi di rinforzo, indicati da Arthur[8], in grado di impedire l'affermazione di altre tecnologie, anche quando considerate migliori e più efficienti: gli effetti di apprendimento, gli effetti di coordinamento e le aspettative adattive. Questi effetti contribuiscono a consolidare e conservare la situazione, quantomeno per inerzia. La mancanza del quarto - elevati costi fissi o di installazione - agevola la possibilità di cambiare, ma non basta da sola a rendere comprensibile il cambiamento in atto.

Nella sua struttura teorica, il North, indica le organizzazioni e gli imprenditori che le dirigono come il motore del cambiamento. Esse nascono intenzionalmente sulla base delle opportunità consentite dall'insieme dei vincoli[9]; sono gruppi di persone unite dal comune proposito di raggiungere un fine. Il contesto istituzionale influisce sia sulla nascita, sia sull'evoluzione delle organizzazioni che a loro volta, nello sforzo di realizzare i propri fini si rivelano come i soggetti più significativi nel ruolo di agenti del cambiamento istituzionale (forma e direzione). Il cambiamento, sottolinea North "è un processo complicato perché le variazioni marginali possono essere la conseguenza di modificazioni delle regole, dei vincoli informali e della natura e efficacia dell'applicazione della legge. Le istituzioni, inoltre, si trasformano secondo una logica incrementale piuttosto che a salti discontinui. La ragione di ciò, ma anche del fatto che le discontinuità (come una rivoluzione o una conquista) non sono mai completamente tali, risiede nel radicamento sociale dei vincoli informali"[10].
Il North costruisce la sua teoria delle istituzioni su quella del comportamento combinata alla teoria dei costi di transazione, in tal modo, documenta le ragioni dell'esistenza delle istituzioni e ne chiarisce l'influenza sulle attività economiche.
Le istituzioni hanno un ruolo essenziale quando negoziare implica dei costi; questa idea innovativa, introdotta da Coase nel 1960 con il suo famoso saggio 'The problem of Social Cost'[11], mette a fuoco il costo dell'informazione come costituente fondamentale dei costi di transazione, che comprendono anche, i costi di misurazione delle caratteristiche dell'oggetto di scambio, quelli per la tutela dei diritti e quelli di garanzia di applicazione e rispetto dei contratti[12].
Dopo aver ricordato i problemi dell'analisi teorica della cooperazione volontaria[13], elemento mancante nella dottrina neoclassica per la costruzione di una teoria dell'azione in grado di decifrare il funzionamento delle economie reali, il North, si ricollega al teorema di Coase dove sostiene che "quando negoziare è senza costi si ha la soluzione concorrenziale efficiente predetta dalla teoria neoclassica"[14]. Perché ciò si avveri occorre trovarsi in un contesto di piena concorrenza che elimini l'informazione incompleta e asimmetrica, con una retroazione efficiente delle informazioni (per avvicinarsi alla condizione di costi di transazione nulli). In tale situazione gli attori che dovessero imboccare modelli erronei e diversi verrebbero ricondotti al corretto comportamento dal processo di retroazione che punisce i devianti.
La realtà dei fatti, sottolinea North[15], è molto lontana da queste condizioni: l'azione si basa su modelli soggettivi, assai spesso, erronei e su informazioni incomplete, quindi, anche la retroazione delle informazioni non può essere efficiente. Inoltre le regole formali, in situazione di costi di transazione non nulli, sono influenzate da coloro che hanno un più forte potere contrattuale. L'attività di questi attori, nel perseguire i propri obiettivi privati, può determinare, se le circostanze sono favorevoli, il prodursi di sistemi istituzionali che in seguito si rivelano socialmente efficienti.

Il recupero di posizioni del Software Libero, in un mercato già completamente occupato in ogni sua nicchia, è dato dal suo sistema istituzionale di riferimento che favorisce l'abbassamento dei costi di transazione e in parte quelli di produzione attraverso comportamenti di tipo cooperativo. Questo comporta un abbassamento della soglia d'ingresso sul mercato e del livello di rischio, e da qui, più in generale, incoraggia la concorrenza, la flessibilità e l'innovazione, con un migliore rendimento complessivo, che si avvicina un po' di più alla soluzione concorrenziale perfetta, predetta dalla teoria neoclassica.
All'inverso, il sistema istituzionale di riferimento del modello Proprietario, disincentiva gli esiti cooperativi tra attori nel mercato del software, per la pressoché totale mancanza di scambio di informazioni: le informazioni che vengono fornite all'esterno sono solo quelle indispensabili. L'implementazione avviene praticamente tutta all'interno nelle organizzazioni economiche di tipo proprietario, che si devono fare totale carico dei costi di produzione e dei rischi d'impresa. La chiusura ed il segreto servono per poter mantenere una posizione di monopolio sul codice e usufruire delle asimmetrie informative sui propri prodotti[16].

Riguardo la logica dell'azione nel modello del Software Libero, il comportamento derivante dal rispetto delle sue istituzioni informali, che richiede accesso libero e illimitato all'informazione e incoraggia alla collaborazione, condivisione del lavoro, delle abilità ed esperienze e degli strumenti per realizzarlo, influenza positivamente le attività economiche e favorisce la crescita della concorrenza. La massimizzazione della ricchezza degli attori che si dedicano al Software Libero anche commerciale, avviene attraverso vincoli liberamente accettati (che hanno come massima espressione la licenza GNU/GPL) che conducono a comportamenti di tipo altruistico, all'interno di un sistema istituzionale che tende, così, ad essere socialmente più efficiente.
Ripercorrendo i punti delle 'regole del gioco hacker', analizzati nel secondo capitolo, si possono individuare le singole istituzioni informali che portano verso un esito cooperativo volontario, che conduce a guadagnare dallo sviluppo degli scambi piuttosto che ad esiti collusivi e monopolistici.
Innanzitutto, la libertà di informazione che, per il software, ha come precondizione la disponibilità del codice sorgente, e come fulcro la condivisione delle informazioni sui programmatori e sulla programmazione. Per estensione è applicata a tutto il sapere in senso lato, creando un sistema che favorisce il recupero e la ricombinazione creativa delle idee, fondamentale per il procedere del progresso in qualsiasi disciplina.
Questo, dà luogo a strategie di azione sul versante produttivo, che portano alla cooperazione dei soggetti e ad una condivisione delle informazioni, dei mezzi e degli strumenti portando vantaggi che incidono sui costi totali.
La produzione del Software Libero, tra l'altro poggia su basi ideologiche forti, quelle hacker, tanto simili alle regole della comunità scientifica (esposte nel secondo capitolo), che favoriscono questi atteggiamenti.

Negoziare implica dei costi. I costi di transazione sono costi variabili che dipendono dalle condizioni di incertezza e dalla carenza di informazioni principalmente sull'offerente e sulla qualità del prodotto, che possono dare spazio a comportamenti opportunistici. Dato che i costi di transazione, occupano una voce importante per la performance economica, il loro abbassamento la influenza notevolmente. Tra i costi di transazione il più importante è quello di informazione.
Una delle regole cardine delle comunità di sviluppo del Software Libero presente anche nel suo adattamento commerciale, è quella che prescrive la massima libertà di informazione che ha come precondizione l'accesso al codice sorgente del software ma che viene estesa a tutte le altre fonti di informazione, anche non strettamente legate alla programmazione, reperibili presso le comunità di sviluppatori, che in base ai dati di ricerca come la FLOSS[17] è caratterizzata, tra l'altro, da un elevato livello di istruzione dei partecipanti e/o background professionale nel settore IT.
La condivisione libera (e gratuita) del software e delle informazioni sui programmi (inclusi eventuali problemi e difetti), sui modi, tempi e soggetti che hanno effettuato la scrittura e le modifiche del codice (dalle credit list), la possibilità di contattare gli autori o di trovare all'interno della comunità (nelle apposite liste di discussione) persone esperte cui chiedere informazioni o aiuto, incidono direttamente su un'altra voce dei costi di transazione, ovvero, sui costi di misurazione delle caratteristiche del software oggetto di scambio. A questo va aggiunto che il fatto di seguire le regole del metodo scientifico pone il Software libero nelle condizioni di essere valutabile da chiunque abbia la competenza per farlo. Quando l'acquirente è un'azienda, spesso può avvalersi del proprio personale tecnico per le verifiche del caso. In mancanza, potranno comunque affidarsi ad aziende con marchi conosciuti e buna reputazione, in grado di offrire adeguate garanzie sul Software Libero che commercializzano.
All'inverso, il software Proprietario non consente di attenuare i costi di un'informazione incompleta e asimmetrica. Per cui bisogna fidarsi di ciò che viene dichiarato dal produttore o fidarsi della garanzia data

dal prestigio del marchio. Per superare il problema possono essere offerte delle garanzie o un periodo di prova ma il prodotto in sé rimane celato dietro il segreto industriale restando preclusa una verifica diretta.
Le caratteristiche di totale apertura informativa del Software Libero, consentono la retroazione delle informazioni tra gli operatori economici tra loro e con l'utenza che si trova in una posizione più simmetrica, di relativo equilibrio nella forza contrattuale o addirittura, nel caso dell'utente-sviluppatore, è virtualmente inclusa nell'organizzazione aziendale.
Gli altri costi di transazione, ossia quelli di tutela dei diritti di proprietà, e quelli di garanzia e rispetto dei contratti sono diversi rispetto a quelli del modello Proprietario per la parte che dipende direttamente dal tipo di licenza d'uso che accompagna il Software.
La tutela dei diritti di proprietà nei due modelli è sostanzialmente diversa per via della forma, che in ambito Proprietario prevede dei divieti dove il Software Libero dà delle libertà[18]:
Libertà di eseguire il programma, per qualsiasi scopo (libertà 0)[19].
Libertà di studiare come funziona il programma e adattarlo alle proprie necessità (libertà 1). L'accesso al codice sorgente ne è un prerequisito.
Libertà di ridistribuire copie in modo da aiutare il prossimo (libertà 2).
Libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio (libertà 3). L'accesso al codice sorgente ne è un prerequisito.
La violazione dei divieti oltre ad essere difficile da individuare danneggia l'autore e/o l'azienda proprietaria, che perde gli introiti monetari dati dalla vendita delle licenze (che in questo senso riguarda soprattutto i distributori di pacchetti software standardizzati).
Il danno derivante dalla violazione delle libertà (ad esempio la distribuzione come software Proprietario) provoca un danno qualitativamente diverso: interrompe il flusso di informazioni danneggiando la comunità di sviluppo. Le organizzazioni per la promozione del Software Libero che ospitano i progetti di sviluppo, si occupano anche della tutela giuridica dalla violazione del copyright di cui sono detentrici, che è in questo modo assunta collettivamente[20].
Riguardo la garanzia e rispetto dei contratti i costi relativi si avvicinano. Nel Software Libero non c'è garanzia. Ad esempio, nel preambolo della GPL, che ne spiega anche la ragione, si legge: "per proteggere ogni autore e noi stessi, vogliamo assicurarci che ognuno capisca che non ci sono garanzie per i programmi coperti da GPL. Se il Programma viene modificato da qualcun altro e ridistribuito, vogliamo che gli acquirenti sappiano che ciò che hanno non è l'originale, in modo che ogni problema introdotto da altri non si rifletta sulla reputazione degli autori originari". Sarà eventualmente il distributore a fornire una qualche garanzia simile a quelle con cui viene distribuito il software Proprietario, che tuttavia, risulta molto debole nella maggior parte dei casi, salvo accordi specifici con il cliente. Il software pacchettizzato per il mercato di massa è però escluso da questa eventualità.
Anche i costi di produzione sono diversi nei due modelli. Come descritto nel III capitolo, la particolare natura immateriale del software lo rende riproducibile identico all'originale e 'trasportabile' ovunque nel mondo (purché si possieda un computer ed un collegamento alla Rete) ad un costo irrisorio. Inoltre la sua struttura dei costi di produzione è affetta da diseconomia di scala dei costi di sviluppo rispetto alla dimensione del prodotto. Laddove è possibile 'riciclare' il software[21], così come consentito dalle libertà delle licenze, il risparmio dato dal fatto di non dover contare solo su energie interne, svincolando risorse umane e finanziarie impiegabili per la scrittura di software aggiuntivo, per il miglioramento, l'adattamento, per i servizi, ecc.
Questo, come già accennato, abbassa la soglia d'ingresso in molti comparti del mercato IT[22]. Attraverso il Software Libero commerciale gli operatori possono offrire prodotti e servizi, adattando il costo alla propria clientela. Questo porta ad una migliore allocazione delle risorse ed è segno della presenza di istituzioni che portano ad un esito socialmente più efficiente rispetto al sistema Proprietario. Ad esempio, nei paesi in via di sviluppo consente di rivolgersi ad imprese locali e di mantenere localmente le risorse, che altrimenti andrebbero verso i paesi industriali produttori di software Proprietario.
Si può affermare a questo punto che le istituzioni proprie del sistema del Software Libero sono più modulabili, rispondono meglio al cambiamento delle preferenze, rispondono meglio al cambiamento dei prezzi relativi[23].

4.2.1 Capitale Sociale nel mercato del software.

Ma, l'esito più importante del sistema istituzionale del Software Libero, che a parità di condizioni lo distingue nettamente da quello proprietario, è la presenza di un elevato capitale sociale[24], di natura positiva[25], vale a dire, quello che aiuta lo sviluppo economico generando informazioni e fiducia, il tutto, all'interno di un contesto che estende lo spazio per la formazione di modelli flessibili di organizzazioni economiche in un quadro concorrenziale.

In questo sistema, il capitale sociale è il sottoprodotto dell'attività di programmazione del software a sorgenti aperti sottoposta a licenze Free ed Open Source; esso non è divisibile né consumabile e i suoi vantaggi non sono appropriabili individualmente, ma vanno a tutti coloro che partecipano alla rete. Queste caratteristiche però sono anche quelle del Software Libero, ovvero, del prodotto.
Per le aziende che propongono sul mercato il Software Libero[26], trattandosi di un bene collettivo, può non esserci un incentivo diretto ad investire e/o contribuire alla produzione di software direttamente presso le comunità di sviluppo, mettendo in atto in tal modo un comportamento da free rider; l'attività di free riding, tuttavia, produce un effetto paradossale sia sul prodotto che sul sottoprodotto: ne accresce indirettamente il valore. Infatti, la maggior diffusione del particolare software oggetto di scambio, e del Software Libero in generale, ne accresce il valore d'uso e nel contempo aumenta la possibilità che arrivino nuovi partecipanti, con il relativo incremento di relazioni e di risorse per l'azione (informazioni, competenze, fiducia, reputazione, ecc.).
Un altro effetto paradossale, si ottiene invertendo i concetti di prodotto e sottoprodotto: l'esito non cambia. In altre parole, si potrebbe considerare, provocatoriamente, il Software Libero (implementato nel contesto della reciprocità), come un residuo delle interazioni o il pretesto per attivarle; l'ipotesi è plausibile, se si considera che la ricerca FLOSS ha rilevato che le prime due ragioni che inducono ad unirsi e spingono a continuare la partecipazione, sono 'learn and develop new skills' e 'share knowledge and skills' (mutano le percentuali ma non la posizione), che quasi il 70% spende meno di 10 ore settimanali nello sviluppo di Software Libero e che mediamente i progetti contano 5,1 autori[27].

Il ruolo dei fattori culturali e delle reti di relazioni sociali, incide sulla performance economica.
Visto che, nelle reti di comunità del Software Libero, la libertà di informazione è elevata a principio assoluto, con il gioco della reputazione e i meccanismi di retroazione a garantirne la qualità[28], e, visto che quelle del Software Libero, sono reti di conoscenze personali, contatti informali fungibili o legami deboli, in continua interazione o interdipendenza, ed estese in molte direzioni operative, allora, si può comprendere come possano essere in grado di soddisfare i bisogni di informazione reciproca, in breve tempo, e mettendo a fuoco efficacemente le risorse che occorrono all'azione economica. E si può comprendere come tutto ciò produca effetti positivi sul rendimento complessivo di tutto il sistema, economico e sociale.
La differenza sostanziale del Capitale sociale generato dal modello del Software Libero rispetto a quello Proprietario, anch'esso presente e attivo, è dovuta, proprio, al fatto che la rete sociale che si determina in ambiente Free è aperta verso l'esterno; nel senso che si ha la possibilità di ingresso e partecipazione alla base come semplice utente newbie (novizio) per arrivare a qualsiasi livello a seconda della propria competenza, acquisibile sul campo, in una rete senza soluzione di continuità, tale per via dell'accessibilità libera ai sorgenti del software. Nel modello Proprietario, infatti, l'indisponibilità e il segreto dei sorgenti, obbliga ad una chiusura su molti aspetti cruciali per l'innovazione e per le strategie d'azione custoditi all'interno di reti diverse e scollegate; inoltre, il capitale sociale che ne scaturisce, in molti casi, non è di natura positiva e favorisce esiti collusivi e monopolistici che danneggiano il sistema economico e sociale.

5 Conclusioni.

5.1 Elementi in gioco nel cambiamento istituzionale.

Rispetto al mercato del software nel suo complesso, il Software Libero è ancora un fenomeno marginale, ma in forte crescita per via del successo che sta avendo il sistema operativo GNU/Linux, arrivato a maturità come prodotto in tempi recentissimi (e per alcuni versi ancora immaturo o incompleto). La sua presenza ed azione, però, sta cambiando, per ora ai margini, le regole del gioco di un settore assolutamente strategico che coinvolge praticamente tutto il resto dell'economia dei paesi sviluppati ed in via di sviluppo.
Il Software Libero, come fin qui illustrato, ha una struttura istituzionale profondamente differente rispetto a quella Proprietaria, fino ad ora indiscussa, nonostante evidenti contraddizioni[29].
Sui fronti, del prodotto, della domanda e dell'offerta, ci sono dei vantaggi innegabili che possono fornire la leva per il cambiamento, ma anche svantaggi, resistenze e costi di migrazione da un sistema all'altro che hanno la forza per renderlo più lento. Inoltre, per una parte dell'utenza (tra quelli che hanno almeno una minima cognizione della possibilità di adoperare delle alternative), una tecnologia oppure l'altra è assolutamente indifferente e non ravvisano alcuna motivazione per investire tempo e/o denaro nel cambiamento.
Il quadro istituzionale così differenziato, i vantaggi e gli svantaggi di metodo, i benefici e i costi, devono essere tenuti ben presenti se, e quando, i diversi attori dovessero decidere di migrare da un modello/tecnologia all'altro.
Occorre ricordare, per prima cosa, che una caratteristica fondamentale del prodotto software, che sia grande o piccolo, proprietario o Free, è la sua modularità, che lo rende estremamente versatile, flessibile e adattabile.
Un sistema operativo, ossia, il software che fa funzionare un calcolatore, i vari applicativi (che consentono all'utente di eseguire compiti specifici), o gli altri tipi di software, non sono elementi monolitici, ma composti da moltissimi pezzi che vanno a formare dei sistemi coerenti di parti elementari. In fase di implementazione, queste parti sono scelte tra quelle già disponibili presso fonti esterne o interne, oppure sono scritte appositamente.
Ogni 'pezzo', da solo o insieme ad altri, per l'implementazione di un programma o di un intero sistema, può essere prodotto internamente ad un'organizzazione (o da un singolo), se si segue il modello proprietario, o essere oggetto di un progetto Free; ma, solo in questo secondo caso potranno essere liberamente 'riciclati' prodotti altrui senza alcuna formalità.
In ogni caso, non si tratta di operazioni banali: più è complesso e 'penetrabile' il prodotto e maggiore dovrà essere la preparazione e la competenza necessarie per manipolarlo e proporlo sul mercato con adeguata perizia. Un prodotto Proprietario sarà penetrabile e manipolabile, con le modalità stabilite dall'autore/produttore, fino a un punto di confine, laddove un prodotto Free è invece completamente trasparente; pertanto, richiede conoscenza e competenza che dovranno essere calibrate sulla diversa penetrabilità e flessibilità del prodotto.
Per sviluppare software in ambiente Free, sia questo un piccolo modulo o un intero sistema operativo non basta lanciare un progetto e aspettare, ma, come indicato nel secondo capitolo, occorrono molti elementi perché 'prenda vita'.
La maggior parte dei progetti consiste nella mera pubblicazione del software con i sorgenti, mantenuti solo dagli autori originari, senza che si formi la massa critica di utenti e sviluppatori collaboranti in modo attivo, perché il progetto possa evolvere in modo proficuo. Il più delle volte la situazione di stallo dipende da inadeguatezza della leadership, che per condurre un progetto di successo deve avere delle doti fuori dal comune, oppure, per scarso interesse generale verso quel particolare programma, perché rimane anonimo tra i tanti disponibili, perché c'è già un sostituto migliore, o ritenuto tale, con relativa comunità di sviluppo o, non ultimo, perché nemmeno l'autore ritiene necessario uno sviluppo ulteriore.
Gli elementi da mettere insieme per avere un progetto funzionante, non sono facili da raggiungere o da riprodurre artificialmente. L'autore/leader del progetto deve essere in grado di avviare, condurre, pubblicizzare, far espandere, coordinare e governare il progetto per la sua buona riuscita, senza sapere se

e quando avrà successo e senza compenso diretto per il suo lavoro - salvo progetti attuati all'interno di aziende, enti o altre organizzazioni.
Un programma Libero, anche se limitato alla sola pubblicazione, rimane comunque come patrimonio comune e in qualsiasi momento può essere scoperto, studiato, utilizzato (tutto o in parte) o lanciato da altri appartenenti alla comunità.
Il patrimonio di 'pezzi' di ogni genere e dimensione è già molto consistente e utilizzabile come 'eredità' su cui costruire nuovo software o imprese rivolte al mercato.
Le organizzazioni commerciali che, dal lato dell'offerta in ambito di Mercato, si dedicano al Software Libero come base per il proprio business, devono calibrare i propri prodotti alla dimensione e capacità aziendale e alla dimensione e capacità dell'utenza che hanno scelto di soddisfare. Hanno comunque un prodotto modulare, flessibile e personalizzabile, in tutte le sue componenti, senza limiti imposti dall'autore/produttore, come in ambiente Proprietario.
Ma, il fatto che si manipoli Software Libero, richiede una consapevolezza profonda delle sue istituzioni informali, così da evitare una visione deterministica o romantica del fenomeno, quindi, un'adozione acritica, in grado di generare grossolani errori di valutazione che si possono facilmente trasformare in fallimenti.
I progetti liberi hanno dei tempi e delle dinamiche che non sono quelle di mercato. Si può intervenire sviluppando internamente alla propria impresa le parti ritenute mancanti o finanziando il progetto del programma che interessa perché si muova anche nella direzione di proprio interesse, ma, rimane il fatto che più grande e complesso è il programma e maggiori sono gli investimenti che occorrono per ottenere i risultati ipotizzati e che questi non sempre arrivano nei tempi e nei modi attesi[30]. Per avere voce su come procede lo sviluppo e ottenere una nuova caratteristica in modo più rapido, molte grandi compagnie commerciali come, IBM, Sun Microsystems, Hewlett-Packard[31], solo per citare le più conosciute, aiutano a mantenere il Software Libero collaborando con risorse economiche, strumentali ed umane.
Il comparto delle distribuzioni GNU/Linux commerciali pacchettizzate, inoltre, è già saturo dal lato dei grandi competitori generalisti, nonostante la giovinezza del sistema (il kernel è nato nel '91) e della sua commercializzazione, iniziata tra il '93 e il '94.
Due compagnie, Red Hat e MandrakeSoft[32], si spartiscono la maggior parte del mercato e un consorzio di distribuzioni minori sta tentando la risalita. A queste si aggiunge dal versante della reciprocità, la distribuzione Debian, organizzazione senza scopo di lucro, completamente free, unica rimasta da questo lato[33], per 'selezione naturale'.
Comunque, l'estinzione di un competitore Free dal mercato, per liquidazione, fallimento, fusione, ecc., non determina la scomparsa dei suoi prodotti Liberi che, alla peggio, smettono di evolvere, ma, restano nel patrimonio comune e altri ne possono acquisire, senza particolari oneri, l'eredità.
"La scelta di quale distribuzione adottare risulta ora molto più semplice di quanto fosse un tempo. Tra il '95 e il '96 il mercato è stato seriamente scosso e alcune delle distribuzioni commerciali sono emerse come dominanti mentre altre sono rimaste prive di aggiornamento o sono addirittura scomparse [...]. La selezione tra le distribuzioni non commerciali ad uso generale[34] è stata persino più severa con Debian (e derivati), unica superstite in questa categoria.
Come risultato di tutto questo il mercato su tre livelli delle distribuzioni (produttori principali di distribuzioni, editor di distribuzioni a "valore aggiunto" e assemblatori di CD contenenti molteplici distribuzioni) è praticamente collassato. Per essere competitivo nel 1997 un produttore (commerciale o non) doveva essere in grado di offrire un ragionevole livello di supporto ai suoi clienti [...]. Coloro che desiderano una distribuzione sempre all'ultimo grido troveranno difficile adottare una delle distribuzioni minori per la maggior lentezza degli aggiornamenti"[35].
Il passaggio da un modello Proprietario ad uno Libero per gli operatori economici, che già da tempo operano sul mercato dal lato dell'offerta, non è un intervento che si può fare in modo repentino. Comporta trasformazioni radicali dell'impresa e costi di migrazione piuttosto consistenti, per l'aggiornamento del personale, per una non semplice rivisitazione di tutta la struttura e delle modalità in cui il prodotto è proposto all'utenza[36], con i relativi rischi di una ristrutturazione radicale. In più, vanno, in parte, rivisti i metodi produttivi e le gerarchie interne, per tener conto del fatto che una parte dello sviluppo del software non è governabile direttamente dall'azienda e avviene al di fuori delle regole e della mentalità prevalente nel mercato del software, strutturata sull'idea proprietaria di monopolio sul prodotto; l'azienda che opera una tale trasformazione è, e deve rimanere, sul mercato facendo piccoli passi e tenendo conto delle diverse regole del Software Libero, se decide che può essere vantaggioso adottarlo o se ne è costretta per la nascita di concorrenza dove prima operava in monopolio.
Relativamente più semplice è iniziare direttamente con offerta articolata sul modello Free, sviluppando la struttura sullo schema ibrido che essa necessita. Un vantaggio non trascurabile per l'offerente è che gli utili non possono essere erosi dalla duplicazione abusiva del software, che affligge la parte di mercato

articolata sul modello proprietario di vendita delle licenze d'uso; infatti, il bene commercializzato, essendo costituito da servizi che, in quanto tali, non sono scindibili dall'offerente non è soggetto a questo fenomeno. Chi duplica Software Libero per disporne gratuitamente, senza riconoscere qualcosa in cambio a chi lo ha messo a disposizione, non viola alcuna norma giuridica, tutt'al più può essere accusato di non rispettare le istituzioni informali che regolano il dono.
Dal lato della domanda di software troviamo l'utenza. Anche questa, va ricordato, non è un qualcosa di monolitico. Utente è chi usa un computer in casa per divertirsi con i videogiochi, ma anche, un ospedale per gestire tutto il funzionamento della struttura, uno stato per coordinare la sicurezza nazionale, un azienda cinematografica per creare film d'animazione in 3D. Ovviamente, l'offerta per soddisfare questi utenti sarà profondamente diversa.
Utenti domestici, piccole aziende, artigiani, studenti, ecc. quasi certamente acquisiranno (legalmente o meno) il software attraverso pacchetti pronti da installare, o preinstallati, e lo useranno così com'è o, se occorre, con assistenza limitata alle fasi iniziali o di saltuaria piccola manutenzione.
Aziende medie e grandi, enti, pubbliche amministrazioni, utenti con esigenze particolari, ecc. più probabilmente avranno bisogno di qualcosa di molto più articolato, di Soluzioni, ovvero, della fornitura di hardware, software - di cui una parte andrà personalizzata su richiesta dell'utente - consulenza, assistenza, ecc.
Gli utenti acquistano beni e servizi basati sul Software Libero, solo se ottengono in cambio un valore aggiunto per loro di qualche utilità o non acquisibile altrimenti, ad un prezzo migliore, con migliori garanzie rispetto alla concorrenza o se riscontreranno altri vantaggi di medio/lungo periodo; meno di frequente per motivi 'ideologici', anche se per alcuni saranno fondamentali nell'orientare la scelta.
Di fatto, rispetto al prodotto Proprietario, a parità di qualità intrinseca del prodotto e di TCO[37], il Software Libero, attraverso la disponibilità del codice sorgente e il tipo di licenza, garantisce l'apertura degli standard e dei formati, il dialogo tra sistemi diversi, la loro interoperabilità, una migliore salvaguarda di sicurezza e privacy, l'accessibilità a fornitori in concorrenza e la possibilità di libera scelta da parte dell'utenza (privati, imprese, enti, amministrazioni pubbliche, ecc.) - elementi importanti sia sul breve, sia sul lungo periodo e imprescindibili per alcune categorie di utenti. Non trascurabile, infine, l'aspetto riguardante il Capitale Sociale, specifico del Software Libero, che rende prontamente disponibile uno stock di risorse, efficaci ed efficienti, utili alle strategie per l'azione dei partecipanti al network di relazioni, a tutti i livelli.

5.2 Forma e direzione del cambiamento istituzionale.

Il Software Libero, per le sue caratteristiche istituzionali, descritte nei precedenti capitoli, è più favorevole all'innovazione tecnologica di cui beneficia tutta la società. È da molti punti di vista - politici, economici e sociali - una tecnologia migliore. Ad ogni modo, non basta proporre una tecnologia intrinsecamente superiore[38] e a costo inferiore, per innescare il cambiamento.
Tra gli addetti ai lavori i vantaggi del Software Libero veicolati dal GNU/Linux sono già conosciuti, o facilmente conoscibili, ma già poco al di fuori dell'ambiente informatico e degli ancora pochi utenti diretti, la conoscenza di questo sistema operativo, del Software Libero e del suo modello, non è così scontata: "Linux è entrato in azienda in tempi relativamente recenti, e come spesso accade ci è entrato dalla porta di servizio: tipicamente, se lo sono portato dietro i neoassunti sistemisti che lo usavano con soddisfazione all'università. Lo hanno messo silenziosamente al lavoro per svolgere compiti oscuri (print server, file server, eccetera) o per portare velocemente l'azienda su Internet (sfruttando Apache), facendo leva sulle due caratteristiche più importanti del sistema: è gratis, e funziona. Spesso le macchine Linux venivano messe in linea senza che i direttori dei sistemi informativi ne fossero informati"[39].

Per ridurre la complessità dell'analisi, terrò fermo il punto di vista del prodotto, rivolgendo l'attenzione, come principale riferimento, all'ambiente desktop, vale a dire, quello più coinvolto dalla trasformazione in corso: il mercato dei sistemi operativi desktop - perché su di essi si struttura tutto il resto - e degli applicativi per all'utente medio non informatico.
Il processo di cambiamento, rileva North, è di tipo totalmente incrementale, esso avviene per via delle variazioni dei prezzi relativi o delle preferenze e l'agente del cambiamento è "l'imprenditore individuale sensibile agli incentivi presenti nella struttura istituzionale"[40]. E prosegue: "Il cambiamento consiste in aggiustamenti marginali al complesso di regole, norme e garanzie di applicazione che costituiscono l'ordinamento istituzionale"[41], esso forma il sistema di vincoli composto da regole formali stabili, se il loro

cambiamento ha un costo superiore alla loro conservazione e da vincoli informali, ovvero, abitudini, consuetudini, tradizioni e convinzioni, più tenaci e difficilmente mutabili di quelle formali.
Per comprendere il processo di cambiamento del settore del software, seguendo le teorie di North, occorre riprendere brevemente alcuni 'fatti storici' - già illustrati nei precedenti capitoli - che hanno coinvolto tutto il sistema:
* Il settore desktop, è interessato da una situazione di quasi monopolio della Microsoft con il suo sistema operativo Windows (nelle varie versioni), che per un 'accordo' tra produttori e Microsoft veniva e tuttora viene venduto insieme al computer come fosse una parte solidale con esso (chiedere un computer 'nudo', per non dover pagare un sistema operativo che non interessa, a parte rari casi, porta ad un rifiuto, a complicazioni o all'apertura di un contenzioso, che pochi hanno voglia di affrontare). Microsoft, in questo modo, ha raggiunto una copertura di quasi il 95% del settore[42] e con l'utilizzo di standard chiusi ha praticamente cancellato le possibilità di emersione della concorrenza diretta ai propri prodotti (soprattutto per sistemi operativi e suite da ufficio). Considerando la situazione, produttori di software hanno dedicato i propri sforzi solo su programmi applicativi, non in diretta concorrenza, o utility di sistema per piattaforme Microsoft, alimentando un circolo vizioso che rinforza la sola Microsoft - ovvero, sono in opera i meccanismi di rinforzo di Arthur[43].
* La diffusione di massa dei computer, via via meno costosi, di Internet, sempre più veloce e la creazione di programmi per lo scambio di file in modo anonimo e diretto tra utenti[44] ha aggravato il già diffuso fenomeno della duplicazione abusiva del software e di altri prodotti digitali, quali musica e film e ha scatenato la reazione delle Major dei settori interessati che, alleatesi, hanno chiesto, e spesso ottenuto, leggi anti-pirateria e anti-aggiramento delle protezioni, molto più severe in numerosi stati.
* Nell'ultimo periodo si è cominciato a considerare il sistema operativo GNU/Linux sul versante desktop, che ha raggiunto, nel frattempo una relativa facilità d'uso per i non esperti e dispone di applicativi equivalenti a quelli proprietari per poterli sostituire ed è ottenibile anche gratuitamente, perché la sua duplicazione è consentita dalla licenza d'uso (oltre alle altre libertà che interessano e coinvolgono più direttamente l'utenza esperta).
Sempre seguendo le teorie di North, oltre ai 'fatti storici', dato che "Il cambiamento consiste in aggiustamenti marginali al complesso di regole, norme e garanzie di applicazione che costituiscono l'ordinamento istituzionale" e che questo avviene in corrispondenza del mutamento delle preferenze o dei prezzi relativi si può avvalorare la tesi di un mutamento in corso, osservando entrambi questi elementi.
Con l'emergere del Software Libero sono mutati i prezzi relativi, che hanno alterato gli incentivi individuali all'azione: si può ottenere e manipolare liberamente e gratuitamente; le informazioni che lo riguardano sono più agevoli da recuperare, sono complete e circolano liberamente; è ritenuto da molte fonti autorevoli una tecnologia migliore; modifica il rapporto tra i fattori del lavoro e del capitale; agevola l'acquisizione di sapere e conoscenza (sul prodotto, sul suo utilizzo e sulle possibilità di manipolazione); contribuisce all'abbassamento del suo prezzo cambiando i costi di misurazione (nel modello Proprietario occorre fidarsi solo del dichiarato dal produttore per tutti gli aspetti non direttamente verificabili); infine, consente negoziazioni e contratti con fornitori diversi dall'originale, uscendo dalla costrizione data dal monopolio sui sorgenti, con una evidente redistribuzione del potere contrattuale.
La variazione nelle preferenze e nei gusti è più lenta e meno evidente. Un numero sempre crescente di utenti, comincia a chiedere sistemi più trasparenti, sicuri e rispettosi della libertà e della privacy, domanda più difficile da soddisfare in assenza dei sorgenti. Ma, alcuni fattori quali l'inasprimento delle pene nei confronti di coloro che copiano abusivamente il software, combinato con l'introduzione di procedure macchinose per la verifica della titolarità per effettuare l'installazione (o reinstallazione) del software acquistato regolarmente, insieme a licenze d'uso che, una volta accettate, comportano l'assegnazione del permesso al produttore, di verificare la titolarità del possesso (e di blocco in caso di abuso) e la possibilità di installare 'aggiornamenti'[45] anche da remoto e senza preventivo permesso, possono far optare per sistemi e programmi Free equivalenti purché sia conosciuta questa possibilità.

Per comprendere il processo in atto, occorre ora rilevare, quali strategie d'azione, all'interno di uno scenario così definito, sono state portate avanti dagli imprenditori individuali[46], quali motori del cambiamento.
Gli elementi intorno ai quali si articolano le strategie degli attori, sia dal lato Proprietario, sia da quello del Software Libero, sono l'informazione e l'azione sul 'mercato politico'.
Tra i due schieramenti[47] c'è una vera e propria lotta attraverso l'informazione, che prevede alternanza di notizie, vere e false, e controinformazione.
Per gli agenti dal lato del Software Libero, l'attività di informazione sulle caratteristiche, peculiarità, metodi, ecc., all'esterno dell'ambiente degli addetti ai lavori trova spesso ostacoli dovuti a pregiudizi, non più sostenibili, formatisi nel periodo di immaturità del sistema (difficoltà di utilizzo, assenza di applicativi,

incompatibilità, ecc. ora quasi del tutto superate), dovuti a superficialità di alcuni media, cui si aggiunge però, una vera e propria campagna di disinformazione, attraverso la cosiddetta FUD (Fear, Uncertainty, and Doubt), attuata deliberatamente dalla Microsoft[48], cui si aggiungono strategie da camaleonte che si basano su un'imitazione di facciata degli aspetti vantaggiosi e più appariscenti del Software Libero per contrastarne l'avanzata mostrando ad osservatori superficiali caratteristiche 'equivalenti'[49].
Nonostante questa schiacciante asimmetria nella forza contrattuale a favore di Microsoft, il sistema operativo GNU/Linux continua ad avanzare (favorito solo in parte dalla congiuntura economica negativa, che induce alla ricerca di soluzioni meno costose e vincolanti) e man mano che aumentano i suoi utenti, dai singoli agli stati, sono essi stessi a smentire le argomentazioni sfavorevoli opponendo argomentazioni contrarie e realizzazioni concrete.
Lo stato del sistema istituzionale consente, praticamente ovunque, di esprimere le proprie opinioni a costo relativamente basso[50]. Ma solo in ambito Free idee ed ideali basati sulla condivisione delle conoscenze, sulla collaborazione e su una giusta concorrenza - appoggiati da una moltitudine di programmatori sparsi in tutto il mondo - sono sostenuti con impegno e sostanziale sacrificio.
È pur vero che le strategie FUD, incluse quelle che tentano di dare un colore politico o una valenza estremista e sovversiva al movimento, hanno innalzato il prezzo dell'espressione delle idee del Software Libero ma si sono anche rivelate un boomerang, ogniqualvolta è emersa la verità e il buonsenso, con smentite provenienti da fonti autorevoli[51], in costante crescita.
A questo punto entra in gioco il mercato politico. Per North, infatti, il processo di cambiamento[52], può essere così descritto: "Una modifica dei prezzi relativi spinge una o entrambe le parti allo scambio, sia politico che economico, perché si rendono conto dei miglioramenti ottenibili con un accordo o contratto diversi. È in atto un tentativo di rinegoziazione, tuttavia data la collocazione del contratto in un sistema gerarchico di regole, questa può non essere possibile senza riformulare un insieme di regole superiore (o violando una qualche norma di comportamento). In questo caso, la parte interessata a migliorare la propria posizione contrattuale può effettivamente essere indotta a impegnare ulteriori risorse nella modifica, delle regole al livello più elevato. Quando si tratta di norme di comportamento, una variazione dei prezzi relativi o un mutamento di gusti genera una loro graduale trasformazione e una conseguente sostituzione a opera di altre norme. Nel tempo una regola formale può essere cambiata o semplicemente accantonata e non resa esecutiva; allo stesso modo una consuetudine o una tradizione può essere gradualmente abbandonata e rimpiazzata"[53].
Come poi indica North[54], è possibile tracciare un quadro della situazione introducendo in questa descrizione, l'attività degli attori principali - singole persone e organizzazioni direttamente coinvolte - quali agenti del cambiamento.
Nel quadro istituzionale del mercato del software, si trovano, innanzitutto, le regole formali esecutive (applicate ed efficaci), che, uguali per tutti, restano sullo sfondo; trattandosi di un fenomeno presente in ogni punto del globo, l'uguaglianza va intesa per aree geografiche/giuridiche omogenee; infatti la legislazione, il modo di applicare e far rispettare le leggi cambia se ci troviamo negli Stati Uniti, in Europa o negli altri continenti e spesso per i gradi inferiori cambia da stato a stato. In secondo luogo, ci sono le regole informali, del modello Proprietario e di quello del Software Libero, che marciano parallele e rimangono anch'esse ferme sullo sfondo. A queste vanno aggiunte le consuetudini e usanze degli utenti. Infine, troviamo quelle regole formali che combinate con quelle informali, sono applicate male o in modo distorto, risultano inefficaci per lo scopo previsto oppure non sono applicate perché il costo sociale per farle rispettare supererebbe di molto i benefici.
Ma, la modifica dei prezzi relativi avvenuta con l'emergere del Software Libero, anche commerciale, cui ho accennato poc'anzi, è maturata all'interno del sistema istituzionale fin qui tracciato. Oggi le parti, o tornando alla metafora del gioco, le squadre del Software Libero, di quello Proprietario e degli Utenti, vorrebbero cambiarlo; vorrebbero modificare le regole del gioco, riformulando l'insieme di istituzioni formali, per migliorare o mantenere la propria posizione, agendo su quelle specifiche normative[55], in grado di mutare l'assetto attuale, da sole o combinate con le rispettive istituzioni informali.
Tra queste troviamo le norme anti-pirateria, (a protezione dalle violazioni del diritto d'autore, del segreto industriale e dei brevetti), quelle antitrust e quelle sui brevetti, tutte interessate nei loro diversi gradi (dai regolamenti locali agli statuti internazionali) da pressioni contrapposte, per la loro corretta e pronta applicazione o integrazione o modifica.
La ricerca di nuovi accordi e contratti di tipo politico, o di aiuto presso gli organi di garanzia, spendendo risorse, economiche e/o umane, è l'attualità di questi giorni.
A questo punto, però, la divisione non è più tra software proprietario e Libero ma tra monopolisti e tutti gli altri, quindi, imprese e programmatori sia Liberi sia Proprietari e in parte anche gli utenti. I primi per conservare la propria posizione di monopolio, gli altri perché si venga a formare un riequilibrio nella forza contrattuale di tutte le parti ed un ripristino pieno della concorrenza.

Le normative anti-pirateria, soprattutto negli Stati Uniti con la normativa antiaggiramento del '98, il Digital Millennium Copyright Act, hanno provocato una serie di controversi casi legali, avutisi dopo la sua introduzione, che fanno pensare ad un uso come strumento strategico per bloccare la concorrenza invece che come valido aiuto per bloccare la pirateria: "In pratica, le norme anti-aggiramento sono state usate per limitare un'ampia gamma di attività legittime, anziché per bloccare la pirateria. Come risultato, il Digital Millennium Copyright Act (DMCA) si è tramutato in una seria minaccia a tre importanti priorità della politica pubblica: la libertà d'espressione e la ricerca scientifica, gli usi consentiti, la competizione e l'innovazione"[56].
In Europa, dove si sta discutendo se adottare una legislazione simile, l'EUCD (European Union Copyright Directive), la parte non monopolista, con prevalenza di 'attivisti' nei sostenitori del Software Libero, sta spendendo le proprie risorse per denunciare le conseguenze ed i pericoli legati ad una sua integrazione nelle legislazioni degli Stati dell'Unione Europea, per contrastare l'attività di lobbying dei monopolisti, in modo che i legislatori abbiano un adeguato feedback, sui possibili abusi e lesioni alle libertà di utenti, programmatori, ricercatori in generale e non circoscritta al solo mercato del software, che questa normativa lascerebbe entrare[57].
Riguardo le norme antitrust, i danneggiati da situazioni di monopolio fanno sentire sempre di più la propria voce e cominciano ad ottenere qualche risultato sia negli USA sia in Europa[58].
Molti stati[59], inoltre, stanno introducendo normative che, formulate nell'interesse generale, avvantaggiano in particolare il Software Libero proprio per le sue peculiari caratteristiche.
Il problema che rimane sullo sfondo è quello della lentezza del processo politico volto a riequilibrare il peso delle parti in un mercato concorrenziale e della giustizia che censuri l'abuso di posizione dominante, in un mercato, per dirla alla Gates, del 'Business alla velocità del pensiero'[60].
Infine c'è il 'fronte' dei brevetti. Il software non sarebbe brevettabile ma di fatto, con un espediente contrario allo spirito e alla lettera delle leggi in materia, viene brevettato sia in America sia in Europa. Ma la pratica legale, è differente nei due continenti, e nel sistema americano ha un peso normativo rilevante, per cui solo nel nuovo continente, decidere di programmare in un certo modo, magari inserendo un algoritmo [61] preso da un periodico specializzato, fa correre ogni volta il rischio di essere citati in giudizio, "oggi la situazione è proprio questa, e la ragione sono i brevetti sul software. Presto potrebbe essere lo stesso nella maggior parte dell'Europa. I paesi che gestiscono l'Ufficio Europeo dei Brevetti, spronati dalle grandi compagnie e incoraggiati dagli avvocati specializzati in brevetti, si stanno muovendo per consentire ai brevetti di coprire i calcoli matematici"[62].
Chi produce nel campo tecnologico negli Stati Uniti, rischia di subire una causa di risarcimento in ogni momento, per il solo fatto di aver realizzato qualcosa. Non è un caso che in questo settore la piccola e media impresa in America quasi non esista.
La spinta da parte delle multinazionali perché vengano riconosciuti ed istituzionalizzati maggiori vincoli sull'utilizzabilità delle conoscenze, a parte le implicazioni morali ed etiche - cioè, fino a che punto è lecito considerare come proprie, appropriabili e privatizzabili, le idee, che per quanto innovative esse siano non nascono nel vuoto, ma si fondano su idee altrui - andrebbe contro la ragione di utilità sociale per cui il brevetto è stato creato e che non è neppur certo dia reali benefici allo stato attuale: "Gli economisti hanno esaminato l'economia del sistema dei brevetti in generale per anni, ma non sono mai stati in grado di dimostrare in modo inconfutabile che i vantaggi di un regime brevettuale siano superiori agli svantaggi.
Persino la tesi di base, secondo cui i brevetti promuovono l'innovazione in misura tale da giustificare i costi e i vantaggi compensano gli inconvenienti, è discutibile: secondo alcuni studi, potrebbe essere altrettanto vero il contrario"[63].
Senza dubbio, all'interno del sistema istituzionale del mercato del software, sarebbe uno strumento strategico di enorme portata, per fermare l'avanzata del Software Libero, ma anche del software scritto dalle piccole e medie aziende Proprietarie, che in Europa sono numericamente consistenti e comprensibilmente contrarie all'introduzione del brevetto in quest'ambito.

Se il modello del Software Libero riuscirà a contrastare lo sviluppo del mercato del software, che negli ultimi 20 anni, si è portato verso il modello Proprietario, e di monopolio dei sistemi operativi della Microsoft [64], in futuro da fenomeno marginale potrà raggiungere una posizione più centrale. L'esito di questo scontro, per nulla scontato, darà il sentiero di sviluppo del mercato del software di domani.
L'esito delle modifiche al quadro istituzionale, potrebbe influenzare anche il campo della reciprocità; considerando che si tratta sostanzialmente dello stesso quadro su cui poggia la ricerca scientifica, le future scelte, potranno garantire una situazione di scambi in un contesto di cambiamento dinamico ed evolutivo, oppure, andranno a creare situazioni di scambio irrigidite, introdotte da regole formali tali da impedire produzione e distribuzione del software come avvenuto fino ad oggi, e non solo, dato che si tratta di restrizioni che danneggerebbero la ricerca scientifica nel suo complesso.

Dall'esito del processo politico attualmente in corso (che però varia da paese a paese!) e dalla forza che le parti riusciranno ad esercitare, dipende la direzione del processo e il suo consolidamento su un sentiero evolutivo oppure no.

1 North, 1990, pag. 25.
2 Ibidem.
3 Cella, 1997; Polanyi, 1944.
4 Per scambio di mercato si intende la forma di regolazione dell'economia che implica lo scambio attraverso il commercio all'interno dei mercati in cui si forma il prezzo in base all'incontro tra domanda e offerta. Per reciprocità, invece, si intende la forma di regolazione dell'economia dove beni e servizi vengono prodotti e scambiati in base alle aspettative di ricevere altri beni e servizi secondo modalità e tempi fissati da norme sociali condivise.
5 Fondamentale l'introduzione del sistema grafico a finestre e icone.
6 Nella maggior parte dei comparti è in posizione marginale. Sicuramente centrale per il settore server e per ciò che è strettamente legato alla Rete.
7 Qui inteso come modello puro. Alcune imprese pur seguendo il modello Open Source distribuiscono i propri prodotti parzialmente chiusi, per riprodurre la dipendenza tipica dell'ambiente Proprietario.
8 Arthur, 1988, pag. 10; North, 1990, pag. 137.
9 Istituzionali e tradizionali della teoria economica. Nel caso specifico sono rilevanti quelli tecnologici.
10 North, 1990, pag. 27.
11 Coase, 1960.
12 Quello istituzionale è un processo circolare, pertanto di seguito, a differenza dei capitoli precedenti in cui ho isolato le varie istituzioni informali e formali, cercando di illustrarne l'evoluzione, ora indicherò il ruolo delle istituzioni esistenti sui costi di transazione e sugli altri elementi anzidetti per i due modelli qui considerati, quindi il senso causale risulta invertito.
13 North, 1990, pagg. 33-40.
14 Coase, 1960, cit. in North, 1990, pag. 39.
15 North, 1990, pag. 40.
16 Questo stato di cose, come accennato in precedenza, ha portato alla distorsione del mercato che si è allontanato dalla situazione ideale di concorrenza per via dell'affermazione di un grande produttore, Microsoft, in posizione di quasi monopolio, che ha conquistato la quasi totalità del mercato dei sistemi operativi per desktop, delle suite da ufficio ed una buona porzione degli altri comparti.
17 Cit.
18 http://www.gnu.org/philosophy/free-sw.it.html
19 Nel mondo UNIX i conteggi cominciano dallo zero.
20 Non tutto il Software Libero è ospitato presso associazioni. Per il progetto GNU si veda: http://www.gnu.org/licenses/gpl-violation.html
21 Non in tutti i casi conviene il Software Libero. Ad esempio, non conviene assolutamente se il software è abbinato ad altri prodotti e incide sui costi totali solo in piccola parte, come ad esempio nei macchinari industriali con funzioni automatizzate attraverso l'uso di software dove non è il caso di rivelare alla concorrenza un elemento fondamentale che fornisce all'azienda il vantaggio competitivo e dove la disponibilità di software libero non incide sulla soglia d'ingresso.
22 Bob Young e Marc Ewing hanno fondato la Red Hat (distribuzione Linux diretta concorrente di Microsift per tipologia di prodotti) nei primi anni '90. La prima sede fu la lavanderia di casa. Nel '99 Red Hat è entrata in Borsa realizzando uno dei principali exploit del Nasdaq. Oggi, 2002, ha centinaia di dipendenti e uffici di rappresentanza sparsi in tutto il mondo e nonostante le difficoltà del 'nuovo mercato' è in attivo.
23 Torno comunque a sottolineare che il Software Libero commerciale è una forma intermedia e se ne deve tenere conto in sede di pianificazione delle strutture interne aziendali e delle strategie di mercato.
24 Coleman, 1990. Concetto introdotto nel par. 2.5 'Cultura hacker e Capitale Sociale'.
25 Occorre ricordare che il Capitale Sociale ha natura indeterminata per cui in alcuni casi può generare fiducia e informazioni che aiutano la cooperazione e lo sviluppo, ma, in altre situazioni, può generare reti che sul mercato servono agli appartenenti ad eludere la concorrenza a spese dei consumatori o delle organizzazioni economiche attraverso la collusione.
26 Commercializzato con le modalità già esposte in precedenza.
27 Ma la mediana è 2 e la moda è 1. La maggior parte dei progetti riguarda programmi di piccole dimensioni. FLOSS, cit.
28 Ad esempio in una lista di discussione pubblica le risposte arrivano tendenzialmente dai soggetti più attivi e competenti e comunque chiunque risponda lo fa sotto il 'controllo' degli altri partecipanti che

interverranno per integrare informazioni incomplete o per riprendere quelle imprecise o errate. L'invio reiterato di risposte contenenti errori ed imprecisioni provocano rapidamente l'isolamento del suo autore.
29 Il quasi monopolio di Microsoft per i sistemi operativi desktop raggiunto, con regolari strategie, ma, anche con modalità illecite come, ad esempio, gli accordi capestro con i produttori e distributori di computer per la vendita abbinata hardware-sistema operativo, tollerati o difficilmente contrastabili per l'intempestività o indifferenza dei sistemi giudiziari (la vita media di un programma software è più breve del tempo di attesa per una sentenza) e con la sostanziale tolleranza nei confronti della riproduzione illegale del software usata come strategia di penetrazione nel mercato.
30 Questo fatto è compreso e accettato nel versante della reciprocità dove è più importante un programma sicuro e stabile piuttosto che arrivare prima degli altri - essenziale sul mercato - con qualche difetto celato ad arte e sistemato in un secondo momento.
31 Cfr. http://www.it.debian.org/partners/
32 Mentre Red Hat, prima distribuzione nata nel '94 in America e prima in quel mercato, regge bene e, nonostante la congiuntura negativa, produce utili, la MandrakeSoft, francese, e meglio posizionata sul mercato europeo è attualmente (2002-2003) in crisi. Eventuali nuovi competitori, dovrebbero fare cospicui investimenti, nell'incertezza assoluta del risultato, per posizionarsi in modo analogo.
33 Coerentemente alle regole della comunità che disapprova la duplicazione degli sforzi e lo spreco di risorse che essa causa.
34 Non specializzate per nicchie particolari.
35 The Linux Documentation Project: http://tldp.org/HOWTO/CD-Distributions-EN-HOWTO/
Trad. it.: http://ildp.pluto.linux.it/HOWTO/Distributions/en-distributions.html
36 Non sarebbe accettato un passaggio così radicale da coloro che fino a poco prima pagavano per le licenze d'uso.
37 Total Cost of Ownership: costo totale di un computer o di un altro dispositivo. Oltre all'acquisto, si calcola anche l'installazione, i dispositivi accessori ed indispensabili, la manutenzione, i ricambi programmati ed ogni altro costo necessario al funzionamento corretto sul medio/lungo periodo.
38 Per ammissione diretta ed indiretta della stessa Microsoft, si vedano a proposito gli Halloween Documents (http://www.opensource.org/halloween/) e il nuovo Shared Source Licensing Programs (http://www.microsoft.com/licensing/sharedsource/) con cui cerca di imitare alcuni elementi dell'Open Source.
39 'Attacco al pinguino' di Silvano Corridolo, Network News 5 ottobre 2002. Disponibile online c/o: www.vnunet.it/downloads/20021023003_01_03/ primopiano.pdf
40 North, 1990, pag. 123.
41 Ibid.
42 Anche con modalità ritenute illecite con cui si è arrivati a questa situazione di monopolio. Per un approfondimento si vedano, ad esempio, gli atti dei processi antitrust contro Microsoft disponibili online: http://www.microsoft.com/presspass/trial/archive.asp
43 Arthur, 1988; qui par. 4.2.
44 I programmi "peer-to-peer" hanno facilitato enormemente lo scambio di file di ogni genere tra utenti 'anonimi', inclusi programmi già crackati senza dover prendere contatto con altre persone e senza formalità (ad esempio trovare un gruppo di scambio file via ftp e diventarne membro)
45 Windows XP.
46 Intesi come singoli e come dirigenti delle organizzazioni, non solo di tipo economico.
47 Più un 'Microsoft contro tutti' considerando la sua posizione.
48 Cfr. Halloween Documents (http://www.opensource.org/halloween/).
49 Alla mancanza di trasparenza e collaborazione contrappongono le licenze Shared Source (disponibile online alla pagina: http://www.microsoft.com/licensing/sharedsource/) e all'eccessivo costo delle licenze contro la gratuità di quelle libere, risponde omaggiando molti stati con computer e software Microsoft (destinati alle scuole e agli enti pubblici) e negoziando trattamenti più economici per i contratti futuri al primo sentore di tendenza verso la concorrenza.
50 Lo sfondo è quello dei paesi industrializzati.
51 Un esempio alla portata del grande pubblico, può essere la campagna pubblicitaria 2002 dell'IBM ai server Linux.
52 In una versione della teoria utile per comprenderne il processo, ma, per forza di cose semplificata.
53 North, 1990, pag. 127.
54 North, 1990, pag. 128.
55 Cfr. capitolo 3.
56 Bernardo Parrella, 'Tutte da rifare le norme sul copyright digitale': (http://www.apogeonline.com/webzine/2003/01/14/13/200301141301) che riporta un comunicato diffuso

dalla Electronic Frontier Foundation sulla legislazione statunitense che dal 1998 regola il copyright digitale (http://www.eff.org/IP/DMCA/20030102_dmca_unintended_consequences.html).

57 Qualche esempio:
Campagna di sensibilizzazione sui pericoli della EUCD (European Union Copyright Directive) promossa dall'Associazione Software Libero. http://www.softwarelibero.it/progetti/eucd/index.shtml
Una lettera della Società degli archivisti britannica, contenente una critica all'EUCD: http://web.archive.org/web/20010223233342/http://www.pro.gov.uk/about/copyright/copyrightdraft.htm (in inglese).
La petizione che mira ad escludere le pubblicazioni scientifiche dall'ambito dell'EUCD è annunciata e motivata da un articolo di Paul Caro, per l'Accademia delle scienze francese: "Si deve applicare alle pubblicazioni scientifiche la Direttiva europea 2001/29/CE del 22 maggio 2001 sul copyright?" http://www-mathdoc.ujf-grenoble.fr/DA/ (in francese).

58 Qualche esempio:
Da Repubblica.it 'Microsoft e Antitrust si muove anche l'Europa' http://www.repubblica.it/online/tecnologie_internet/bill/antitrust/antitrust.html
Esito della causa intentata nel '99 dai consumatori californiani contro Microsoft raccontata da Paolo Attivissimo nell'articolo 'California, quantificato il prezzo del monopolio Microsoft' (http://www.apogeonline.com/webzine/2003/01/14/01/200301140101)

59 Brasile, Norvegia, Germania, Gran Bretagna, Francia, Finlandia, Messico, Corea, Tailandia, Filippine, Taiwan, Cina e Giappone. A queste si sarebbero aggiunte anche Perù ed India ma hanno rinunciato all'adozione dopo l'intervento diretto di Bill Gates (che ha fatto visita ai rispettivi governi) e in cambio di cospicui investimenti. In Italia è stato presentato un progetto di legge a favore del Software Libero, è attiva una commissione composta da membri di Camera e Senato di entrambe le coalizioni ed stata istituita una commissione di studio per l'introduzione del Software Libero nella Pubblica Amministrazione. Il 31 gennaio 2003 Bill Gates è stato in visita nel nostro paese...

60 Dal titolo di un libro del '99 scritto da Bill Gates, edito in traduzione italiana da Mondadori.

61 L'algoritmo e' una serie di istruzioni ben definita che ha lo scopo di risolvere un problema. [In sostanza è una formula matematica.] Un classico esempio di algoritmo e' quello matematico del calcolo della radice quadrata o quello finanziario del calcolo di un tasso di interesse (glossario online di PCPratico: http://www.pcpratico.it/).

62 Tratto da 'Salvare l'Europa dai brevetti sul software' della Free Software Foundation http://www.gnu.org/philosophy/savingeurope.it.html.

63 Parlamento Europeo; documento di lavoro della Direzione generale degli Studi, 'La brevettabilità dei programmi per elaboratore. Analisi della legislazione a livello europeo nel campo dei brevetti per il software', di Reinier Bakels e P. Bernt Hugenholtz.
Disponibile online all'indirizzo: http://www.europarl.eu.int/hearings/20021107/juri/study_it.pdf.

64 Il problema non è il fatto in sè, ma, che alla situazione di monopolio associa standard chiusi che escludono gli altri produttori dal mercato e impediscono l'interoperabilità dei sistemi.

Bibliografia:

AAVV, (New Logic & Redazione Scientifica Rizzoli).
1987 Il mondo del computer, Milano, RCS Rizzoli.

Arthur, W.B.
1988 "Self-Reinforcing Mechanisms in Economics", in The Economy as an Evolving Complex System, a cura di P.W. Anderson, K.J. Arrow e D. Pines, , Addison-Wesley, Mass.

Axelrod , R.
1997 The Evolution of Cooperation, New York, Basic Books; trad. it. Giochi di Reciprocità, Milano, Feltrinelli, 1985.

Becker, G.S.
1964 Human Capital, New York, Columbia University Press.

Berra, A., Meo A.R.
2001 Informatica solidale, Torino, Bollati Boringhieri.

Carlini, F.
2002 Divergenze digitali, Roma, Manifestolibri.

Cella, G.P.
1997 Le tre forme dello scambio, Bologna, il Mulino.

Coleman, J.S.
1990 Foundations of Social Theory, Cambridge, The Belknap Press of Harvard University Press.

Coase, R.H.
1960 "The Problem of Social Cost", in "Journal of Law and Economics", 3, pagg. 1-44.

DiBona, C., Ockman, S., Stone, M. (a cura di)
1999 Open Sources. Voices from The Open Source Revolution, Sebastopol, O'Reilly & Associates; trad. it. Open Sources. Voci dalla rivoluzione Open Source, Milano, Apogeo, 1999.
Disponibile come documento elettronico nello spazio Web di Open Press all'indirizzo:
http://www.apogeonline.com/openpress/libri/545/index.html

Gambaro, M., Ricciardi, C.A.,
1997 Economia dell'informazione e della comunicazione, Roma-Bari, Laterza.

Godbout, J.T.
1992 L'Esprit du don, Paris, La Découverte; trad. it. Lo spirito del dono, Torino, Bollati Boringhieri, 1993
Gubitosa, C.
1999 Italian Crackdown, Milano, Apogeo.

Hafner, K., Lion, M.,
1996 Where wizards stay up late. The origins of the Internet; trad. it. La storia del futuro. Le origini di Internet., Milano, Feltrinelli, 1998.

Hersey, P., e Blanchard, K.,
1982 Management of Organizational Behavior, Englewood Cliffs, NJ, Prentice Hall,; trad. it. Leadership situazionale, Sperling & Kupfer, 1984.

Himanen, P.,
2001 The Hacker ethic and the spirit of information age, Random House; trad. it. L'etica hacker e lo spirito dell'età dell'informazione, Milano, Feltrinelli, 2001.

Kuhn, T.S.

1962 The Structure of Scientific Revolutions, The University of Chicago; trad. it. La struttura delle rivoluzioni scientifiche, Torino, Einaudi, 1969.

Levy, S.
1984 Hackers. Heroes of the computer revolution, Garden City, Doubleday; trad. it. Hackers. Gli eroi della rivoluzione informatica, Milano, Shake, 3° ed. 1999.

Lévy. P.
1995 Qu'est-ce que le virtuel?, Paris, La Découverte; trad it. Il virtuale, Milano, Raffaello Cortina, 1997.

Mauss, M.
1950 Sociologie et anthropologie, Paris, Presses Universitaires de France; trad. it. Teoria generale della magia e altri saggi, Torino, Einaudi, 1965.

Melucci, A.
1991 L'invenzione del presente. Movimenti sociali nelle società complesse, nuova edizione, Bologna, il Mulino.

Merton, R.K.
1959 Social Theory and Social Structure, Glencoe, Free Press; trad. it. Teoria e struttura sociale, Bologna, Il Mulino, 1992
1973 The Sociology of Science: Theoretical and Empirical Investigations, The University of Chicago Press, Chicago, Usa,; trad. it. La sociologia della scienza, Milano, Franco Angeli Editore, 1981.

Moody, G.
2001 Rebel Code. Linux and the Open Source Revolution, London, Allen Lane The Penguin Press.

North, D.C.
1990 Institutions, Institutional Change and Economic Performance, Cambridge, Cambridge University Press; trad. it. Istituzioni, cambiamento istituzionale, evoluzione dell'economia, Bologna, Il Mulino, 1994.

Olson, M.
1965 The logic of Collective Action, Cambridge, Harvard University Press; trad. it. La logica dell'azione collettiva, Milano, Feltrinelli, 1983.

O'Reilly, T.
1999 "Dieci miti del software Open Source", L'articolo è la trascrizione di un seminario tenuto il 14/12/1999 da Tim O'Reilly ad un gruppo di dirigenti di Fortune 500 nell'ottobre del '99, documento elettronico disponibile in trad. it. all'indirizzo:
http://www.hops.it/myths.html

Paccagnella, L.
2000 La comunicazione al computer, Bologna, il Mulino.

Pizzorno, A.
1999 "Perché si paga il benzinaio. Nota per una teoria del capitale sociale", in Stato e Mercato n° 57, dicembre, pp. 373-394.

Polanyi, K.
1944 The Great Transformation; trad. it. La grande trasformazione, Torino, Einaudi, 1974.

Raymond, E.S.
1975 prima versione, 1., (iniziata da Raphael Finkel a Stanford; documento di pubblico dominio) qui versione 4.3.1, 29 JUN 2001 attualmente mantenuta da Raymond
"The on-line hacker Jargon File"[Jargon file] documento elettronico disponibile all'indirizzo:
http://www.tuxedo.org/~esr/jargon/jargon.html#
1996 prima versione, 1.1; qui versione 1.92, 2000
"How To Become A Hacker" documento elettronico disponibile all'indirizzo:
http://www.tuxedo.org/~esr/faqs/hacker-howto.html#WHY_THIS
1997 prima versione, 1.16; qui versione 1.42, 1999

"The Cathedral and tha Bazaar"[CatB], documento elettronico disponibile all'indirizzo:
http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar.html
trad. it. "La cattedrale e il bazaar", documento elettronico disponibile all'indirizzo:
http://www.apogeonline.com/openpress/doc/cathedral.html
1998 prima versione, 1.2; qui versione 1.14, 1998
"Homesteading the Noosphere"[HtN], documento elettronico disponibile all'indirizzo:
http://www.tuxedo.org/~esr/writings/homesteading/homesteading.html
trad. it. "Colonizzare la Noosfera", documento elettronico disponibile all'indirizzo:
http://www.apogeonline.com/openpress/doc/homesteading.html
1999 prima versione 1.5; qui versione 1.15, 1999
"The Magic Cauldron", documento elettronico disponibile all'indirizzo:
http://www.tuxedo.org/~esr/writings/magic-cauldron/magic-cauldron.html
trad. it. "Il calderone magico", documento elettronico disponibile all'indirizzo:
http://www.apogeonline.com/openpress/doc/calderone.html

1999b The Cathedral & the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary, Sebastopol, O'Reilly & Associates.
Comprende Raymond 1996; 1997; 1998; 1999, 1.17.

Scelsi, R.V, (a cura di)
1994 No Copyright. Nuovi diritti del 2000, Milano, Shake.

Torvalds, L., Diamond, D.
2001 Just for Fun, by Linus Torvalds and David Diamond; trad. it. Rivoluzionario per caso. Come ho creato Linux (solo per divertirmi), Milano, Garzanti, 2001.

Weber, M.
1919 Politik als Beruf, Wissenschaft als Beruf, Berlin, Duncker & Humblot; trad. it. Il lavoro intellettuale come professione, Torino, Einaudi, 1948.

Williamson, O. E.
1985 The economic Institutions of Capitalism, New York, Free Press; trad. it. Le istituzioni economiche del capitalismo. Imprese, mercati, rapporti contrattuali, Milano, Angeli, 1988.

SITOGRAFIA:

Apache Software Foundation
http://www.apache.org/

Apogeonline OpenPress
Casa editrice Apogeo: libri e notizie dedicate all'Open Source
http://www.apogeonline.com/openpress/index.html

AsSoLi - Associazione Software Libero
http://www.softwarelibero.org/

CollabNet
provides platforms and services based on open source tools and principles
http://www.collab.net

The Computer Museum History Center
http://www.computerhistory.org/

CoSource
a collaborative, reverse-auction web site
http://www.cosource.com/
Digital Millennium Copyright Act - Overview c/o:
http://www.gseis.ucla.edu/iclp/dmca1.htm

EFF Homepage - the Electronic Frontier Foundation
http://www.eff.org/

Eurolinux Alliance
http://www.eurolinux.org/

FLOSS 2002
Indagine sui Software Developers del Free ed Open Source Software, con supporto della Commissione Europea, Programma IST. Studio dell'Institute of Infonomics, Universita' di Maastricht e Berlecon Research, Berlino.
Risultati disponibili all'indirizzo:
http://www.infonomics.nl/FLOSS/report/

FLOSS-US 2003
Indagine condotta da ricercatori dello Stanford Institute for Economic Policy Research (SIEPR, Universita' di Stanford). È parte di una ricerca intitolata: 'Organizzazione economica e fattibilita' dell'open source software', sotto l'egida del programma SIEPR: Knowledge, Networks and Information for Innovation Program (KNIIP). La ricerca e' finanziata dalla National Science Foundation. Il questionario e' stato sviluppato in collaborazione con Rishab Aiyer Ghosh (MERIT e Infonomics, University of Maastricht) che ha coordinato il questionario FLOSS della comunita' di programmatori open source/free software (OS/FS), finanziato dalla Commissione Europea nell'anno 2002 [http://www.infonomics.nl/FLOSS/report/].
http://www.stanford.edu/group/floss-us/

Fetchmail
Utility per "scaricare la posta" da server di posta remoti e renderla disponibile in locale:
http://www.tuxedo.org/~esr/fetchmail/

Free patents
http://www.freepatents.org

Free Software Foundation (v. progetto GNU)
http://www.fsf.org/

Free Software Foundation Europe
La FSF Europe è stata avviata il 10 Marzo 2001 e opera su tutti gli aspetti Europei del Software Libero e in particolar modo del Progetto GNU. Supporta attivamente lo sviluppo di Software Libero e i sistemi basati su GNU come GNU/Linux. Inoltre fornisce un centro di competenze di politici, avvocati e giornalisti in modo da assicurare un futuro legale, politico e sociale al Software Libero.
http://www.fsfeurope.org/

Freshmeat
http://freshmeat.net/

Gnome project
http://www.gnome.org

GNU: Progetto GNU della Free Software Foundation
Il sito ufficiale progetto GNU (GNU's Not Unix!) creato da Richard Stallman:
http://www.gnu.org
Mirror delle pagine anche all'indirizzo della Free Software Foundation:
http://www.fsf.org

Halloween Documents I, II, III (ita.)
documenti interni della Microsoft (pubblicamente riconosciuti autentici) commentati da E.S. Raymond. Al 2 gennaio 2003 ne sono stati prodotti 8.
http://www.opensource.org/halloween/
in trad.it.:
http://pkg.lugbs.linux.it/PACCHETTI/Documenti/CONTENTS/varie/halloween/halloween1it.html
http://pkg.lugbs.linux.it/PACCHETTI/Documenti/CONTENTS/varie/halloween/halloween2it.html

http://pkg.lugbs.linux.it/PACCHETTI/Documenti/CONTENTS/varie/halloween/halloween3it.html

Hopslibri
Casa editrice Hops: opensource, internet, libri
http://www.hopslibri.com/

Interlex
Informazioni giuridiche che riguardano la rete.
http://www.interlex.it/

K Desktop Environment
http://www.kde.org/

Linux Home Page at Linux Online
http://www.linux.org/

Linuxcare: Laboratori Prosa di Linuxcare Italia.
Laboratorio che si occupa di supporto, ricerca, sviluppo e di formazione. Per statuto utilizzano esclusivamente software libero avvalendosi degli strumenti GNU e di Linux.
http://www.prosa.it/index.it.html
http://www.linuxcare.com/ (ing.)
http://www.linuxcare.it/

Linux Magazine - Chronicle of the Revolution
http://www.linux-mag.com/

Mozilla
http://www.mozilla.org/

No Patents
Contro i brevetti del software e le nuove forme di monopolio accettate dagli uffici brevetti.
http://no-patents.prosa.it/

OpenLabs - Risorse per gli utilizzatori di Linux e tecnologie aperte.
Associazione culturale per la diffusione e l'esercizio cosciente della libertà sia riguardo alle scienze informatiche e telematiche che riguardo all'uso delle relative tecnologie nella società civile.
http://www.openlabs.it/

Open Source Development Network (OSDN)
http://www.osdn.com/

Open Source Initiative
Il sito ufficiale della Open Source Initiative guidata da Eric S. Raymond:
http://www.opensource.org

Perl
http://www.perl.org

Pluto Linux User Group
La più importante comunità italiana di appassionati di Linux (LUG). Documentazione in italiano e il Pluto Journal:
http://www.pluto.linux.it

Slashdot
http://slashdot.org/

SourceForge
Sito dedicato allo sviluppo Open Source, ospita 'il più grande archivio' di codice Open Source e applicazioni disponibili su Internet. Fornisce servizi gratuiti e liberi agli sviluppatori Open Source. http://sourceforge.net/

Glossario

Algoritmo: Procedura costituita da una successione finita di operazioni aritmetiche e/o logiche. Un algoritmo viene progettato per assolvere un determinato compito (nell'informatica musicale, ad esempio, per sintetizzare o elaborare un suono), e tradotto in un linguaggio di programmazione; "è una descrizione formale di come ottenere un risultato a partire da certi materiali di partenza. Non per caso viene sovente utilizzata la metafora delle ricette di cucina, che non è perfettamente aderente, ma rende l'idea. La ricetta del pesto alle genovese è la descrizione precisa di ingredienti (sale, olio, basilico, formaggio grana e pecorino, qualche volta noci e pinoli) e della sequenza delle operazioni da compiere su tali input per ottenere in output una morbida salsa. Un algoritmo matematico piuttosto semplice è quello che dice come scomporre un numero in fattori primi: si parte dai numeri primi più bassi e si verifica per quali esso sia divisibile con resto intero, cosi appurando, per esempio, che 42 è fatto di 2x3x7".[1]

Apache: Server web per Linux (e ora anche per UNIX e Windows). Apache è il server Web più usato in assoluto. Un server Web è un programma che riceve le richieste sulla Rete dai browser Web (client) come Netscape o MS Explorer e "serve" (ovvero restituisce) i documenti HTML al browser.

Applicazione - Programma applicativo: Un programma per computer progettato per eseguire un compito specifico come la contabilità, l'analisi di dati scientifici, l'elaborazione testi, o l'impaginazione. In generale, i programmi applicativi si distinguono dal software di sistema (il sistema operativo che fa funzionare il computer), dalle utility di sistema (programmi che eseguono compiti come la produzione di copie di backup o il ripristino di file cancellati) e dai linguaggi di programmazione (usati per creare nuove applicazioni)[2].

Arpa: (Advanced research projects agency) agenzia per i progetti di ricerca avanzata. Aveva lo scopo di gestire i progetti finanziati dal ministero della difesa americano.

Assemblatore: (Assembler) Programma di servizio che traduce le istruzioni di un altro programma scritto in linguaggio simbolico (assemblativo) in codici del linguaggio macchina di un determinato processore.

Assemblativo: (Assembly) Linguaggio di programmazione in cui le istruzioni in codice macchina vengono sostituite con codici mnemonici. Si confonde spesso con il linguaggio macchina dal quale differisce perché le istruzioni sono brevi sigle alfabetiche di significato più facilmente comprensibile anziché codici numerici.

Basic: Acronimo di "Beginners All-purpose Simbolic Instruction Code". Linguaggio ad alto livello semplice e molto noto. Il progenitore del Visual Basic.

Beta: così è definito un programma nella sua versione ancora instabile in corso di revisione.

Beta Test: Periodo di prova di un software nel quale non si garantisce la stabilità e che ha il compito di individuare eventuali bug attraverso l'utilizzo costante degli utenti finali.

Beta tester: Utenti dei programmi beta impegnati nella revisione di un programma.

Blue-box: apparecchi per fare telefonate gratis all'insaputa le compagnie telefoniche.

Bug: Errore di programmazione che può causare un funzionamento errato del software. Il termine ha origine nel 1945 ad Harvard, quando G. M. Hopper rimuove una falena di cinque centimetri dall'interno di un computer sperimentale, in tilt proprio a causa della presenza dell'insetto. Da questo incidente nasce la consuetudine di dire che un computer contiene un bug ("insetto", appunto) quando non funziona correttamente[3].

Bugs report: segnalazione di errori di programmazione e della loro correzione.

Debuggare: (Debugging) correzione degli errori di programmazione.

C: linguaggio di programmazione che utilizza costrutti e strutture ad alto livello, ma che ha capacità di controllo a basso livello tipiche del linguaggio assembly.

Client: indica un programma software, oppure un computer che si rivolge ad una sorgente di informazioni, il server.

Cracker: termine che indica il pirata-vandalo informatico. Il cracker ha gli stessi strumenti e le conoscenze tecniche degli hacker, che usa per rompere le sicurezze dei sistemi per furto o vandalismo. La parola è nata nell'85 dagli hacker per difendersi dall'uso improprio da parte dei giornalisti della parola hacker. Di solito i cracker tendono a riunirsi in "società segrete" e non rivelano all'esterno le loro tecniche.

Codice Sorgente: il codice sorgente di un programma è quella sequenza di istruzioni (codice di linguaggio) che una volta compilata, attraverso un apposito programma detto compilatore, dà origine al programma eseguibile (in codice macchina). Il ruolo del compilatore è paragonabile a quello dell'interprete che sta in mezzo a due soggetti che parlano lingue diverse. Nel modello proprietario la sequenza di istruzioni espressa in codice di linguaggio costituisce segreto industriale ed il programma è già "tradotto" per la macchina. Se distribuiti in forma aperta, la disponibilità dei sorgenti consente qualsiasi controllo e/o modifica del programma.

Compilatore: Programma che trasforma il codice scritto in un linguaggio di programmazione in linguaggio macchina e viceversa.

Credit list: lista di coloro che hanno contribuito alla realizzazione di un progetto. Generalmente è allegato ai programmi sotto forma di file.

Dec: Digital equipment corporation.

ENIAC: Negli USA il primo fu l'ENIAC di Eckert e Mauchly, è circa mille volte più veloce dei suoi predecessori viene completato nel 1945 nel laboratorio dell'Università di Pennsylvania.

Fortran: Uno dei primi linguaggi ad alto livello orientato alle applicazioni scientifiche. Contrazione dei due termini "FORmula TRANslation" a sottolineare la possibilità di tradurre le formule matematiche.

Free Software: Il termine inglese free significa sia gratis che libero. Free in questo contesto è da intendere sempre come libero. Infatti il "Free Software" viene distribuito sia gratuitamente che dietro compenso. Ma a differenza del software gratuito ma proprietario è accompagnato dai sorgenti e da licenze che consentono delle libertà che gli altri copyright non riconoscono.

Hacker[4]: "[...]1. Una persona a cui piace esplorare i dettagli dei sistemi programmabili e i modi per estendere le loro caratteristiche, in contrasto con la maggioranza degli utenti che preferisce imparare solo il minimo necessario. 2. Qualcuno che programma appassionatamente (persino ossessivamente) o che ami programmare piuttosto che limitarsi a teorizzare sulla programmazione. 3. Una persona in grado di apprezzare i valori dell'hacking (vedi). 4. Una persona abile a programmare velocemente. 5. Un esperto in un particolare programma, o che lo usa con particolare frequenza o ci lavora su; come in "Unix hacker". (Le definizioni dalla 1. alla 5. sono correlate e le persone che le soddisfano sono assimilabili.) 6. Un esperto o un appassionato di qualunque tipo. Uno può essere un hacker dell'astronomia, per esempio. 7. Uno che ama la sfida intellettuale di superare o aggirare creativamente le limitazioni. 8. [deprecato] Un ficcanaso maligno che cerca di scoprire informazioni riservate infilandosi da tutte le parti. Da qui "hacker delle password", "hacker di rete". Il termine corretto in questo senso è cracker.
Il termine hacker va inoltre a connotare l'appartenenza alla comunità mondiale definita da Internet [...]. Ciò comporta che le persone descritte devono abbracciare consapevolmente una delle versioni dell'etica hacker (vedi etica hacker). È meglio essere definiti hacker dagli altri piuttosto che descrivere se stessi in questo modo. Gli hackers si considerano in qualche modo come un élite (una meritocrazia basata sull'abilità), nonostante ciò i nuovi membri sono accolti volentieri."

Hackerare: programmare appassionatamente.

Higham institute: club studentesco chiamato scherzosamente così perché si trovava in Higham street a Cambridge.

History file: cronologia delle versioni di un software che include l'indicazione dei singoli miglioramenti e dei relativi responsabili. Generalmente è allegato ai programmi sotto forma di file.

IBM: Acronimo di "International Business Machine". E' stata per lunghissimo tempo l'azienda leader, a livello mondiale, nella produzione e nella distribuzione di computer, per molti, è ancora sinonimo di computer.

Implementare: Creare, completare, perfezionare un programma seguendo una procedura che parte dallo studio, per arrivare al programma funzionante e alla definitiva messa in opera.

Ingegneria sociale: processo di estrazione di informazioni importanti o riservate da coloro che le detengono. L'ingegneria sociale si basa su alcuni presupposti psicologici, come la tendenza delle persone a rispondere a domande dirette che non si aspettano o la tendenza ad aiutare qualcuno che sembra essere in difficoltà. Gli hacker spesso adottano tecniche di ingegneria sociale per raccogliere informazioni nelle fasi iniziali.

Intelligenza Artificiale: Intelligenza Artificiale, la branca dell'informatica che si occupa della simulazione dell'intelligenza umana per mezzo dei computer.

Kernel: è il nucleo essenziale di un sistema operativo responsabile della connessione tra le componenti fisiche di base e tutte le altre parti del sistema stesso. Ha in pratica la funzione di svolgere le operazioni fondamentali, come ad esempio la ripartizione dei carichi di lavoro elaborativo della CPU ("Central Processing Unit", parte del calcolatore dove vengono processate le informazioni), ma gestisce anche le funzioni hardware del sistema.

LAN: Local Area Network. Sistema di cavi, apparati con i relativi software e protocolli che consente lo scambio di informazioni in formato elettronico tra personal computer e server. Il termine indica un sistema che si estende su limitate superfici (tipicamente un palazzo).

Linguaggio macchina: linguaggio di programmazione che consiste in sequenze di cifre binarie (O e I) direttamente interpretabili dal processore.

Lock hacking: hackeraggio delle serrature e/o abile superamento di blocchi fisici (porte, schedari, cassette di sicurezza, ecc.)

Mac (Progetto): Multiple access computing (elaborazione ad accesso multiplo) gruppo di studio per realizzare l'utilizzo multiutente, denominato "time-sharing" (a partizione di tempo) sistema in cui più periferiche fanno capo ad un elaboratore centrale. Finanziato dal ministero della difesa attraverso l'Arpa.

Mcws: Midnight computer wiring society (associazione di mezzanotte per il cablaggio del computer), un'organizzazione creata ad hoc da alcuni hacker del MIT che, in caso di necessità, avrebbe aggirato i regolamenti dei laboratori informatici, contro le manipolazioni non autorizzate dei computer.

MIT: Massachusetts institute of technology, dell'Università di Cambridge, sede dell'la lab, laboratorio per l'Intelligenza Artificiale.

Modalità hacker: Termine di gergo che si riferisce alle "modalità" in cui si può trovare un computer e, per estensione, ad indicare alcune condizioni della vita reale.

Open Source: (sorgenti aperti) un software per poter essere definito Open Source deve fare uso di una delle licenze certificate come conformi dalla Open Source Initiative, organizzazione esclusivamente destinata alla gestione della campagna Open Source e della sua certificazione di marchio, che prevedono come fondamentale e comune condizione il rilascio dei codici sorgenti (da qui Open Sources); nelle intenzioni iniziali "Open Source" doveva essere una certificazione (una forma speciale di marchio che potesse applicarsi secondo i termini ai prodotti altrui) registrata ma non è stato possibile ufficializzarlo. Il marchio attualmente in via di registrazione è "OSI Certified"; il termine Open Source è stato introdotto dall'ala 'moderata' della comunità hacker, più permissiva riguardo al mercato per renderlo ad esso accettabile e per evitare il termine free di Free Software usato dagli hackers più radicali con il suo doppio significato (libero e gratuito) e la sua valenza provocatoria e anticommerciale.

Porting: modifica di un programma in modo tale che sia disponibile per diversi sistemi operativi.

Progetto Mac: (Multiple access computing: elaborazione ad accesso multiplo) gruppo di studio presso il MIT, per realizzare l'utilizzo multiutente, denominato time-sharing (vedi voce) finanziato dal ministero della difesa, attraverso l'Arpa (Advanced research projects agency).

Public Domain: (pubblico dominio) un programma di pubblico dominio è un programma sul quale l'autore abbia rinunciato a tutti di suoi diritti di copyright. In sostanza non ha una licenza; Chiunque può usarlo come meglio crede, può trattarlo come personale proprietà. Si può perfino ri-licenziare, rimuovendo quella versione dal pubblico dominio, o togliendo il nome del suo autore e trattarlo come opera propria.

Real programmer: I "Real programmer", così furono chiamati i primi programmatori, precursori degli hacker, provenivano dai settori dell'ingegneria, della matematica e della fisica. Programmavano in "linguaggio macchina" ossia con sequenze di O e I, operazione lunga e complicata che richiedeva la conoscenza perfetta dell'architettura del calcolatore, dei codici di tutte le istruzioni ed il controllo mentale di molti altri elementi.

Release: Nuova versione di un programma, di un sistema operativo o di un driver, sviluppato con funzionalità aggiunte o problemi risolti.

Server: Può essere due cose, fra loro collegate: un software che consente ad un computer di offrire un particolare servizio ad un altro computer sul quale gira il corrispondente client ma anche il computer che esegue il software server.

Spacewar: primo videogioco della storia.

S&P: Signal and power subcommittee (sottogruppo del Tmrc) i suoi membri gestivano l'impianto per il controllo del plastico ferroviario del club. L'attrezzatura a disposizione dei modellisti era stata in gran parte regalata dalla compagnia telefonica Western Electric, tramite piano di donazioni alle università. Usando quell'attrezzatura i membri dell'S&P approntarono un "impianto mostruosamente ingegnoso" che consentiva il controllo dei singoli treni in qualsiasi punto del plastico.

Subroutine: Sottoprogramma. Porzione di programma che, pur essendo scritta una sola volta, viene utilizzata ripetutamente durante l'esecuzione del programma, permettendo così di accorciarlo, talvolta in modo notevole. Una routine può essere parte di un'altra routine.

Tech Square: edificio del MIT che ospitava i computer.

Time-sharing: (a partizione di tempo), denominato utilizzo multiutente. sistema in cui più periferiche fanno capo ad un elaboratore centrale.

Tmrc: (Tech model railroad club), organizzazione studentesca che si dedicava al modellismo ferroviario. divisa in due gruppi. Il primo riproduceva meticolosamente i modellini dei treni e le scenografie. Il secondo girava intorno al Signal and power subcommittee (S&P).

Tx0: uno dei primi computer funzionanti a transistor.

---

1 Carlini, 2002.
2 http://www.pcpratico.it/servizi/glossario/
3 Simpson's Contemporary Quotations; Time, 16 aprile 1984, cit in "Internet Timeline - Cronologia della Rete" http://www.attivissimo.net di Paolo Attivissimo.
4 Raymond "Jargon File" 4.2.2 20/08/2000

UNIVERSITÀ DEGLI STUDI DI MILANO
Facoltà di Scienze Politiche

SVILUPPO E ORDINAMENTO ISTITUZIONALE NEL MERCATO DEL SOFTWARE: IL MODELLO DEL SOFTWARE LIBERO.

Relatore: Prof. Gian Primo CELLA
Correlatore: Prof. Paolo BORSATO

Tesi di Laurea di:
Sabrina RAGUSA

Anno Accademico 2001-2002