Enabling environments:
Internet, Web, GNU/Linux, Debian

Alberto Cammozzo

22-10-06 rev.3

Internet, il Web e GNU/Linux, sono tecnologie innovative sulle quali si sono basate altre innovazioni. Hanno tra di loro considerevoli elementi in comune, tra cui il fatto di essere aperte a livello della propria architettura (il prodotto), nella loro realizzazione (il processo), nella loro diffusione (distribuzione) e infine nella gestione (la governance). Sono interconnessi e costituiscono un fattore abilitante uno per l'altro. Compongono una catena del valore nella quale ha particolare rilevanza il modo in cui viene affrontata la complessità e accumulata e distribuita la conoscenza.
Il progetto del sistema operativo Debian, basandosi su GNU-Linux e sfruttando a proprio vantaggio le possibilità offerte da una supply chain aperta basata su Internet, affronta la complessità attraverso la stratificazione, la standardizzazione e modularizzazione dei componenti. A sua volta diviene abilitante per altri progetti a valle.

I progetti analizzati, ai quali collaborano sia volontari che imprese, contribuiscano a creare una ambiente abilitante (enabling environment) fertile per l'innovazione e lo sviluppo.

Indice

Introduzione

Questo documento è diviso in tre parti: nella prima si cercheranno i punti in comune tra Internet, il Web e due progetti di *free software* analizzandone questi aspetti: il contesto e i fattori competitivi, alcuni aspetti architetturali (modularità, stratificazione, standardizzazione, interfacce), la regulation e i processi di governance, i principi che ne definiscono la *vision* e conseguenti strategie.

Nella seconda si cercherà di analizzare in maggior dettaglio quale particolare rilevanza hanno Internet come infrastruttura di distribuzione e ambiente per lo sviluppo del *free software* in genere e di come ne costituisca un fattore competitivo specifico.

Nella terza parte si analizzerà un progetto *free software* particolare di grandi dimensioni, il progetto *Debian*, secondo il punto di vista della *value chain,* mettendo in evidenza la trasmissione e elaborazione della conoscenza e come il modello definito nella prima parte (modularità, stratificazione, processi aperti di governance e definizione di strategie) unitamente ai vantaggi competitivi analizzati nella seconda parte ne costituiscano gli elementi caratterizzanti e determinanti per il suo successo.

Nel corso del lavoro si costruirà il termine *enabling environment* per descrivere quell'ambiente (quali quelli analizzati nella prima parte) dotato delle caratteristiche di modularità, stratificazione, processi aperti, govenanace aperta e strategia definita che abilitano non solo attività economiche e competizione ma che possono essere usati a loro volta come infrastruttura o componenti per altri enabling environments.

I - Internet, *world wide web*, GNU/Linux, Debian project: elementi comuni

Practical revolution is made of two things: proof of concept and running code
Eben Moglen, – *Die Gedanken Sind Frei*

The trick fo managers and programmers is to create enough structure to keep projects under control but not so much that "the process" stifles creativity and flexibility
Michael Cusumano – *The business of software*

La più recente delle rivoluzioni della comunicazione è basata sulla rete delle reti chiamata Internet e sulla struttura di gstione delle informazioni ipertestuale e multimediale detta "Web" che vi si appoggia. Il comportamento della rete è determinato dal software e dall'hardware che ne costituisce l'architettura fisica. Il *free/open source software*[1], ovvero codice governato da licenze che ne consentono liberamente visione, modifica e ridistribuzione, è da sempre al centro di questa rivoluzione. E' possibile vedere il *free software* accompagnare la rivoluzione informatica e telematica sin dai suoi albori e influenzarla sia attraverso contenuti tecnologici propriamente detti, sia attraverso il proprio

1 Senza entrare nel merito delle distinzioni terminologiche e filsofiche, si userà prevalentemente il termine *free software* riferendosi alle origini e alla natura aperta riguardante la conoscenza e il termine *open source* per riferirsi all'aspetto commerciale e tecnico. La sigle f/oss (*free/open source software*) denota in modo neutrale una sintesi dei due termini precedenti.

particolare processo di comunicazione della conoscenza (Benussi 2005).

Un particolare *free software*, il sistema operativo GNU/Linux, è nato da una comunità di programmatori distribuita nel mondo e sviluppatasi grazie a Internet. In realtà è composto dalla fusione di due progetti: il primo, denominato GNU (*GNU is Not Unix*) è nato negli anni '80 con lo scopo di produrre un sistema Unix libero, cioè disponibile per esser studiato, adattato, modificato, migliorato e ridistribuito da chiunque avesse la competenza e l'interesse a farlo. Il secondo è quello del finlandese Linus Torvalds, che grazie alla collaborazione di altri programmatori volontari reclutati in Intenet, ha costruito e ora coordina il *kernel* (nucleo) del nuovo sistema operativo. I due progetti, inizialmente in parziale competizione, si sono combinati, pur mantenendo la loro individualità, in un unico progetto modulare.

Il progetto Debian cura, esclusivamente su base volontaria, lo sviluppo e la manutenzione di una delle più diffuse e stimate modalità di distribuzione (*distribution*) del sistema operativo GNU/Linux.

Si possono notare a prima vista alcuni elementi chiave in comune tra Internet, il Web, GNU/Linux e la distribuzione Debian:

- sono sviluppati grazie alla *cooperazione* di soggetti coordinati solitamente in modo non stretto;
- sono *modulari*, il che rende agevole il lavoro cooperativo dei vari soggetti coinvolti in modo indipendente;
- sono *stratificati*: la loro costruzione in strati o livelli di astrazione successivi consente di isolare gli aspetti di complessità;
- si articolano attorno a interfacce *standard* tra strati e moduli, e gli standard sono *aperti* e pubblici;
- sono condotti con dei processi di governance aperti, centrati su documenti che definiscono scopi, *mission*, e procedure chiari e condivisi. I processi decisionali sono focalizzati sulla creazione di consenso da parte dei vari stakeholder;
- la natura aperta dei progetti li ha resi versatili e adattabili alle opportunità presentatesi, facendoli prevalere su altri progetti concorrenti;
- essi stessi, stratificandosi tra loro, concorrono a creare un *ambiente abilitante* che è fertile humus per altri progetti, spesso con rilevanti risvolti economici.

Di seguito analizzeremo per ciascuno dei progetti: fattori competitivi, stratificazione, modularità, governance, strategie.

1 – Internet

Fattori competitivi

Teorizzate negli anni '60, le reti geografiche di calcolatori si sono diffuse commercialmente negli anni '70. Le principali erano: SNA, architettura proprietaria di IBM; DECnet, di Digital; Bitnet, architettura aperta accademica. Oltre a queste, vi erano quelle che oggi sarebbero definite *peer-to-peer*: hobbystiche, tra cui la più popolare era Fidonet, o accademiche come UUCP.

Sviluppata negli USA grazie a finanziamenti governativi tra gli anni '80 e '90, Internet si è imposta su altre reti e le ha quasi completamente sostituite, diventando lo standard *de facto* di rete geografica per l'interconnessione di reti locali. Perchè? Gli elementi di vantaggio presentati da Internet sono stati sicuramente molteplici. Hanno certamente avuto un ruolo importante le tecnologie aperte dei

protocolli TCP/IP che ne compongono la base: queste erano già presenti nei computer Unix diffusi in ambito accademico ed erano disponibili nei PC che in quegli anni si stavano diffondendo. Anche chi non sapeva dell'esistenza di Internet probabilmente già usava o comunque disponeva delle tecnologie sulle quali questa si basa. Le tecnologie aperte non imponevano, a differenza di quelle proprietarie, di affidarsi a un fornitore specifico per la connettività e di dipendere dalle sue scelte commerciali e tecnologiche.

Un altro fattore cruciale nella genesi di Internet è stato l'aspetto cooperativo: le reti si sono aggregate in base alle esigenze concrete emergenti, formando un tutt'uno grazie ad un processo che non ha richiesto un intervento verticistico programmato. Anche se per alcuni aspetti tecnicamente inferiore ad alcune architetture di rete avanzate, grazie alla sua flessibilità Internet riuscì a fornire gli stessi servizi offerti dalle reti preesistenti e a sostituirsi gradualmente ad esse.

Inoltre, lo sviluppo del *world wide web,* alla base del successo presso i consumatori, della e-mail, e in seguito dei motori di ricerca ha innescato delle esternalità di rete grazie alle quali Internet è divenuta *La Rete* per antonomasia. Il processo di generazione complessivo che ha portato all'emersione dell'attuale Internet è quello di *coalescenza*, in cui reti si sono aggregate prima tra loro e poi alla Rete come delle goccioline di liquido diventano gocce più grandi appena entrano in contatto tra loro.

Stratificazione

La stratificazione è una caratteristica comune a ogni rete di calcolatori avanzata: ogni strato rappresenta un'astrazione che fornisce servizi allo strato superiore, senza che quest'ultimo debba conoscere i dettagli implementativi degli strati sottostanti. Lo strato più basso è quello fisico, che gestisce la trasmissione di impulsi elettrici, radio o ottici, quello più alto è quello applicativo, a cui corrispondono servizi specifici come la trasmissione di posta elettronica o di file.

Internet non si preoccupa di definire gli strati più bassi, cioè quello di interconnessione fisica e gestione della connessione tra punti fisici, lasciando il compito a chi ha progettato le tecnologie di collegamento (ad esempio le reti cablate o *wireless*). Gli strati superiori, quelli dei programmi che interagiscono con l'utente, sono lasciati ai programmatori: questo fa si che gli strati intermedi specifici di Internet possano essere molto pochi: al limite due.

Questa semplicità e indipendenza dall'infrastruttura fisica ne ha consentito l'implementazione su reti eterogenee senza richiedere costosi e interventi strutturali. Il numero ridotto di strati ha facilitato l'implementazione, anche a costo di qualche compromesso tecnico. Una implementazione concorrente a Internet, proposta dalla International Organization for Standardization (ISO), molto avanzata sul piano teorico, soffriva di una eccessiva complessità (7 strati) con conseguente pesantezza implementativa. I sette livelli della stratificazione ISO-OSI vengono usati tutt'ora a scopo didattico-teorico per distinguere concettualmente le funzioni pertinenti ai vari livelli, ma nella pratica Internet si è rivelata sufficientemente funzionale.

Modularità

Internet può essere definita come una Rete composta da reti indipendenti e autonome una rispetto alle altre. Interconnetere tra loro le singole reti non richiede modifiche alla loro struttura interna. Alcune reti si sono specializzate nel fornire interconnessioni tra altre reti locali contigue, costituendo una infrastruttura di interconnessione: una rete locale (impresa, università, ente, privato) si collega a una rete specializzata nel fornire connettività (*Internet service provider* – ISP) che a sua volta è connesso con altri ISP.

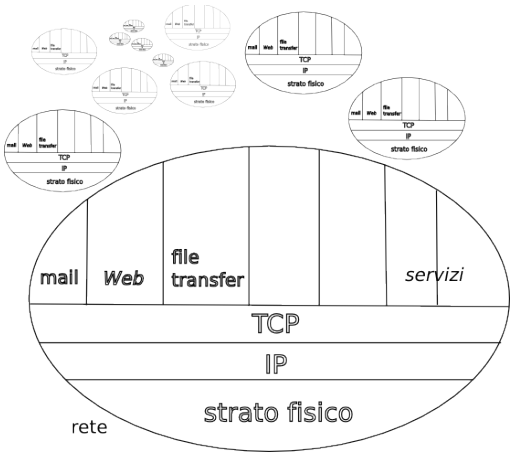

Figura 1: strati e moduli in Internet

Ogni rete può essere vista come un modulo il cui funzionamento è autonomo ed indipendente dalle altre reti, e che può collegarsi alla rete di reti anche temporanemente. Le conoscenze necessarie per la realizzazione e gestione di una rete sono concettualmente identiche per ogni rete, indipendentemente dalla scala.

La modularità si ritrova anche al livello dei servizi che una rete è in grado di offrire (posta elettronica, navigazione di siti Web, trasferimento file), per cui ad ogni servizio corrisponde uno o più protocolli standard specifici, indipendenti dagli altri, che possono essere affiancati tra loro arricchendo le capacità funzionali della rete che li ospita (Figura 1).

Cooperatività, coordinamento e *governance*

La gestione e il corretto funzionamento della Rete non è affidato alle cure di un solo ente o impresa, ma, grazie alla modularità, ai singoli gestori delle reti che la compongono, che dovranno rispettarne regole di interoperabilità e standard tecnici. Ad alcuni enti è demandata la minima indispensabile attività di coordinamento tecnico. Questi enti sono principalmente due: l'Internet Engineering Task Force[2] (IETF), che accetta le proposte dei protocolli considerati ufficiali in base a criteri prevalentemene tecnici, e la *no profit corporation* Internet Corporation For Assigned Names and Numbers[3] (ICANN) la cui attività prevalente è l'assegnazione di una numerazione univoca alle singole reti.

Nonostante questa indipendenza tecnica, negli aspetti pratici ICANN dipende dal Governo degli Stati Uniti, sia per motivi storici, dato che Internet è nata da reti finanziate dal governo USA, sia per il peso considerevole delle reti ivi presenti. E' in corso un dibattito sulla internazionalizzazione di questo ente.

IETF, nell'ambito della Internet Society[4] è una organizzazione di volontari che sviluppa e promuove gli standard su cui si basa Internet. Ha il ruolo di regolamentare il modo in cui i moduli e gli strati si interfacciano tra loro attraverso protocolli, e quindi è un ruolo di enorme importanza. E' organizzata in gruppi di lavoro *ad-hoc* che si occupano di progetti specifici all'interno di più ampie aree tematiche. I responsabili delle varie aree compongono lo *steering committee*.

All'interno dei vari gruppi di lavoro le decisioni vengono prese in base a criteri di consenso (Bradner 1998):

> *The core rule for operation is that acceptance or agreement is achieved via working group "rough consensus". [...]*

2 Per informazioni su IETF: http://www.ietf.org/
3 Per informazioni su ICANN e la sua funzione: http://www.icann.org/general/
4 La Internet Society è una organizzazione internazionale con la finalità di promuover lo sviluppo di Internet: http://www.isoc.org/isoc/

IETF consensus does not require that all participants agree although this is, of course, preferred. In general, the dominant view of the working group shall prevail. (However, it must be noted that "dominance" is not to be determined on the basis of volume or persistence, but rather a more general sense of agreement.) Consensus can be determined by a show of hands, humming, or any other means on which the WG [Working Group] agrees (by rough consensus, of course). Note that 51% of the working group does not qualify as "rough consensus" and 99% is better than rough. It is up to the Chair to determine if rough consensus has been reached.

Una buona sintesi e un significativo segno della differenza della infrastruttura di governance di Internet da quelle tradizionali può essere rilevato da come questi organismi vengono visti dall'ITU, organismo internazionale per il governo delle telecomunicazioni di impostazione più tradizionale(Hayashi 2003).

In order to maintain the integrity of the global telecommunications network, standard setting, rule making and monitoring of the system is necessary on the global scale. Such a task has been performed by ITU, ICANN and IETF. ITU is administered as part of the United Nations on the traditional international legal framework, i.e., its membership comprises individual sovereign nations, decisions are made on the unanimity basis, and the agreements are concluded as international treaty between ITU and the participating nations.
On the other hand, the internet technology requires new forms of global organization. For example, the Internet Corporation for Assigned Names and Numbers (ICANN) handles IP address space allocation, protocol parameter assignment, domain name system management, and root server system management functions. The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the internet architecture and the smooth operation of the internet.
The outstanding characteristics of these new organizations are that they take individuals as members. No corporations or countries have representation in them. They are not inter-national organizations since they do not act on nations. They are trans-national organizations because ***they involve concerned individuals*** *across countries and regions. Also, they use a non-traditional approach in which participation and compliance is only voluntary.*

Principi e sviluppo di strategie

Come si è visto IETF ha regole formalizzate ma molto poco formali[5]. Analogamente alle imprese, IETF e gli altri enti basano la loro azione strategica su degli *statement*, delle affermazioni di principi, che a volte sono formulate esplicitamente come *mission* o *vision.* Come vedremo, in altri casi queste affermazioni assumono la forma di statuti, dichiarazione di principi o addirittura dichiarazione di una *filosofia.*

Ad esempio, per quanto concerne IETF, vi sono scopi e mission ben definiti (Alvestrand 2004):

The ***goal*** *of the IETF is to make the Internet work better.*
The ***mission*** *of the IETF is to produce high quality, relevant technical and engineering documents that influence the way people design, use, and manage the Internet in such a way as to make the Internet work better. These documents include protocol standards, best current practices, and informational documents of various kinds.*

5 Come esempio di informalità basti il *dress code* di IETF:
Dress Code: since attendees must wear their name tags, they must also wear shirts or blouses. Pants or skirts are also highly recommended. Seriously though, many newcomers are often embarrassed when they show up Monday morning in suits, to discover that everybody else is wearing t-shirts, jeans (shorts, if weather permits) and sandals. http://www.ietf.org/tao.html#2.3

Lo stesso documento delina il processo e le risorse attraverso il quale intende raggiungere lo scopo rimenendo nell'ottica della propria *mission*; il che, secondo diverse definizioni (Pellicelli 2005), coincide con una *strategia* (Alvestrand 2004):

> *The IETF will pursue this mission in adherence to the following **cardinal principles**:*
>
> ***Open process** - any interested person can participate in the work, know what is being decided, and make his or her voice heard on the issue. Part of this principle is our commitment to making our documents, our WG mailing lists, our attendance lists, and our meeting minutes publicly available on the Internet.*
>
> ***Technical competence** - the issues on which the IETF produces its documents are issues where the IETF has the competence needed to speak to them, and that the IETF is willing to listen to technically competent input from any source. Technical competence also means that we expect IETF output to be designed to sound network engineering principles - this is also often referred to as "engineering quality".*
>
> ***Volunteer Core** - our participants and our leadership are people who come to the IETF because they want to do work that furthers the IETF's mission of "making the Internet work better".*
>
> ***Rough consensus and running code** - We make standards based on the combined engineering judgement of our participants and our real-world experience in implementing and deploying our specifications.*
>
> ***Protocol ownership** - when the IETF takes ownership of a protocol or function, it accepts the responsibility for all aspects of the protocol, even though some aspects may rarely or never be seen on the Internet. Conversely, when the IETF is not responsible for a protocol or function, it does not attempt to exert control over it, even though it may at times touch or affect the Internet.*

Queste dichiarazioni di principi non sono solo affermazioni generiche o tese a creare un'immagine dell'ente, ma reali principi ispiratori che devono essere condivisi dai partecipanti alle attività, incluse le imprese[6] e che ne indirizzano le attività.

Per un approfondimento sulla governance *multi-stakeholder* di Internet si veda l'esteso rapporto delle Nazioni Unite sulla governance di Internet (MacLean 2004).

2 – Il World Wide Web

Fattori competitivi

L'idea che l'informazione potesse assumere dimensioni diverse con l'avvento dell'informatica ha radici lontane. Negli anni '60 con NLS, ovvero "oNLine System", Doug Englebart, oltre all'impiego del mouse, delle finestre e della grafica[7], dimostrava l'uso degli *ipertesti*. Ted Nelson nello stesso periodo iniziava a lavorare al progetto Xanadu[8], ambizioso software ipertestuale e multimediale che già dagli anni '70 prendeva in considerazione anche l'accesso ai documenti via rete[9]. Poco prima che Tim Berners-Lee inventasse il *world wide web* (Www o *web*), era in uso un altro

6 Sta suscitando un grande dibattito la violazione di uno dei principi, la *neutralità* di Internet rispetto al contenuto, sulla base della proposta di alcuni Internet Service Providers di discriminare la performance della loro rete in base al fornitore del contenuto. E' verosimile che la tutela di questo principio richieda un intervento normativo esplicito (Bennet 2006).

7 Le registrazioni di una dimostrazione sono visibili a questo indirizzo: http://sloan.stanford.edu/mousesite/1968Demo.html

8 http://xanadu.com

9 Ted Nelson rilasciò i sorgenti di Xanadu con il nome Udanax nel 1998, nella speranza di veder utilizzati tecniche ed

metodo di diffondere informazione sulla rete, chiamato *gopher*[10], sviluppato all'Università del Michigan nel 1991. Gopher poteva essere usato da un semplice terminale[11], ma non consentiva di visualizzare simultaneamente grafica e testo nella stessa pagina. Inizialmente un progetto a codice sorgente aperto, venne chiuso (cioè il suo codice sorgente reso inaccessibile e proprietario) nel 1993. In quello stesso anno in venne lanciato il *web*, e il il *browser* grafico Mosaic[12] sviluppato da Marc Andreessen. Il nuovo programma, finanziato da fondi governativi USA e distribuito gratuitamente anche se non in forma di codice sorgente, consentiva la visione di testo e grafica simultaneamente. Il *web* ipertestuale e multimediale venne inizialmente adottato da enti di ricerca e università; poi, nel corso degli anni '90, da imprese, enti e privati, fino a diventare uno dei principali canali di comunicazione, assieme all'e-mail.

Oltre all'architettura aperta il vantaggio competitivo del *web* rispetto alle altre tecnologie esistenti fu la disponibilità gratuita di un mezzo di consultazione facile da usare e più attraente.

Stratificazione

In estrema sintesi il *web* si basa sia su una stratificazione tra web server e client (il browser), sia su una stratificazione di protocolli: uno riguarda la composizione delle pagine da visualizzare (HTML - *hyper text markup language*) e gli altri il trasferimento delle informazioni dal server al *browser* (di cui il più diffuso è l'HTTP - *hyper text transfer protocol*).

Questa semplificazione risulta sempre più insufficiente a descrivere una realtà che si arricchisce di protocolli, moduli *plug-in,* estensioni e nuove modalità di dialogo tra server e client in modo talvolta caotico, nel tentativo di introdurre nuove tecnologie soprattutto da parte di imprese che tentano di acquistare vantaggi competitivi basati su tecnologie proprietarie.

Un esempio emblematico di questo fenomeno è quello della *browser war* che vide contrapposte Microsoft e Netscape nel '95 e che in realtà non è ancora finita[13], in particolare se si considerano le potenzialità connesse con il mercato dei "contenuti".

Modularità

Come abbiamo visto per Internet, la modularità può essere vista sia per quanto riguarda le *risorse*, nel caso del web la molteplicità di *siti* che lo compongono, che in quello delle *tecnologie* usate, come già evidenziato nella stratificazione.
La grande utilità del web deriva dall'abbondanza delle informazioni reperibili. Il processo di arricchimento, inizialmente partito da pochi fornitori, coinvolge progressivamente sempre più soggetti che interconnettono le proprie informazioni, in un processo di coalescenza analogo a quello di Internet. Inizialmente erano coinvolte università ed enti di ricerca, in seguito imprese di settori tecnologici (prevalentemente ICT), poi grandi imprese di altri settori, poi pubbliche amministrazioni, fino a i singoli cittadini, con l'esplosione recente dei *blog*. Più aumentano le informazioni, più aumenta l'interesse, specie grazie alla possibilità di ricerca e selezione offerta dai

algoritmi da lui elaborati: http://www.udanax.com/

10 Una descrizione di gopher nel *gopher manifesto*: http://www.scn.org/~bkarger/gopher-manifesto Alcuni server gopher possono essere ancora consultati con i browser moderni, che ne hanno incorporato i protocolli. Si veda ad esempio: gopher://gopher.quux.org/1/Software/Gopher

11 Alcuni sostenitori del protocollo *gopher* lo vedono oggi adatto all'uso in contesti a banda limitata, ad es. telefonini.

12 Il browser divenne il prodotto dell'impresa Netscape Communication e prese il nome di Netscape Navigator, antenato del browser *open source* Firefox. Il codice sorgente del Navigator venne rilasciato in risposta alla posizione dominante di Internet Explorer.

13 http://en.wikipedia.org/wiki/Browser_wars

motori di ricerca.

Cooperatività, coordinamento e governance

L'aspetto cooperativo del web risulta particolarmente evidente nella capacità dei siti di offrire riferimenti incrociati (*link*) gli uni verso gli altri e di fornire assieme un contesto, un ambiente, in cui le informazioni diventano utilizzabili ed acquistano valore di conoscenza. Questo ulteriore passo viene svolto dai motori di ricerca[14] che sono all'apice della complessità caotica formata dalla molteplicità dei siti e delle informazioni eterogenee, e forniscono l'ordine e l'usabilità che trasforma l'informazione in conoscenza.

Un altro aspetto di cooperazione risulta evidente nei siti cooperativi: l'esempio più emblematico è l'enciclopedia *wikipedia*[15], in cui migliaia di *contributors* compilano e correggono le voci di una enciclopedia aperta.

Principi e sviluppo di strategie

La governance del *web* implica la definizione di standard e protocolli tecnici: questi compiti sono affidati ad un consorzio internazionale, denominato World Wide Web Consortium (W3C), guidato da Tim Berners-Lee, inventore del web stesso. Anche se la partecipazione alle attività del W3C sono aperte anche ai privati, esse sono principalmente orientate a organizzazioni e imprese. Gli standard vengono deliberati seguendo una procedura di revisioni successive alla fine della quale viene raggiunto il consenso (*Working Draft, Candidate Recommendation, Proposed Recommendation, Recommendation*). Le raccomandazioni del W3C non sono degli standard ad adozione obbligatoria, per scelta esplicita del W3C: è consentito ai membri di discostarsi da essi. Questo crea problemi di interoperabilità: in assenza di standard obbligatori, vi è la tendenza per chi detiene posizioni di leadership del mercato di difendere il proprio vantaggio creando barriere con l'introduzione di tecnologie proprietarie, o estensioni proprietarie degli standad aperti.

La *mission* dichiarata del W3C è

> *To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web.*[16]

Il processo di governance impone la massima attenzione a tutti gli stakeholders:

> *All stakeholders can have a voice in the development of W3C standards, including Members large and small, as well as the public. W3C processes promote fairness, responsiveness, and progress: all facets of the W3C mission.*[17]

Tra gli scopi, quello di sviluppare un web aperto accessibile a tutti per la condivisione della conoscenza (Bratt 2006).

> *The social value of the Web is that it enables human communication, commerce, and opportunities to share knowledge. One of W3C's primary goals is to make these benefits available to all people, whatever their hardware, software, network infrastructure, native language, culture,*

14 I motori di ricerca sono una tecnologia in fortissima espansione e che vedrà in futuro ancora molte innovazioni. La situazione di sostanziale monopolio detenuta da Google suscita non poche preoccupazioni per la mancanza di trasparenza degli algoritmi di classificazione delle informazioni e l'influenza che questo ha sul processo di formazione della conoscenza.

15 http://en.wikipedia.org

16 http://www.w3.org/Consortium

17 http://www.w3.org/Consortium/process.html

geographical location, or physical or mental ability.

E' opportuno notare che la *policy* nei confronti di eventuali brevetti impone che essi siano *royaly-free.*

3 – Il Kernel Linux

Come già detto, il sistema operativo Gnu/Linux è un sistema composto da due progetti distinti, il *kernel* Linux e la collezione dei programmi di sistema GNU della Free Software Foundation, sopra ai quali si appoggiano gli altri programmi compatibili con il sistema sottostante. Vi sono diversi modi di integrare insieme questi elementi, che si riflettono in diverse *distribuzioni* di GNU/Linux. Ogni distribuzione comprende un kernel Linux, un sistema base GNU e un assortimento di altri programmi. Delle distribuzioni parleremo oltre, in questa sezione prenderemo in considerazione soprattutto il nucleo, lo strato più basso, ovvero il kernel Linux, originariamente sviluppato da Linus Torvalds quando era studente in Finlandia e sempre coordinato da lui anche successivamente.

Fattori competitivi

Al contrario di Internet e del *web,* Linux si trova in un contesto fortemente competitivo. Secondo l'analisi degli Open Source Development Laboratories (OSDL)[18], il mercato nel quale Linux si trova può essere segmentato come segue: *silicon supliers, system vendors, Linux operating system providers, middleware independent software vendors (ISV), applications ISV, system integrators* o *service providers, equipment suppliers, academic-university, end users*[19]. Una analisi del settore va oltre lo scopo di questo lavoro, per cui prenderemo in considerazione solo alcuni segmenti in cui è più evidente un ruolo della *supply chain*. Si tenga presente che diverse imprese sono presenti in più di un segmento.

- *System vendors*: il loro prodotto è il computer completo di sistema operativo: solitamente viene installato Microsoft Windows, ma alcuni fornitori offrono anche GNU/Linux, prevalentemente sui server. In aggiunta i *system vendor* offrono servizi aggiuntivi di assistenza. La partnership con Microsoft ha talvolta comportato effetti sgradevoli per i system vendors[20], e vi sono dubbi sulla loro effettiva libertà di offrire GNU/Linux preinstallato in alternativa a Windows, almeno a breve termine.
  Nonostante il duro confronto con il sistema operativo offerto da un monopolista, GNU/Linux continua a crescere più del concorrente nelle installazioni sui server (Fondati 2006, Shackland 2004), e in misura molto minore, anche nei desktop. E' stato osservato che una volta che un software free-open entra in un mercato, è estremamente difficile elaborare delle strategie competitive per espellerlo (Lindman 2004)
- Gli *equipment suppliers* forniscono apparecchiature quali *appliance*, sistemi di controllo industriali, telefonini, nei quali il sistema informatico è *embedded.* In questi sistemi l'utente interagisce secondo percorsi predefiniti piuttosto rigidi, di conseguenza il produttore esercita un controllo verticale completo su tutte le componenti del sistema, dall'hardware specializzato fino all'interfaccia utente. Diversamente da quanto accade per i sistemi operativi per pc desktop questo segmento è frammentato, e Linux compete con sistemi operativi *embedded* consolidati, quali VxWorks, QNX, e nel quale Microsoft ha una quota confrontabile con quella di altri. Gli

18 http://groups.osdl.org
19 http://groups.osdl.org/about/industry_segments
20 In una analisi del processo a Microsft per abuso di posizione dominante contro Netscape, si evidenzia l'uso dei system vendors da parte di MS per escludere la concorrenza:
«By precluding Navigator from a main channel of distribution (through OEMs) Microsoft virtually ensured Navigator's neutrali zation as an alternate platform.» (Spinello 2003).

equipment suppliers non sembrano avere orientamenti precisi, e spesso usano sistemi operativi diversi per prodotti diversi. Alcuni studi (Turtley 2005) suggeriscono un crescente e generalizzato interesse per Linux, che offre diversi prodotti mirati per i sistemi embedded, e altri sistemi operativi *open*.

- *Linux OS providers*: sono i soggetti che vendono *distribuzioni* sviluppate in proprio e i *servizi* basati su di esse. RedHat e Novell (dopo l'aquisizione di SUSE Linux) sono le più note tra i soggetti commerciali: offrono una propria *distribution* e servizi quali installazione, assistenza e manutenzione, aggiornamenti. Come avviene per il segmento *system vendors*, sono in diretta competizione con Microsoft sul segmento server e si apprestano ad erodere anche quello desktop. Inoltre sono in competizione tra loro. Si rivolgono a imprese che desiderano avere, oltre ai vantaggi propri dei *prodotti* free software, un *servizio* stabile. E' legittimo attendersi delle evoluzioni nell'area dell'estremo oriente, nel quale agiscono alcune imprese che si dimostrano molto dinamiche.

Stratificazione

I sistemi operativi in genere sono stratificati. Il primo grande livello di distinzione è tra *kernel* o nucleo e i programmi applicativi. Nel caso di Linux la stratificazione è presente sia nel processo di creazione del kernel che nel kernel stesso.

Le stratificazioni nel prodotto sono quelle tipiche dei sistemi operativi, e in questo non vi sono significative innovazioni: all'interno del nucleo, partendo dallo strato più vicino all'hardware, vi sono strati successivi di *astrazione* che ne rappresentano le caratteristiche verso l'alto, e mantengono uno stato coerente con delle strutture dati.

Si può vedere una stratificazione ulteriore nei programmi fuori dal nucleo: una decrizione classica del sistema operativo Unix, e di conseguenza di Linux, sfrutta proprio l'immagine della cipolla con i suoi successivi strati (Figura 2). Linux non presenta innovazioni tecniche di gran rilievo rispetto a Unix.

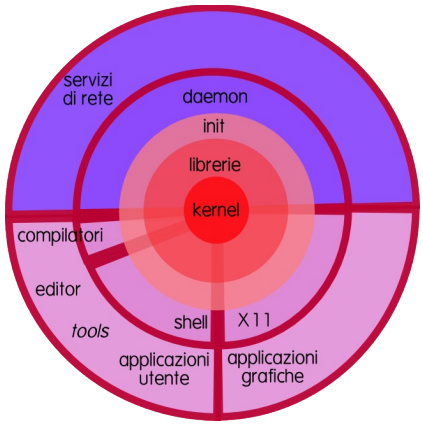

Figura 2: stratificazione nel sistema operativo

Fuori dal nucleo, GNU/Linux è composto da strati sviluppati da soggetti diversi: questo può comportare delle difficoltà di adattamento che devono essere superate da chi compone il sistema, tipicamente da chi realizza la *distribution*. Realizzare una distribuzione richiede quindi un insieme di conoscenze adatto a scegliere e combinare le interfacce tra i vari strati, oltre che tra i vari programmi.

La stratificazione nel processo: analogamente a molti altri progetti software, nel kernel Linux vengono simultaneamente mantenute diverse (solitamente due) versioni concorrenti, con diversi livelli di innovazione: attualmente sono sviluppate le versioni 2.4 e 2.6: la prima subisce solo aggiornamenti di manutenzione, la seconda di reale innovazione. Quando vi saranno innovazioni tali da giustificare una significativa discontinuità con la versione attuale, il ramo 2.6 verrà replicato con la denominazione 2.7, nella quale avranno luogo le innovazioni più significative.

Modularità

La modularità nel kernel Linux sta nella sua architettura, o struttura interna. E' possibile infatti inserire funzionalità nel nucleo senza doverlo sostituire integralmente. Questo consente una ulteriore semplicità nel processo di sviluppo. La modularità è una necessità in programmi di grandi

dimensioni, che richiedono la suddivisione del lavoro tra programmatori, specie se dislocati geograficamente.

Se oltre al kernel vogliamo guardare ad un sistema GNU/Linux intero (Figura 3), la suddivisione in moduli può essere ritrovata nel gran numero di programmi disponibili, sviluppati indipendentemente da una pluralità di soggetti eterogenei (imprese, privati, gruppi formati *ad-hoc* o comunità stabili). Nel nucleo, i sottosistemi di gestione di rete e di file sono moduli, i programmi che forniscono l'infrastruttura grafica (X11, Gnome, KDE) altrettanto, e così via. Come vedremo oltre, non solo il prodotto è modulare, ma anche il processo che porta a comporlo.

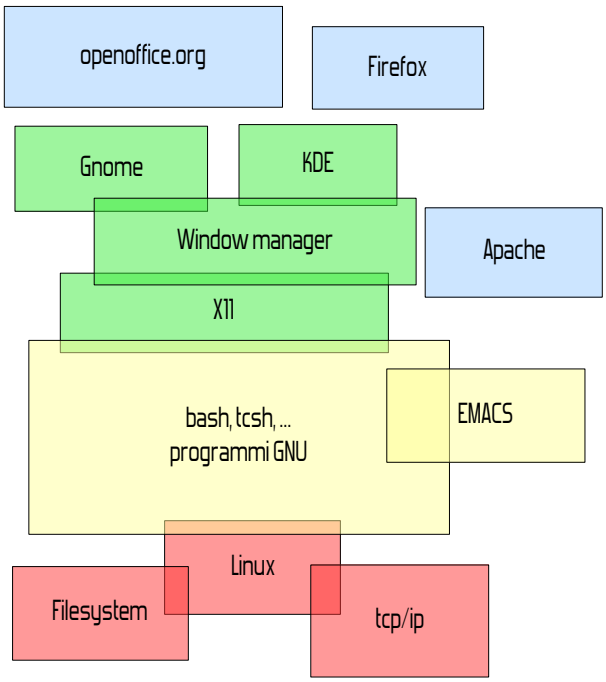

Figura 3: mappa di un sistema free

Cooperatività e coordinamento

Linus Torvalds rimane il coordinatore del progetto Linux, e ha l'ultima parola sullo sviluppo del kernel. I programmatori che sviluppano di nuovi moduli del kernel e seguono l'aggiornamento evolutivo di quelli esistenti collaborano per mezzo di Internet principalmente con liste di discussione dedicate ai loro specifici progetti, ed eventualmente siti web che contengono notizie e documentazione. Siti e liste sono pubblici e consentono l'accesso alle informazioni maturate nel processo di sviluppo.

Dato l'enorme interesse di molte imprese del mercato ICT per Linux, dal 2000 alcune di queste si sono consorziate in una organizzazione non profit per finanziare dei laboratori, gli Open Source Development Laboratories[21] (OSDL), con sedi in USA e Giappone, nei quali lo sviluppo di Linux possa avvenire con la minima difficoltà possibile e per accelerarne lo sviluppo e l'adozione.

Principi e sviluppo di strategie

I principi che guidano lo sviluppo del kernel Linux sono stati in passato e rimangono ora strettamente tecnici e determinati dalle scelte di Linus Torvalds, detentore del marchio e del copyright, che si è mantenuto solitamente distante da prese di posizione ideologiche. Lo sviluppo avviene ora avvalendosi della interazione con gli OSDL, la cui *mission*[22] dichiarata è:

> *To accelerate the deployment of Linux for enterprise computing through:*
> *- Enterprise-class testing and other technical support for the Linux development community.*
> *- Marshalling of Linux-industry resources to focus investment on areas of greatest need thereby eliminating inhibitors to growth.*
> *- Practical guidance to our members - vendors and end users alike - on working effectively with the Linux development community.*

Sono obiettivi eminentemente strategici: identificazione ed eliminazione delle debolezze e criticità nel prodotto, sviluppo di processi efficienti, analisi di minacce ed opportunità. L'insieme delle

21 http://www.osdl.org
22 http://www.osdl.org/about_osdl

imprese che compongono gli OSDL sono tra loro concorrenti, tuttavia contribuiscono allo sviluppo di strategie attorno a un prodotto che, rimanendo libero, è per loro una tecnologia che consente di sviluppare vantaggi competitivi, una *enabling technology*. Ma per quale motivo imprese anche in competizione tra loro dovrebbero investire risorse in un progetto che non possiedono e che viene messo a disposizione anche della concorrenza? E' una azione strategica simile a ciò che Porter e Kramer chiamano, in *Corporate Philantropy* (Porter, Kramer 2002), la *costruzione del contesto competitivo*. Sono presenti infatti tutti e quattro i fronti identificati dagli autori:

> *A company's competitive context consists of four interrelated elements of the local business environment that shape potential productivity: factor conditions, or the available inputs of production; demand conditions; the context for strategy and rivalry; and related and supporting industry.*

1. *Factor conditions.* Un solo prodotto, Linux, costituisce una infrastruttura valida per tutti i soggetti coinvolti.
2. *Context for Strategy and Rivalry.* Il contesto competitivo è definito attraverso una policy comune, costituita dalla licenza GPL valida per tutti gli stakeholder (programmatori, imprese, utenti finali), nelle stesse identiche condizioni nei confronti del codice prodotto.
3. *Related and Supporting Industries.* La standardizzazione del prodotto consente la creazione di *cluster* di imprese che lavorano in identiche condizioni di partenza, sullo stesso identico prodotto standardizzato, e sul quale possono intervenire in base alle proprie scelte strategiche e valorizzando la propria creatività.
4. *Demand conditions.* I *sophisticated users* di Linux non solo sono in grado di anticipare la domanda della maggioranza degli utenti, ma possono entrare nel ciclo dalla parte della produzione, adattando il prodotto alle proprie esigenze e sottoponendo alla collettività degli utenti e dei produttori le proprie proposte.

Per quanto riguarda Linux, Internet e il Web, sembra essere quanto mai appropriata l'affermazione (Porter, Kramer 2002):

> *Philanthropy can often be the most cost-effective way for a company to improve its competitive context, enabling companies to leverage the efforts and infrastructure of nonprofits and other institutions.*

4 – Il progetto Debian

I programmi disponibili nel mondo *f/oss* sono probabilmente decine di migliaia. La maggior parte sono disponibili sul sistema GNU/Linux, ma l'installazione di un programma appena uscito dalle mani del programmatore e integrarlo in un sistema può richiedere esperienza ed essere un processo piuttosto laborioso. Le *distribuzioni* GNU/Linux affrontano questo problema e offrono all'utente i programmi già pronti per essere installati da *pacchetti* integrabili tra loro senza difficoltà.

Il progetto Debian produce e mantiene le più diffuse distribuzioni non commerciali, e le più estese come numero di programmi offerti. I pacchetti attualmente forniti nella distribution corrente sono più di 15'000: si tratta probabilmente del più ampio sistema operativo mai realizzato (Amor-Iglesias et al 2005). Il principale lavoro di Debian non è quello di produrre software ma quello di integrare nel pacchetto l'esperienza di un sistemista esperto, fornire un insieme di servizi per la manutenzione dei pacchetti, e manutenerli. I dettagli di questa *knowledge value chain* di Debian verrà analizzata

oltre.

Fattori competitivi

Esistono numerose distribuzioni[23], alcune delle quali gratuite come quelle di Debian, altre commerciali che vendono servizi legati ai pacchetti. Alcune di queste usano i pacchetti Debian per le loro distribuzioni (ad esempio Linspire) e altre decidono di creare pacchetti secondo i propri criteri: è la scelta di RedHat e Novell, che usano i pacchetti di formato RPM.

Le distribuzioni si differenziano tra loro per costo (gratis o a pagamento), per la produzione da parte di un'impresa che fornisce assistenza e relative garanzie, per le piattaforme (desktop, server) o le architetture (Intel, Apple) disponibili e soprattutto per gli utenti a cui è destinata: principiante, esperto, utente che vuole un ambiente standard o al contrario ha esigenze specifiche (lingue particolari), applicazioni in ambiti specifici (musica, sicurezza informatica, eccetera).

Debian orienta il suo prodotto a un utente relativamente esperto, che vuole il massimo controllo su tutte le possibili scelte, anche dovendoci investire del tempo. Confrontata con distribution per principianti, passa per *difficile.* E' la scelta di preferenza dei programmatori e dei sistemisti, che non tollerano facilmente vincoli tecnici. Questo fatto assume una certa importanza, infatti i sistemisti, dovendo pacchettizzare un programma a loro necessario, lo faranno con la distribuzione di loro scelta, ampliando così l'offerta di pacchetti Debian e contribuendo alla creazione di un vantaggio per tutti gli utenti della distribuzione.

Il limite nell'adozione di Debian, specie su desktop e specie in clienti *corporate*, è forse il suo punto di forza altrove: la massima libertà di scelta[24]. Un servizio di assistenza informatica in outsourcing richiede ambienti il più possibile omogenei, quali quelli offerti da distribuzioni RedHat e Novell. Distribuzioni basate su Debian possono modulare o restringere la libertà di scelta in base alle necessità, per cui possono essere la base di scelta di system integrators indipendenti.

Le varie distribution competono, oltre che sui prodotti (pacchetti) offerti, anche sui servizi aggiuntivi, quali i programmi che rendono più agevole la configurazione o la manutenzione del sistema. Queste innovazioni vengono però rapidamente imitate, per cui comportano vantaggi di breve durata; quelli più durevoli sono relativi a scelte strategiche quali il tipo di pacchettizzazione, il numero di programmi disponibili, la qualità dei servizi di aggiornamento automatico e della sicurezza. Il sistema di gestione dei pacchetti usato da Debian gestisce la complessità di un sistema (in particolare le interdipendenze tra pacchetti) in modo piuttosto sofisticato ed efficace, e questo è un suo riconosciuto punto di forza.

Stratificazione

Come avviene per il kernel Linux, Debian mantiene simultaneamente diverse distribuzioni, che corrispondono a generazioni successive. La distribuzione consigliata per l'uso, denominata *stabile*, sulla quale vengono apportate solo modifiche che riguardano correzione di errori, prevalentemente di sicurezza.Una è in *test*, candidata a diventare stabile, con software più recente. La terza versione è *instabile*, in perenne sviluppo, dove avviene la sperimentazione. Le versioni stabili precedenti a

23 http://en.wikipedia.org/wiki/Comparison_of_Linux_distributions

24 Dell Computer, nel dubbio su quale distribution preinstallare, attende che il mercato decida:
"It's not that there are too many Linux desktop distributions," Dell said, "it's that they're all different, they all have supporters, and none of them can claim a majority of the market. [...] And, everyone keeps telling us that they want different distributions. So, our conclusion is to do them all and let the customer decide. [...] If the Linux desktops could converge at their cores, such a common platform would make it easier to support. Or, if there was a leading or highly preferred version that a majority of users would want, we'd preload it." (Vaughan-Nichols, 2006)

quella corrente vangono denominate *obsolete*.

Un altro aspetto di stratificazione si può rilevare nelle interdipendenze tra pacchetti che li ordinano gerarchicamente: infatti alcuni pacchetti richiedono la presenza di altri, introducendo la formazione di strati: alcuni programmi formano uno strato necessaro a quelli di strato superiore. Debian classifica i pacchetti secondo categorie che esplicitano questa stratificazione: lo strato più basso è costituito dai brogrammi *base*, quelli immediatamente sovrastanti al *kernel*, che vengono etichettati come *essential.*

Modularità

Il modulo in una distribuzione è il singolo pacchetto, che racchiude due elementi: il programma vero e proprio e una serie di informazioni strutturate e codificate che ne descrivono caratteristiche, funzionalità e interdipendenze con altri pacchetti. Questo secondo elemento è conoscenza pura codificata: queste informazioni non aggiungono nulla al codice e alla funzionalità dei programmi, ma descrivono in modo strutturato caratteristiche e relazioni con altri pacchetti. La conoscenza è la esplicitazione dell'esperienza degli esperti che svolgono l'attività di pacchettizzazione.

Il nucleo dell'operazione svolta da Debian è quindi quello di esplicitare la conoscenza implicita nascosta nei singoli programmi e nelle operazioni compiute dai sistemisti, codificarla, combinarla in una base di conoscenza accessibile da sistemi automatizzati e costruirvi attorno un sistema.

Questa operazione di *knowledge management* consente all'utente finale di cercare tra 15000 programmi diversi quello che gli serve, installarlo senza curarsi delle dipendenze con altri programmi, e analogamente disinstallarlo quando non serve più. Per comprendere la portata della semplificazione della compessità che questo processo introduce, basti pensare che un programma può appoggiarsi a decine tra librerie e ad altri programmi; a sua volta ogni libreria o programma può richiederne altri, e così via. L'installazione, per andare a buon fine richiede la risoluzione di tutte questa dipendenze: un processo che può essere lungo e penoso. La gestione automatizzata delle dipendenze lo rende veloce e completamente trasparente.

Cooperatività e coordinamento

Debian è un'associazione. La pacchettizzazione (packaging), cioè la creazione di pacchetti a partire dai programmi, è affidata ai *Debian developer*, che sono soci volontari membri di Debian. Chiunque può diventare developer a condizione che si sottoponga al superamento di un severo processo di verifica delle competenze tecniche, nonché della comprensione e adesione alle finalità del progetto[25]. Questo processo richiede un certo tempo sotto la supervisione di un developer anziano in funzione di *sponsor*, e si conclude su decisione di un *account manager.* I developer attualmente sono circa un migliaio.

La struttura dell'associazione, definita dalla *Debian constitution*[26], prevede che i developer eleggano i *projet leader* delle loro aree di interesse e un *leader* di tutto il progetto Debian con un complesso meccanismo di voto elettronico segreto. Il leader ha funzioni di rappresentanza, coordinamento e comunicazione. Assume compiti esecutivi in mancanza di responsabilità attribuite esplicitamente.

Altri organi di Debian sono un comitato tecnico che ha l'ultima parola sulle questioni tecniche e un segretario con diverse mansioni, anche esecutive.

Oltre alle attività tipiche dei developer, per il funzionamento del progetto sono richieste attività di correzione errori (*debugging*), traduzione, assistenza agli utenti, stesura di pagine di manuale e

25 Gabriella Coleman ha analizzato in prospettiva sociologica questo processo e il contesto etico da esso associato (Coleman 2005)

26 http://www.debian.org/devel/constitution

documentazione. A queste attività può contribuire qualsiasi utente Debian, anche non socio.

Un aspetto peculiare del progetto è il rapporto privilegiato con una organizzazione no-profit per tutto quanto riguarda la gestione della proprietà. Il progetto Debian si appoggia, per tutto ciò che riguarda supporto legale o la gestione di denaro e altre proprietà alla *Software in the Public Interest inc*[27] (SPI), pur dichiarandosi da essa indipendente. I developer sono soci di SPI.

Principi e sviluppo di strategie

Il progetto Debian ha sviluppato nel tempo un certo numero di documenti che descrivono finalità e strategie. Il processo aperto è finalizzato a creare un sistema che rispecchi le esigenze degli utenti (*The Debian Linux Manifesto"*[28]):

> *The Debian design process is open to ensure that the system is of the highest quality and that it reflects the needs of the user community.*

Il *Debian Social Contract*[29] con la comunità del software libero, assieme alle *Debian Free Software Guidelines*, definiscono le caratteristiche del progetto e del prodotto, indicando di fatto una strategia a lungo termine, orientata dagli utenti e dal free software. Il punto 4 del social contract risulta particolarmente illuminante su come la *mission* del progetto possa essere il creare un ambiente aperto di qualità che dia la massima flessibilità operativa all'utente.

> *4. Our priorities are our users and free software*
> *We will be guided by the needs of our users and the free software community. We will place their interests first in our priorities. We will support the needs of our users for operation in many different kinds of computing environments. We will not object to non-free works that are intended to be used on Debian systems, or attempt to charge a fee to people who create or use such works. We will allow others to create distributions containing both the Debian system and other works, without any fee from us. In furtherance of these goals, we will provide an integrated system of high-quality materials with no legal restrictions that would prevent such uses of the system.*

E' significativo che le *Debian Free Software Guide Lines* siano alla base della *Open Source Definition*[30] universalmente accettato dall'industria ICT come definizione del software *open source*.

Elementi comuni

La comunanza degli elementi dei vari progetti può essere riassunta nella seguente tabella.

	Internet	***WWW***	***GNU-Linux***	***Debian***
Strati, livelli	protocolli: IP, TCP, +singoli protocolli per i servizi	protocolli HTTP, HTML, linguaggio, altrri protocolli si appogiano aai precedenti	kernel, librerie, *core utils*, interfacce grafiche, servizi e applicazioni.	pacchetti, distribuzioni.

27 http://www.spi-inc.org
28 http://www.debian.org/devel/join/newmaint
29 http://www.debian.org/social_contract
30 http://www.opensource.org/docs/definition.php

	Internet	*WWW*	*GNU-Linux*	*Debian*
moduli	- singola rete, connessa in una rete di reti - singoli protocolli per i servizi	- singolo sito indipendente, ma connesso agli altri con link ipertestuali - singoli protocolli	- kernel: moduli - sistema: singoli programmi che interagiscono tra loro	- singoli pacchetti - singole *release*
cooperazione	IETF, Internet Service Providers, Domain name servers, industria.	W3C, produttori di software per server e client Web.	- OSDL (kernel), - liste e forum per kernel e altri progetti	- struttura del Debian Project - *mailing-lists* dei vari progetti
apertura e libertà di uso	connessione e ingresso in Internet.	costruzione e pubblicazione di un sito.	download e installazione liberi.	download e installazione liberi.
architettura aperta	definita da Request For Comment.	adesione a W3C (membership) e partecipazione alle *activity* e relativi gruppi di lavoro.	- proposta di correzioni, migliorie, nuovi moduli da parte di programmatori e utenti. - ingresso in OSDL Lab (Associate)	proposte dei debian developer e di utenti collaborativi
governance aperta	attraverso IETF, aperta ai singoli.	W3C, aperta a organizzazioni	- singoli progetti: rispettive strutture. - GNU: Free software foundation - OSDL: membership	-elezione del leader, -elezione dei project leaders.
principi	utente: *netiquette*, IETF: RFC 3935	W3C Mission, Goals	GNU: Philosophy of the GNU Project OSDL: mission	manifesto, social contract, constitution

Ciò che emerge dalla sintesi degli elementi comuni ai vari progetti può essere così sintetizzato:

1. Struttura **stratificata** e **modulare**. La ripartizione in *strati* orizzontali e *moduli* verticali avviene in base alle conoscenze necessarie per gestirli.
    1. I livelli o strati richiedono insiemi di conoscenze (competenze) diverse tra loro, per cui a un livello possono essere ignorate le conoscenze richieste ad un livello diverso.
    2. I moduli riguardano progetti o ambiti diversi che richiedono lo stesso livello di conoscenze che però si applicano in campi o settori diversi.
    3. Tra strati diversi e moduli che dovessero interfacciarsi devono essere definiti in modo rigoroso interfacce standard che definiscano ogni aspetto del dialogo tra moduli o strati.
    4. Sulla base di questi standard è possibile formalizzare ed automatizzare i processi di interazione dei moduli tra loro e degli utenti con il sistema, rendendo possibile la cooperazione.
2. Modello **cooperativo**: i soggetti che sviluppano e curano i vari moduli del sistema cooperano all'assemblaggio delle parti comuni di pertinenza del proprio strato, secondo le interfacce definite.
   Il coordinamento del processo cooperativo può avvenire su basi gerarchiche, con processi più o

meno democratici, basati su un consenso diffuso.

3. **Apertura** su tutti i livelli:
    1. *uso e distribuzione*: non vi sono barriere all'uso del prodotto o del progetto. Chiunque lo desideri può usarlo, senza limitazioni di scopo;
    2. *architettura del prodotto*: non vi sono barriere normative, tecniche, economiche o di accesso alle informazioni che definiscono il prodotto e il processo. Chiunque può proporre nuovi elementi o moduli o modifiche a quelli esistenti, o miglioramenti nell'architettura e nella struttura del prodotto o servizio e del processo di produzione;
    3. *governance*: la gestione del progetto è potenzialmente aperta a processi decisionali che coinvolgono tutti gli stakeholder. Chiunque abbia i titoli e il consenso può accedere alle cariche previste delle eventuali strutture di gestione e controllo. I processi di decisione generalmente si basano sulla ricerca del massimo consenso nel rispetto dei principi generali e della funzionalità e qualità tecnica.
4. **Principi condivisi.** Gli stakeholder condividono, con un grado crescente di adesione al crescere del coinvolgimento, alcuni *principi* base, una mission, delle linee guida, dei codici di comportamento o altri *foundation documents*. Questi documenti includono di solito non solo dei principi ideali molto spesso vicini a una *vision*, ma anche comportamenti operativi, direzioni future, risposte a minacce e opportunità presentate dall'ambiente che possono a buon titolo essere considerati almeno parte di una *strategia*.
   Sono dei principi ordinatori in una struttura che altrimenti lascia la massima autonomia ai soggetti responsabili dei singoli moduli. Un modo per lasciare che il caos produca il massimo effetto senza uscire dagli scopi prefissati.
5. **Creazione di un ambiente abilitante.** Un "e*nabling context*" viene definto (Von Krogh, Ichijo, Nonaka 2000) come "*what drives knowledge creation*" e viene associato al concetto di *Ba* come spazio organizzativo condiviso che consente l'interazione tra individui e lo scambio e creazione di conoscenza. Secondo questa visione il contesto è quello *interno* di una impresa, o comunque uno spazio relazionale. Nel nostro caso invece il contesto non è tanto un mezzo interno quanto un *ambiente* nel quale gli attori si muovono e che contribuiscono a costruire, e la sua costruzione è simultaneamente mezzo e fine dell'azione. Fine, in quanto l'ambiente è prodotto mai ultimato dell'interazione tra soggetti, cantiere perenne. Mezzo in quanto dato esterno, condizione fatta di un insieme di spazi, relazioni, infrastrutture, codice, progetti, informazioni, investimenti, imprese. Essendo aperto è abilitante nei confronti del mondo, non di uno specifico soggetto; in questo sta appunto almeno parte della sua efficacia.

   Per questo si tratta di un *enabling environment* in evoluzione, nel quale tutti i soggetti possono trovare delle opportunità di ulteriore sviluppo, innescando una competizione sia nell'estensione che nel perfezionamento del prodotto e offrendo delle opportunità di sfruttamento economico soprattutto sul piano dei servizi. L'ambiente abilitante non può essere creato da un singolo attore, sia per le sue dimensioni e il costo connesso, sia perchè l'efficacia è determinata dalla sua diffusione e dagli effetti di rete.

Altre osservazioni che non verranno approfondite:

1. le tecnologie semplici traggono vantaggio da quelle complesse e ne estraggono l'intelligenza semplificandole sul piano tecnico, sfruttandone l'efficacia ma non la complicazione. In questo contesto i modelli teorici si sono dimostrati più utili a spiegare più che a ispirare le tecnologie.
2. Filogenesi: questi ambienti si sono evoluti, sono *emersi* stratificando interventi successivi che non

necessariamente rispondono a una strategia deliberata, ma che traggono vantaggio da opportunità offerte dall'ambiente. In molti casi (Internet, Web) è stato necessario un intervento iniziale per catalizzare la reazione che una volta avviata ha innescato esternalità di rete che le hanno consentito di autoalimentarsi.

3. Ecosistema: dato il coordinamento debole questi ambienti hanno degli equilibri che possono essere fragili e facilmente alterabili. Alterazioni che ne compromettano i fondamenti di apertura e abbondanza di diversità possono essere una minaccia.

II - Internet e strategia nel contesto competitivo del free software

> *The Internet is not only opening up entirely new categories of applications, but it is irrevocably changing the ways in which software is sold and involving software technologies in contentious debates over many public policy issues including national sovereignty, national security and law enforcement, and limitations to information access.*
> *Messershmitt, Szypersky* – Software Ecosystem

> *Get the strategy and the management side right, and the software business can be like having a license to print money [...] but get the business model wrong, and [...] software companies can resemble dinosaurs futilely trying to escape the death grip of an ancient tarpit.*
> *Michael Cusumano* – The business of Software

Mercato del software: economie e ciclo di vita

La creazione del software è un processo articolato in cui solo alcune parti sono automatizzabili in modo industriale, altre restano artigianali, dipendenti da competenze altamente specifiche; altre ancora, quelle più legate all'innovazione, dipendono dalla creatività. Pertanto vi sono molti fattori di incertezza, in particolar modo in merito ai tempi: anche se non vi sono impianti la cui produttività va valutata in anticipo, le risorse devono essere allocate con grande cura.

Come osservato acutamente da Brooks (Brooks, 1995), nella produzione del software non si applicano più di tanto economie di scala, in particolare per la natura imprevedibile dei collaudi e della ricerca dei difetti (*debugging*).

> *The bearing of a child takes nine months, no matter how many women are assigned. Many software tasks have this characteristic because of the sequential nature of debugging.*

Hanno invece grande rilevanza le economie di apprendimento, che influenzano la capacità di produrre del software affidabile, il possesso e la completa padronanza delle tecnologie, la capacità di gestire progetti complessi e in costante evoluzione.

La differenziazione del prodotto ha un particolare ruolo, in quanto dipende fortemente dal segmento del mercato: alcuni programmi informatici hanno funzioni altamente definite se non standard, in altri segmenti la personalizzazione e l'adattamento del programma da parte dell'utente alle proprie esigenze è una caratteristica altamente desiderabile.

Il ciclo di vita di un software identifica gli aspetti essenziali della sua value chain (Messerschmitt,

Szyperski 2003):

> *The major steps in a lifecycle include analysis of user needs, development of the software (including architecture design, programming, testing, integration, and other functions), provisioning of the equipment and software in the user environment (including user training and often organizational changes), operation and administration of the software, and use. These steps form a value chain, where each step depends upon the successful completion of previous steps and adds value to them. The entire cycle is typically repeated multiple times, and the software supplier faces interesting challenges in managing the process of maintenance and upgrade of the coexisting version of the software while it is in use. Distributed applications often cross organizational boundaries, and this leads to many interesting management challenges, and also increases the importance of standardization.*

In seguito vedremo in dettaglio come la conoscenza ha un ruolo di grande rilievo nell'accrescere il valore di ogni singolo passo in questa filiera.

Il mercato del software è, come abbiamo già visto, dominato da un monopolista e da altre imprese che tendono ad alzare barriere di ingresso più che attraverso la gestione dell'innovazione soprattutto con i formati proprietari dei file dati[31], pratica chiamata *lock-in* e ben descritta dalla lapidaria strategia «*get their data, their hearts and minds will follow*» (Cusumano 2004).

In questo contesto il software f/oss nasce spesso da iniziative di singoli e cerca di trarre vantaggio dal massimo coinvolgimento di una comunità di utenti-sviluppatori soprattutto nella fase avanzata nello sviluppo del prodotto (Senyard, Michlmayer 2004):

> *We describe a three phase lifecycle for free software projects. The first phase is characterised by* ***closed development*** *performed by a small group or developer with much in common with traditional software development from which we have named the cathedral phase in reference to Eric Raymond. The second phase is a move from traditional development to* ***community based developmen****t which we have named the transition phase.*
>
> *Only projects with certain properties can successfully pass the transition phase. In order to operate successfully in the bazaar phase a number of activities must be completed. The first six are crucial, numbers seven to eleven are important and twelve is desirable:*
>
> *1. A prototype with plausible promise must have been created.*
> *2. The design of the prototype must be modular.*
> *3. The source code of the prototype must be available and workable (ie. compiles and executes).*
> *4. A community of users and developers must be attracted to the project.*
> *5. The project author must be motivated to manage the project or find a replacement.*
> *6. Project communication and contribution mechanisms must be in place.*
> *7. The scope of the project must be well defined.*
> *8. A coding standard or style must be established.*
> *9. Development versions of software must have short release cycles while user versions must be stable and consistent.*
> *10. A license must be chosen which is attractive to developers.*
> *11. A suitable management style must be selected.*
> *12. An appropriate amount of project documentation must exist.*

31 Infatti la modalità in cui i dati sono registrati sono determinati dal programma che li genera, per cui il produttore del programma ha facoltà di usare dei formati non standard nel salvataggio dei dati, e così facendo obbligare l'utente a utilizzare solo il proprio programma per leggerli. Questa prassi suscita crescente insofferenza soprattutto nelle pubbliche amministrazioni, che rivestono un ruolo cardine nei confronti del cittadino.

> *The final phase is where the project becomes a community based project and gains the associated advantages -- we have named this the **bazaar phase**. The lifecycle model proposed here gives a better understanding of the dynamics of free software and can assist in their success.*

Nella fase finale un certo numero di utenti possono desiderare di correre i rischi connessi a una maggiore instabilità del codice per avere l'ultima versione, per la quale è richiesta una attività di debugging intenso ma nella quale sarà possibile intervenire o richiedere modifiche. In questa fase, quella critica e fertile in cui un programma libero si differenzia maggiormante da uno proprietario, un progetto di free software si appoggia e confonde con i propri utenti avanzati (Neff, Stark 2002)

> *Open source projects display a responsiveness that has allowed users of products to be more directly involved in the design and the design-in-use of these products. Open source organizational forms challenge the traditional notion of boundaries between for-profit and non-profit organizations, as they often have more in common with social movements than traditional formal organizations.*

Il ruolo degli utenti-sviluppatori è particolarmente significativo in quanto si inserisce nella fase del debugging, che come abbiamo visto è quella dai risultati più imprevedibili nel ciclo di vita del programma. Questa strategia non può essere imitata dai produttori di software proprietario in quanto questi devono tenere segreto il codice sorgente, che è quindi indisponibile all'utente avanzato per correggerne gli errori.A conferma si vedano i tentativi di emulazione degli effetti benefici della condivisione senza la "liberazione" del codice sorgente: ad esempio l'iniziativa *Shared Source* di Microsoft[32].

Internet, value chain e strategie

Porter molto lucidamente osservava il ruolo che Internet avrebbe avuto nella value chain e nel far emergere le migliori strategie (Porter 2001):

> *The great paradox of the Internet is that its very benefits making information widely available; reducing the difficulty of purchasing, marketing, and distribution; allowing buyers and sellers to find and transact business with one another more easily also make it more difficult for companies to capture those benefits as profits.*

e inoltre

> *Many have argued that the Internet renders strategy obsolete. In reality, the opposite is true. Because the Internet tends to weaken industry profitability without providing proprietary operational advantages, it is more important than ever for companies to distinguish themselves through strategy. The winners will be those that view the Internet as a complement to, not a cannibal of, traditional ways of competing.*

Questi effetti si sono manifestati in modo particolarmente vantaggioso per il free software, che ha potuto sovrapporre la Rete alle reti sociali di sviluppo del software già esistenti nella tradizione del free oftware, e svilupparvi sopra strategie efficaci in ogni aspetto della value chain.

Sul piano della ***inbound logistics*** ha fornito i mezzi per mettere insieme in un unico "luogo" di produzione i programmatori interessati ad un dato progetto, favorendo l'incontro tra domanda e l'offerta: i programmatori più interessati al progetto si offrono di svilupparlo e manutenerlo. Questo

32 Microsoft Shared Source initiative http://www.microsoft.com/resources/sharedsource/

fa si che chi ha le conoscenze oppure è disposto ad investire per potenziarle ha ottime probabilità di impiegarle.

Senza entrare nel complesso tema delle motivazioni che spingono i programmatori a scrivere free software, un aspetto da non trascurare è l'effetto di pubblicità (*signaling*) che Internet può offrire: chi partecipa a un progetto di successo vedrà aumentare il proprio prestigio personale. Questo, oltre all'appagamento, permetterà di trovare piu facilmente un lavoro ben remunerato.

Anche lo ***sviluppo*** del free software si appoggia a Internet: contrariamente a quanto accade con il software proprietario, segreto industriale custodito nelle mura aziendali, i programmatori possono essere distanti tra loro e dal *repository* centrale del codice. Dato che il prodotto non è più localizzato insieme all'impresa, è molto più agevole anche per le imprese e non solo per i singoli o i gruppi collaborare e contribuire con risorse finanziare ed umane allo sviluppo e alla promozione di software, come avviene con gli Open Source Development Labs (OSDL) che sviluppano Linux. OSDL stima il costo annuo di un programmatore in 100'000 $/anno, per cui considera che per una impresa possa essere vantaggioso finanziare un progetto sviluppato negli OSDL rispetto allo sviluppo in proprio.

Dal momento che l'innovazione non deve più essere protetta dall'imitazione, l'ambito competitivo richiede strategie molto accurate. Data l'estrema dinamicità del mercato, essere i primi a offrire un servizio o un prodotto basato su una determinata innovazione può essere molto vantaggioso, ma data la facile imitabilità, più del prodotto conteranno la qualità, la differenziazione, la reputazione di chi lo fornisce.

Sul piano delle ***outbound logistics***, Internet ha favorito il consumo di free software molto di più di quanto abbia favorito il consumo di software proprietario, dato che non vi sono limitazioni alla copia e alla circolazione dei programmi.

Per i produttori di software proprietario, legati a processi di distribuzione tradizionali attraverso i *Distributors* e i *Value added resellers* secondo un modello gerarchico e unidirezionale (Wallace 2006), la facilità di copia è una minaccia e un punto di debolezza. Se vogliono distribuire attraverso la rete devono investire in meccanismi di protezione dalla copia costosi, spesso inefficaci e impopolari. Solo in fase di lancio la facilità di copia è stata usata come una opportunità anche dal software proprietario, per invogliare i consumatori a scegliere il proprio prodotto e poi costringerli ad acquistarlo (o ad acquistarne le versioni successive "mettendosi in regola").

Questo meccanismo ha sempre impedito o limitato la fungibilità di Internet come vettore di software proprietario. Al contrario la facilità di copia e diffusione è per il free software opportunità e un punto di forza.

La distribuzione senza restrizione via rete ha degli ulteriori innegabili vantaggi (Messershmitt Szyperski 2003):

- nell'acquisto, l'utente gode della immediata gratificazione evitando il ritardo tra acquisto e uso,
- mitiga i costi di avvio degli effetti di rete, favorendo la diffusione del software,
- velocizza la distribuzione e consente di ridurre i tempi tra una release e la successiva,
- facilita l'assistenza e la manutenzione degli aggiornamenti di servizio di upgrade.

Quest'ultimo punto ci introduce ai vantaggi nell'ambito ***service*** del modello porteriano della value chain. L'aggiornamento dei programmi, specie per motivi di sicurezza, è una caratteristica irrinunciabile. Quanto possa diventare problematica la distribuzione delle correzioni dei programmi lo ammette esplicitamente la *chief security officer* in Oracle (Davidson 2005).

Vendors may also need to provide fixes on multiple versions/platforms or bundle multiple security fixes together to minimize patching costs to customers, not to mention various testing on the products shipped to ensure the fix does not break anything else.
As an example, Oracle has done 78 fixes for a single vulnerability, which took more than five days to complete. We also release bundled fixes quarterly on dates tied to financial reporting calendars (e.g., many customers will not touch their production systems during quarter-end).
A two-line code change can take five minutes, but getting a fix into customers' hands in such a way that they will apply it takes way more than a few minutes.

Per Oracle, il *servizio* di *security assessment* fornito dalla comunità degli esperti di sicurezza è una minaccia, e la necessità di distribuire gli aggiornamenti di sicurezza una difficoltà.

Microsoft stessa è rimasta a lungo incerta se lasciare che anche le copie considerate illecite dei propri programmi potessero accedere via rete agli aggiornamenti di sicurezza oppure no, aumentando la percezione di insicurezza nei confronti dei proprio prodotti (e il livello complessivo di programmi vulnerabili).
Questo problema non si pone per il software libero: la distribuzione un aggiornamento di sicurezza non ha nulla di diverso dalla prima acquisizione del software, e in un modello che centra le sue strategie sulla rete sia la disponibilità del codice per ispezione sia la facilità di aggiornamento sono fattori di potenziamento della sicurezza, e non di minaccia.

Lungo tutta la value chain, il modello di sviluppo e diffusione del software proprietario non può quindi sviluppare delle strategie che riescano a sfruttare appieno i vantaggi che Internet offre, anzi, deve spesso considerarli come una minaccia.

Un esempio

Per riassumere i vantaggi rappresentatti della pacchettizzazione del software libero distribuito via Internet rispetto al software proprietario, seguiremo sommariamente i passi necessari per l'acquisizione (*procurement*) di un componente software per un sistema (un programma che svolge un determinato servizio, oppure una libreria software).

Distingueremo tra software proprietario distribuito tradizionalmente (su supporto o via rete con chiave di attivazione), software libero non pacchettizzato e software libero pacchettizzato. La seguente tabella riassume i principali passi che riguardano il procurement e le relative differenze.

	software proprietario	**software libero, non pacchettizzato**	**software libero pacchettizzato**
indagine di mercato	ricerca in Internet, su riviste di settore, portali, ecc	ricerca in Internet, su riviste di settore, portali, ecc	ricerca nel repository della distribution
richiesta di copia di prova o demo	ai singoli produttori	non necessaria	non necessaria
installazione di prova	a tempo determinato o con funzionalità limitata	controllo delle dipendenze, compilazione, installazione	con programma di gestione pacchetti; full-feature, senza limitazioni

	software proprietario	**software libero, non pacchettizzato**	**software libero pacchettizzato**
richiesta di informazioni supplementari o chiarimenti	al servizio di assistenza dell'impresa	alla comunità di utenti o al servizio assistenza di una ditta	alla comunità di utenti o al servizio assistenza di una ditta
richiesta preventivo e acquisto	all'impresa produttrice o al distributore. Verifica della accettabilità della licenza.	non necessario	non necessario
installazione	installazione,	controllo delle dipendenze, compilazione, installazione	installazione con programma di gestione dei pacchetti
modifiche o aggiornamenti	richiesta all'impresa produttrice	controllo delle dipendenze, compilazione, installazione	direttamente o via imprese di servizio
aggiornamento	al servizio dell'impresa produttrice, a pagamento	controllo delle dipendenze, compilazione, installazione	con programma di gestione pacchetti

Dall'analisi si evidenzia che il software libero *pacchettizzato* unisce ai vantaggi tipici del software proprietario (facilità di installazione) quelli del software libero.

III - Una value chain della conoscenza: il caso Debian

[...] the only way to protect ideas is to not communicate them, which in turn dramaticaly reduces their intrinsic value.

- Sawhney, Prandelli
Communities of Creation

Composizione di un sistema libero

Abbiamo già visto (figura 3) come nei sistemi operativi liberi una miriade di fornitori diversi contribuiscono a comporre i vari elementi del sistema, dal nucleo fino ai programmi utente, a differenza di quanto accade in un sistema proprietario in cui un fornitore principale offre il sistema operativo e definisce le interfacce alle quali altri fornitori si devono adattare per aggiungervi altri programmi.

Alcuni componenti provengono da singoli programmatori che li scrivono nel tempo libero, altri da gruppi coordinati, da realtà commerciali o no-profit che pagano dei programmatori professionisti da affiancare ai volontari. Sempre più frequentemente si vedono imprese che contribuiscono allo sviluppo del sistema aggiungendo componenti nuove o manutenendo quelli già esistenti.

Come abbiamo già visto, moduli affini o sostituibili tra loro comporranno degli strati, i quali si appoggeranno uno sull'altro mediante interfacce. Maggiore la standardizzazione delle interfacce, maggiore l'interoperabilità dei moduli all'interno di uno strato.

Il coordinamento dei programmatori attorno al progetto avviene spontaneamente attraverso i vincoli

tecnici imposti dalla gerarchia dei componenti (ad es. il *kernel* alla base impone alcuni vincoli che vanno rispettati a valle, e così via) e attraverso strumenti di coordinamento dei partecipanti ai progetti (principalmente mailing lists) attraverso i quali vengono prese delle decisioni.

Modularità e standardizzazione dei componenti

La compresenza di diversi moduli e programmi alternativi offre una maggiore scelta e utilità all'utente finale, seguendo di fatto una strategia di differenziazione. E stato osservato che vi è un rapporto tra innovazione, differenziazione e coordinamento (Devetag, Zaninotto 2001): la modularità aiuta a gestire la differenziazione e l'innovazione, ma richiede un coordinamento meno stretto per consentire la sperimentazione.

> *Modularity seems better suitable than other approaches when the adaptation to the demand variety and the speed of search in a given problem space (i.e., the rate of technological innovation) are more important than tight co-ordination. If this is the case, division of labour must occur in such a way that considerable space for local adjustment and experimentation is assured, unlike the case of a traditional assembly line.*

Nel *kernel*, come in altri progetti, la modularità è uno strumento per minimizzare la complessità rappresentata dalle interdipendenze tra componenti. L'obiettivo della modularizzazione è quindi:

- gestire l'incertezza e la variabilità nel problem solving,
- favorire la decomposizione del prodotto e del compito,
- aumentare la varietà del prodotto alla base di una strategia di differenziazione (Narduzzo Rossi 2005).

In considerazione delle problematiche di divisione del lavoro la modularità rappresenta non solo una esigenza tecnica, ma un modello di sviluppo.

> *We argue that modularity, which can be regarded as an innovative manufacturing paradigm for the design and the production of complex artifacts, is a key element in explaining the development and the success of many F/OSS projects, and it offers a comprehensive explanation of many key issues in F/OSS development, such as how division of labor takes place within developers, how coordination is achieved and how code testing and integration is deployed, how innovation occurs, and so on.* (Narduzzo Rossi 2005)

La teorizzazione spinta della modularizzazione talvolta fallisce, in quanto richede una conoscenza perfetta del sistema e in particolare delle interdipendenze. Laddove queste dipendenze non vengono correttamente valutate nascono malfunzionamenti. Questo è uno degli elementi di maggiore ciriticità nell'impostare e manutenere un progetto ampio con architettura modulare.

> *[...] Unfortunately, this neat description of modular design sometimes does not succeed; most of the times, after the integration of the independently developed modules, inconsistencies come up on and the system do es not work properly. The most common reason for this failure is the emergence of some interdependencies which were left out at the beginning, at the time of architecture and interfaces definition.* (Narduzzo Rossi 2005)

Va osservato che la suddivisione in moduli riusabili, o componenti, è una prassi consolidata nello sviluppo del software, e non rappresenta una novità del free software. Vi sono delle implicazioni di efficienza produttiva e di economie di conoscenza (Messershmitt, Szyperski 2003).

> *There is growing interest in software reuse and component software as a way of improving the productivity of development organizations, addressing increasing complexity, and improving quality. The idea is to assemble applications largely from existing components, which are modules created independently of any particular system context and constructed for multiple uses. To derive the full benefit, component software will require a marketplace for buying and selling components, which would in turn result in marked changes in the organization of the software industry.*

E' possibile vedere nell'offerta di software f/oss un rudimentale ma efficace mercato delle componenti software, anche se la ricchezza di diversità va a discapito del coordinamento. La definizione di una *architettura* basata su *interfacce aperte* è un elemento fondamentale per consentire l'integrazione e l'interoperabilità dei componenti.

> *Software architecture is the first stage of software implementation. It decomposes and partitions a software system to modules that can be dealt with somewhat independently. This modularity has significant business implications, including open interfaces and API's, system integration, and the composition of different applications and infrastructure.*

La modularizzazione non riguarda solo il software, ma i sistemi informatici composti da diversi programmi che forniscono i servizi. Grazie ad un corretto approccio modulare, chi deve realizzare un sistema completo in base a specifiche esigenze può comporre il sistema confidando nel fatto che tutti componenti vadano al loro posto senza intralciarsi a causa di interdipendenze non previste, che possano essere aggiornati quando serve in modo indolore, e che la composizione del sistema abbia i minori costi possibili in termini di ricerca, prova, installazione e messa in produzione dei componenti stessi. Per consentire il coordinamento, sono necessarie delle interfacce standard e la stratificazione. (Messersmitt, Szyperski 2003)

> *Coordination among firms is essential to getting operational software into the hands of the users. Standardization is one way in which software suppliers coordinate themselves, and it is growing in importance as distributed applications move across administrative and organizational boundaries. Driven largely by the rapid success of the Internet and distributed applications and new requirements for mixing different information media within a single application, the industry organization has been moving from a vertical to a horizontal structure, the latter consonant with an architecture concept called layering. Layering provides a relatively flexible way to partition an infrastructure to provide a wealth of services, with the possibility of adding additional features and capabilities by building layers on top of an existing infrastructure.*

Abbiamo già visto come stratificazione e modularizzazione facciano parte della struttura dei sistemi basati su architetture aperte. La standardizzazione viene raggiunta, come abbiamo visto, con la pacchettizzazione dei programmi che li trasforma in componenti standard e la loro integrazione in distribuzioni che costituiscono un *framework* coerente e omogeneo, rispondente a sua volta a criteri standard.

Distributions e pacchetti nella value chain del software libero

Ogni programma informatico in una *distribution* GNU/Linux viene *pacchettizzato* per poter essere poi impiegato come componente standard nella costruzione di un sistema informatico

complesso. Questo consente di stratificare la complessità totale del sistema definendo delle interfacce tra strati e permettendo di trascurare le complessità di livello inferiore per poter affrontare quella degli strati superiori. La conoscenza viene organizzata e cristallizzata per essere condivisa, senza semplificazioni e perdita di contenuti.

Usare una distribution consente di comprimere i costi di ricerca dei programmi-componenti, di integrazione del programma nel sistema, di manutenzione e aggiornamento.

Analizziamo in dettaglio come avviene il passaggio da codice "grezzo" a pacchetto standard, con particolare attenzione al ruolo che la conoscenza svolge e in che modo questa venga esplicitata e incorporata nel prodotto lungo la supply chain (Figura 4).

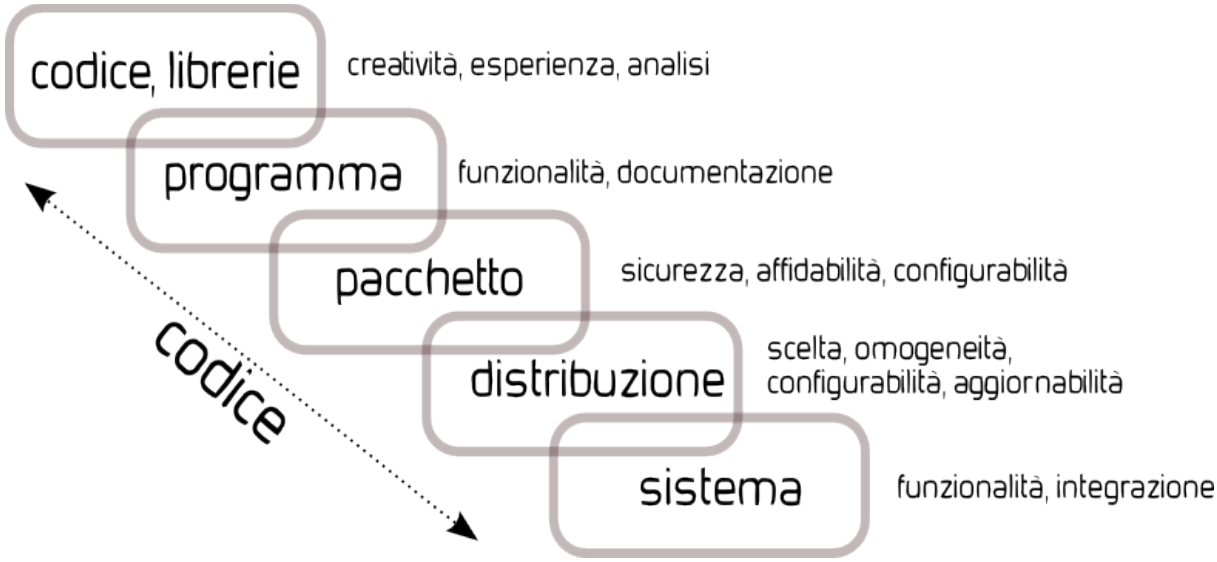

Figura 4: dal codice al sistema

Codice

Il programmatore, o un gruppo di programmatori, scrive del codice sorgente per realizzare un progetto. Può impiegare codice già pronto (riuso) che include in quello proprio con o senza modifiche. Deve essere sicuro di poter riusare codice altrui: le limitazioni possono essere sia di natura tecnica (ad esempio dipendenze) che legale, ad esempio codice coperto da una licenza che non ne consente l'impiego, incompatibile o da brevetti. Il programmatore, oltre alla competenza necessaria a scrivere il programma, deve conoscere le limitazioni tecniche e legali conseguenti all'inclusione di altro codice.
Un programma non è quasi mai un prodotto finito, statico, ma viene costantemente modificato per correggerne errori o malfunzionamenti (manutenzione) o per l'aggiunta di nuove funzionalità (evoluzione).
Dato che l'esigenza degli utenti è quella di avere un programma stabile e affidabile e non in continuo mutamento, vi è quasi sempre una versione consolidata e stabile del programma, manutenuta ma non in evoluzione, alla quale si affianca una in evoluzione, o di sviluppo.

Programma.

Prima che il codice possa diventare un programma funzionante bisogna che questo si integri con il sistema che lo ospita, ma non solo. Deve essere approntato anche tutto l'apparato che lo rende utilizzabile: documentazione (manuali, tutorial, esempi di impiego), siti web di riferimento per gli aggiornamenti, mailing list per il supporto. Devono essere descritti i file di configurazione e le modalità di interazione con il sistema, con altri eventuali programmi e con l'utente.
Questo compito viene svolto non solo dal programmatore ma da utenti esperti, con l'aiuto di utenti di prova (*beta testers*) che possono anche contribuire direttamente alla stesura della documentazione. La qualità della documentazione è essenziale al successo del programma. Vi sono siti web (*repository*) appositamente concepiti per conservare il codice e la documentazione e fornire il supporto allo sviluppo programmi f/oss[33].

Pacchetto.

Il programma viene usualmente distribuito in forma di codice sorgente. Per essere reso funzionante deve essere compilato, o tradotto in forma binaria per una determinata architettura. Come abbiamo visto, questo compito può essere svolto dal sistemista ogni volta che decide di installare un dato programma, oppure da un terzo che fornisce un *pacchetto* che contiene: l'eseguibile già compilato, una descrizione dello stesso e delle sue funzioni e la formalizzazione delle conoscenze relative delle eventuali dipendenze richieste dal programma, incompatibilità, alternative, programmi che esso eventualmente rimpiazza. Una volta introdotto il pacchetto nel sistema un insieme di programmi di gestione consentono la ricerca, la gestione automatizzata delle installazioni e la risoluzione automatica delle dipendenze.
Il valore aggiunto dal pacchetto è enorme: risparmia al sistemista del tempo in ricerca, del tempo della compilazione, e molto tempo nella eventuale risoluzione delle dipendenze, processo faticoso, lento e frustrante.

Distribuzione.

Insiemi di pacchetti compatibili tra loro costituiscono una distribuzione, e raccolgono quindi versioni successive dei programmi a grandi salti discreti. Necessariamente incorporano una serie di ulteriori scelte che offriranno all'utente un tutt'uno coerente e compatto, cercando di livellare eventuali differenze. Un altro elemento che emerge con la distribuzione, spesso fattore di differenziazione tra loro, è l'insieme di programmi per l'installazione, la mantuenzione e l'aggiornamento, in particolare quelli automatici di sicurezza. Richiede una infrastruttura online di server sempre raggiungibili (*repository*) con tutti i pacchetti aggiornati.

Sistema

A livello di sistema l'utente (sistemista) può effettuare le scelte che desidera tra i vari pacchetti che offrono le funzionalità richieste. Un sistema che usi esclusivamente programmi pacchettizzati è coerente: tutti i pacchetti si rifanno a un modello comune (ad esempio nella dislocazione dei file di configurazione) e integrato, in modo che i programmi sono compatibili tra loro e interoperabili. L'interfaccia di gestione è comune e omogenea.

Il passaggio da una diversità di codici a un unico sistema ha visto l'intervento di diversi specialisti, ciascuno dei quali ha introdotto le sue conoscenze e le ha codificate.
In molti casi (tra cui Debian) viene garantita anche l'indipendenza dall'hardware (portabilità) in quanto i pacchetti sono disponibili per le principali architetture hardware esistenti. All'utente finale

33 tra i più noti repository: www.sourceforge.net, www.savannah.gnu.org

viene offerto il massimo grado di libertà, minimizzando tutti i possibili attriti.

La value chain di Debian

Mentre la supply chain da *codice* a *sistema* è piuttosto generica e si applica a diversi produttori di diverse distribution, il modo specifico in cui questo processo si svolge può essere diverso per i vari produttori, in particolare per quanto riguarda le attività di supporto. Applichiamo a Debian, di cui abbiamo già analizzato la struttura di governance, l'analisi porteriana della value chain.

A - attività primarie

1) **Logistica in entrata**:

Gli input sono principalmente i programmi, in forma di codice sorgente, reperibili dalla rete, principalmente dai *repository* centralizzati. In certi casi i programmatori autori dei programmi stessi sono coloro che curano la gestione operativa.

Trattandosi di un prodotto con un elevato contenuto di conoscenza, vanno considerati come input assai rilevanti, oltre alle conoscenze dei programmatori e dei *developer*, le segnalazioni degli utenti finali, di solito competenti, e le osservazioni raccolte attraverso forum e mailing-list.

2) **Gestione operativa**: trasformazione degli input in output, packaging

Questa fase riguarda la formazione dei pacchetti: richiede lo studio e la codifica delle dipendenze, il controllo che non vi siano incompatibilità.

La pacchettizzazione, svolta dai *Debian developer* richiede la conoscenza del programma, del sistema nel suo complesso e delle particolarità del sistema Debian. Comporta la scelta e l'armonizzazione dei vari componenti in accordo con gli altri developer.

La debolezza in questa fase sta nei lunghi tempi di consolidamento di una distribution stabile, per cui gli utenti possono dover attendere a lungo che le nuove versioni dei programmi che usano compaiano nella distribution.

Le distribution in lavorazione simultaneamente in questa fase sono più di una: la distribution *stable*, che richiede solo minori correzioni e aggiornamenti di sicurezza, quella *unstable* destinata a diventare la prossima stable e quella *testing* nella quale si svolgono i collaudi prima di accettare i pacchetti nella unstable.

Il processo di trasformazione è aperto in tutte e tre i prodotti: qualsiasi utente può installare qualsiasi delle tre e partecipare con segnalazioni al processo di sviluppo.

3) **Logistica in uscita**: distribuzione, marketing

Come abbiamo già visto, questa fase si svolge via Internet: sia la distribuzione che il "marketing" avvengono attraverso gli annunci nelle liste specializzate e il *tam-tam* nei vari siti e riviste specializzate. In occasione della uscita di una nuova release spesso le riviste la allegano in CD-ROM, con reciproci vantaggi per Debian che vede distribuito su supporto fisico il proprio prodotto, e delle riviste, più richieste.

In un'ottica di supply chain integrata, possiamo vedere come una distribuzione Debian possa essere a sua volta l'input per altre *distribution* che ne sfruttano la pacchettizzazione confezionanado un prodotto diverso. Questo è quanto avviene con una distribution più attenta all'utente inesperto, come è il caso di Ubuntu, che fornisce supporto commerciale. Ubuntu dichiara che: «*Debian is "the*

rock upon which Ubuntu is built"»[34], sottolineando sia la stabilità di Debian, sia il fatto di averci costruito sopra.

E' ragionevole pensare che Debian svolgerà sempre di più un ruolo di fornitore di pacchetti per altre distribution sia commerciali che gratuite, che baseranno la loro strategia sull'offerta di servizi, lasciando a Debian la pacchettizzazione. Debian è nella posizione per costituire un *enabling environment* per altri soggetti, anche commerciali, che contribuiscono al progetto con reciproco vantaggio.

4) **Servizi**: postvendita, garanzia.

Debian fornisce direttamente e gratuitamente ai propri utenti un servizio di aggiornamento delle distribution, sia per la manutenzione di sicurezza che per le nuove release dei pacchetti. I servizi sono erogati attraverso siti appositi accessibili attraverso i programmi di aggiornamento automatico. La qualità ed affidabilità di questo servizio è un punto di forza di Debian.

Anche in questo caso, Debian offre al mercato un ambiente nel quale imprese sul territorio possono fornire servizi di assistenza finale allàutente e integrazione sistemistica. Frequentemente piccole imprese di servizi Debian come base nei propri sistemi. Debian non fornisce direttamente un servizio strutturato di assistenza all'utente, tuttavia sono disponibili le mailing list per l'aiuto, i forum, la documentazione online, in cui le informazioni vengono fornite dai soggetti stessi che vi partecipano. Questi, nello svolgere l'attività di assistenza, diventano soggetti attivi nello scambio della conoscenza, *knowledge brokers,* e nel coinvolgimento dei neofiti nella comunità ovvero nel reclutamento, fornendo un raccordo con la logistica in entrata (Sowe, Stamelos, Angelis 2005):

> *Knowledge brokers not only help individuals manage and extract valued software knowledge from software repositories, but also help OSS projects to engage in a discourse and colearning experience with their user communities.*

B - attività di supporto:

1) Gestione **approvvigionamenti**:

I programmatori autori di programmi free software possono anche non essere a conoscenza di essere dei fornitori di Debian. Tuttavia l'attività di pacchettizzazione può mettere in evidenza errori o incongruenze nei programmi che vengono corrette e segnalate dal Debian developer al programmatore, che potrà decidere di integrarle nel programma. Gli utenti che collaborano al progetto di solito lo fanno attraverso le apposite mailing-lists.

2) Gestione **tecnologia**: R&S, supporto alla produzione

La ricerca si articola in due attività: lo sviluppo di nuovi programmi di integrazione del sistema, e quest'attività viene svolta come in relazione a un qualsiasi altro pacchetto, e quella relativa alle scelte tecniche e politiche sul prodotto. Queste avvengono attraverso i già visti meccanismi di governance. La modalità con cui questo avviene, liste di discussione e votazioni con maggioranze qualificate, può generare lunghe discussioni e decisioni sofferte, che possono ripercuotersi negativamente sui tempi di rilascio.

3) Risorse **umane**: selezione personale, formazione, carriere, remunerazione, organizzazione.

Come si è visto il reclutamento dei *developer* avviene in modo molto rigoroso, in quanto il compito comporta grandi responsabilità, ma implica un elevato prestigio sociale, un implicito *signaling* di competenze che può ripercuotersi favorevolmente sulla remunerazione professionale extra Debian.

34 Dal sito di Ubuntu: http://www.ubuntu.com/ubuntu/relationship?highlight=%28debian%29

4) **Infrastrutturale**: management, organizzazione e controllo.

Come già visto, la struttura di coordinamento è gerarchica, centrata sulle competenze e sul consenso. Il processo di coordinamento esecutivo avviene attraverso dei leader eletti.

Le vicende che possono avere delle implicazioni economiche o che riguardano proprietà sono esternalizzate a una società no-profit: software in the public interest (SPI). E' singolare che la gestione della proprietà non sia considerata una competenza centrale.

Conclusione dell'analisi:

La distribution Debian è centrata sulla qualità in termini si affidabilità del prodotto e qualità del servizio di aggiornamento automatico, particolarmente per gli aggiornamenti di sicurezza.

Il profilo risultante è compatibile con una strategia di differenziazione, cioè con la scelta di offrire un'ampia gamma dei programmi già pacchettizzati disponibili per un considerevole numero di architetture diverse.

Minor rilevanza viene data alla tempestività dell'uscita di nuovi rilasci con prodotti aggiornati: la nuova release esce quando è matura. Questo può essere visto come un punto di debolezza. Tuttavia l'utente ha la possibilità di scegliere anche una distribution in corso di lavorazione, rinunciando alla stabilità. In questo modo il *trade-off* tra affidabilità e aggiornamento all'ultima versione è una scelta lasciata all'utente.

La centralità della conoscenza nel processo di *value creation* è riscontrabile in ogni fase della catena.

Knowledge value chain e ambiente abilitante

L'attività complessa svolta da Debian è centrata sulla conoscenza e può essere vista come una *knowledge value chain* (Lee, Yang 2000): *acquisizione* delle competenze dei developer che vengono codificate; sviluppo di eventuale *innovazione* per migliorare il prodotto; *protezione* del prodotto da scelte errate e di coerenza con la vision del progetto; *integrazione* delle conoscenze in una unico contesto e *disseminazione* interna ed esterna al progetto.

Nell'ambito della comunità free software lo scambio di conoscenza non è solo il fine dell'organizzazione, ma è alla base del processo che costituisce la comunità stessa e la alimenta (Sowe, Karoulis, Stamelos 2005):

> *The F/OSSD context is a "*symbiotic cognitive system*", where the community learns from its participants, and each individual learns from the community. Ongoing interactions and activities are a means of acquiring valuable knowledge that is worth archiving. Participants learn advanced and basic concepts associated with collaborative software development. Novice and lurkers alike learn by browsing the knowledge base. The legitimate peripheral participant learns as an apprentice, observing and participating. [...] Thus, as a bricoleur, the learner is continuously involved in a discovery, trying to put order where there is disorder, structure where there is no structure, fitting his or her knowledge with the project's knowledge base.*

In particolare l'aspetto di innovazione della conoscenza può aiutarci a integrare questa operazione di knowledge management nell'ambito dell'ambiente abilitante, nel quale i contributi singoli rientrano organicamente in una rete organizzativa e contribuiscono a formarla. (Lee, Yang 2000):

> *Organizational knowledge innovation, threfore, shuld be understood as a process that "organizationanally" amplifies the knowledge created by individuals and crystallizes it as a part of the knowledge network of the organization.*

Caratteristica della value chain che coinvolge un *enabling environment* è di fornire con il proprio output l'input per un altro enabling environment, di cui il primo viene a costituire uno strato: Internet è alla base del Web, fornisce una infrastruttura per le communities che sviluppano *free software* e anche un mezzo assai adatto per la sua distribuzione. Debian raccoglie free software e lo trasforma in modo adatto perché costituisca un enabling environment per imprese che sviluppano prodotti e servizi di IT, come Ubuntu che usa il sistema a pacchetti di Debian per la sua distribuzione commerciale.
Ciascuno di questi elementi singolarmente e il loro inseme complessivo possono essere visti come un ambiente abilitante. Il software free/open source e le communities che lo supportano, incluse le imprese, nel suo complesso può essere pensato come una rete che "fornisce" non solo un prodotto, ma un intero ecosistema, una supply chain della conoscenza a beneficio dell'utente finale e di altri attori del mercato che desiderano avvalersene.

Sarà di particolare interesse verificare quali saranno le interazioni con il mercato tradizionale del software e il sempre maggiore coinvolgimento con imprese che sviluppavano software tradizionale.

Conclusioni

Quindi il Tao fa vivere,
la virtù alleva, fa crescere,
sviluppa, completa, matura,
nutre, ripara.
Le fa vivere ma non le tiene come sue
opera ma nulla s'aspetta,
le fa crescere ma non le governa.
Questa è la misteriosa virtù.

- Lao Tsu, Tao Te Ching[35]*, 51*

Abbiamo analizzato le caratteristiche di alcuni dei progetti alla base della rivoluzione telematica, e ne abbiamo evidenziato alcuni aspetti fondamentali: stratificazione, modularità, apertura a livello di prodotto, di progetto e di governance. Abbiamo ipotizzato che questa struttura, che abbiamo chiamato ambiente abilitante, sia alla base del successo di progetti la cui scala vada oltre la portata di un singolo attore. Il free software sembra essere nel suo complesso uno di questi ambienti, e abbiamo visto come il progetto Debian porti nell'ambiente organizzazione e conoscenza strutturata.

Nel modello così abbozzato, le iniziative di successo si sviluppano ed attraggono maggiori risorse, potenziandosi e contribuendo all'innovazione dell'ambiente, quelle che falliscono si spengono senza lasciare grande traccia e senza coinvolgere tutto l'ambiente nel caso di scelte strategiche sbagliate. Le imprese non vedono gravare sulle proprie spalle tutti i rischi dell'innovazione ma trovano opportunità per offrire servizi su componenti standard sviluppati collaborando tra loro e con comunità di individui. Le imprese con maggiori risorse possono investire in innovazione contribuendo all'ambiente senza accollarsi tutti i costi, rinunciando però al monopolio, anche temporaneo, sull'innovazione del prodotto, e concentrandosi sulle strategie di servizi.

Open sembra una parola destinata a rimanere. Non solo *open source*, termine efficace per il marketing del *free software*, ma *open knowledge*, percorso di sviluppo cooperativo di tecnologie che concorrono a sviluppare ambienti abilitanti in cascata che ampliano le possibilità di individui, comunità e imprese e ne amplificano le opportunità. Questo scenario apre una prospettiva innovativa: oltre l'impresa vuota e l'outsourcing della ricerca e sviluppo[36] si affaccia una "impresa non-impresa" basata sul progetto che contribuisce a fondare un ambiente competitivo nel quale convive con le imprese, insieme contribuendo ad alimentarlo. Organizzazioni formali ed informali, imprese e community insieme gestiscono l'innovazione e creano valore strutturandosi e cooperando; realizzano prodotti, perseguono obiettivi e insieme delineano e seguono strategie operando in un mercato. Attraverso il mercato lasciano emergere i prodotti e le strategie migliori e, rinunciando a controllare in modo sistematico e proprietario il frutto della propria creatività con una organizzazione burocratica ed una visione mercantile, sollecitano la partecipazione esterna attraverso meccanismi di condivisione della conoscenza e spirito di sfida.

35 Da LiberLiber: http://www.liberliber.it/
36 http://innocentive.com

Riferimenti

- Harald T. Alvestrand, *RFC 3935, A Mission Statement for the IETF*, October 2004
- Juan-José Amor-Iglesias, Jesùs M. Gonzàlez-Barahona, Gregorio Robles-Martìnez, and Israel Herràiz-Tabernero *Measuring Libre Software Using Debian 3.1 (Sarge) as A Case Study: Preliminary Results*; ; http://www.upgrade-cepis.org/issues/2005/3/up6-3Amor.pdf
- Jonathan Bennett, *Berners-Lee calls for Net neutrality* CNET News.com May 23, 2006; http://news.com.com/2100-1036_3-6075472.html
- Lorenzo Beussi, *Analysing the technological history of the Open Source Phenomenon: Stories from the Free Software Evolution,* Working paper 2005, http://opensource.mit.edu/papers/benussi.pdf
- Scott Bradner, ed, *RFC2418: IETF Working Group Guidelines and Procedures,* September 1998; http://rfc.net/rfc2418.html
- Steve Bratt, *Worldwide Participation in the World Wide Web Consortium, W3C,* May 2006, http://www.w3.org/2005/02/W3C-Global-Focus.html
- Fredrick P. Brooks, *The mythical Man-Month, Essays on Software Engineering,* Addison-Wesley 1975 (2nd ed. 1995)
- Gabriella E. Coleman, *Three Ethical Moments in Debian,* 2005, SSRN: http://ssrn.com/abstract=805287
- Michael Cusumano, *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad,* Free Press, 2004
- Mary Ann Davidson, *When security researchers become the problem*, News.com July 27, 2005 http://news.com.com/When+security+researchers+become+the+problem/2010-1071_3-5807074.html
- Giovanna Devetag, Enrico Zaninotto *The imperfect hiding : some introductory concepts and preliminary issues on modularity*. ROCK Working Papers. 2001 http://eprints.biblio.unitn.it/archive/00000388/01/DZModularity.pdf
- Pino Fondati, *Il Pinguino corre, ma la strada è ancora in salita*, Il Sole 24 Ore, 12 aprile 2006, http://www.ilsole24ore.com/fc?cmd=art&codid=20.0.1899767783&chId=30&artType=Articolo&DocRulesView=Libero
- Jesús M. González-Barahona, Miguel A. Ortuño Pérez, Pedro de las Heras Quirós, José Centeno González, Vicente Matellán Olivera. *Counting potatoes: the size of Debian 2.2* http://opensource.mit.edu/papers/counting-potatoes.html
- Toshihiko Hayashi, *Fostering globally accessible and affordable ICTs*; International Telecommunication Union, 2003; http://www.itu.int/osg/spu/visions/papers/accesspaper.pdf
- Juho Lindman, *Effects of Open Source Software on the Business Patterns of Software Industry*, Helsinki School of Economics Department of Management, 2004, http://opensource.mit.edu/papers/lindman.pdf
- Ching-Chyi Lee, Jie Yang, *Knowledge value chain*, Journal of management development, vol 19, n.9, 2000
- Narduzzo A., Rossi A., *The Role of Modularity in Free/Open Source Software Development*, in S. Koch (ed by), 2005, http://opensource.mit.edu/papers/narduzzorossi.pdf
- Gregorio Robles, Jesus M. Gonzalez-Barahona, Martin Michlmayr *Evolution of Volunteer Participation in Libre Software Projects: Evidence from Debian*,, http://www.cyrius.com/publications/robles_barahona_michlmayr-evolution_participation.pdf
- Dan MacLean, ed., *Internet Governance: A Grand Collaboration*, United Nations ICT Task Force, 2004
- David Messerschmitt,Clemens Szyperski, *Software Ecosystem – Understanding An Indispensable Technology and Industry,* MIT Press, 2003
- Eben Moglen, *Die Gedanken Sind Frei*, Berlin June 10th 2004 – Wizard of OS 3 speech

http://emoglen.law.columbia.edu/publications/berlin-keynote.html

- Gina Neff, David Stark; *Permanently Beta: Responsive Organization in the Internet Era*, September 2002 http://opensource.mit.edu/papers/neff-stark.pdf
- Anthony Senyard, Martin Michlmayr, *How to Have a Successful Free Software Project,* Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC'04), 2004; http://opensource.mit.edu/papers/senyardmichlmay.pdf
- Sulayman K. Sowe, Athanasis Karoulis, Ioannis Stamelos, *A Constructivist View of Knowledge Management in Open Source Virtual Communities,* in Antonio Dias de Figueiredo, Ana Paula Afonso, *Managing Learning in Virtual Settings: The Role of Context*, Idea group, 2005. http://sweng.csd.auth.gr/~sksowe/Publicat/IDBK001-C16_285-303_.pdf
- Sulayman K. Sowe, Ioannis Stamelos, Lefteris Angelis; *Identification of Knowledge Brokers that Yeld Software Engineering Knowledge in OSS projects,* Information and Software Technology, In press, 2006, avaliable online.
- Michael E. Porter, *Strategy And the Internet*; Harvard Business Review, March 2001
- Michael E. Porter, Mark R. Kramer, *The Competitive Advantage of Corporate Philanthropy*, Harvard Business Review, December 2002
- Giorgio Pellicelli, *Strategie d'impresa,* Università Bocconi Editore, 2005
- Mohanbir Sawhney, Emanuele Prandelli, *Communities of Creation, managing distributed innovation in turbulent markets,* California Management Review, vol.42, n.4, summer 2000
- Richard A. Spinello, *The case against Microsoft: An ethical perspective*, Business Ethics: A European Review, Volume 12, Issue 2, Page 116-132, Apr. 2003
- Stephen Shankland IDC: *Linux is now mainstream* ; CNET News.com December 16, 2004 http://news.zdnet.co.uk/software/linuxunix/0,39020390,39181356,00.htm
- Jim Turley, *Embedded systems survey: Operating systems up for grabs*, Embedded Systems Design 2005, http://www.embedded.com/showArticle.jhtml?articleID=163700590
- Steven J. Vaughan-Nichols *Mr. Dell opens up about Desktop Linux*, 7 Marzo 2006 http://www.desktoplinux.com/news/NS3822185143.html
- Von Krogh, G., Ichijo, K., & Nonaka, I. *Enabling knowledge creation: how to unlock the mystery of tacit knowledge and release the power of innovation.,* Oxford University Press, USA, 2000
- Greg Wallace, *Open Source: Changing the Enterprise Software Supply Chain for Good*, Linux.sys-con, Feb 2006, http://linux.sys-con.com/read/173425.htm