consip

quaderni consip

ricerche, analisi, prospettive

XII [2007]

Il fenomeno Open Source: considerazioni di ordine tecnico, economico e legale

MEF Ministero dell'Economia e delle Finanze

consip

quaderni consip

ricerche, analisi, prospettive

Elio Luzzi, Gabriele Mezzacapo

XII [2007]

Il fenomeno Open Source: considerazioni di ordine tecnico, economico e legale

MEF Ministero dell'Economia e delle Finanze

Un ringraziamento a:

Leonardo Bertini, Matteo Cavallini, Sebastiano Lomuscio, Maria Raffaella Migliorini, Monica Neri, Luca Nicoletti, Emilia Occhiuto e Elisabetta Traisci per il loro prezioso contributo al lavoro.

Indice

1. Prefazione

Sono molto lieto di questa iniziativa editoriale di Consip che mi pare significativa e importante. Significativa perché dimostra l'attenzione di Consip verso un fenomeno, quello del software libero, che molti altri soggetti considerano ancora immaturo e non degno di attenzione, in ragione della sua inadeguatezza alle esigenze del mercato. Importante perché le indicazioni del volume appaiono utili ai fini della promozione del software libero nella pubblica amministrazione centrale e periferica.
Colgo la gradita opportunità di questa prefazione - un grande onore per me - per proporre una breve osservazione.

Il software libero rappresenta una fortunata occasione per l'economia italiana. Infatti, secondo calcoli approssimati che ho sviluppato partendo dai dati di mercato del rapporto ASSINFORM, l'importo complessivo delle licenze software versato ad aziende straniere è dell'ordine di due miliardi di euro all'anno, circa dieci volte il costo della politica secondo i calcoli di un'autorevole fonte. Mi domando spesso per quale ragione dei costi della politica si parli molto spesso mentre l'argomento dei costi del software non viene mai sfiorato né nei decreti associati alle varie finanziarie, né nel dibattito politico.
Ma vi è una seconda ragione, ancora più importante della prima, per la quale il software libero rappresenta un'opportunità per il sistema economico del nostro Paese. Recentemente Google ha lanciato un nuovo motore di ricerca, chiamato "Krugle", che è stato dedicato esclusivamente al codice sorgente e alla documentazione dei prodotti contenuti nei molti "repository" come Source Forge. L'indice coprirà circa 100 milioni di pagine, per l'equivalente di alcune migliaia di miliardi di byte e quindi molti miliardi di istruzioni di codice. Questo patrimonio di conoscenze sta cambiando le regole della competizione internazionale. Molti dei prodotti industriali che nasceranno nei prossimi anni avranno un corpo piccolissimo, come quello di un'ape, un processore da pochi dollari e un po' di elettronica ausiliaria, e una grande anima, l'intelligenza dell'alveare, costituita da un enorme volume di software. Pochi atomi e molti bit.
La necessità di enormi investimenti per realizzare quella grande anima ha reso sinora impossibile l'ingresso nel settore delle piccole e medie imprese industriali. Ma con l'avvento del software libero la realtà sta cambiando, perché ora quella grande intelligenza collettiva, rappresentata da miliardi di istruzioni liberamente e gratuitamente distribuite sulla Rete, è disponibile anche ai più piccoli operatori.

Poiché la realtà industriale italiana è caratterizzata, molto più di quanto avviene in altri paesi, da un grande numero di piccoli operatori, il nostro paese più di altri potrà trarre beneficio dall'avvento del software libero. Quell'enorme patrimonio di conoscenze scientifiche e tecnologiche che è costituito dalle librerie del software libero, può divenire il combustibile per riavviare il motore tecnologico dell'industria nazionale.

Prof. Angelo Raffaele Meo
Politecnico di Torino
Presidente della Commissione Nazionale per l'Open Source
Ministero per le Riforme e le Innovazioni nella PA

2. Introduzione

L'Open Source è un fenomeno sociale di portata epocale che negli ultimi venti anni ha profondamente cambiato il mondo del software offrendo alle società industriali opportunità di crescita non solo tecnologica ma anche economica ed organizzativa. Il successo e la forte diffusione di prodotti software quali Linux ed Apache ha spinto molti studiosi a rappresentare il fenomeno Open Source come la naturale evoluzione dell'industria del software ancor oggi basata fortemente sull'approccio proprietario del software.

L'Open Source è stato interpretato erroneamente come un movimento anti-Microsoft, sovente considerato un modello di sviluppo capace di produrre solamente software di qualità amatoriale e molte volte confuso con il software freeware, gratuito ma non "libero"; negli ultimi anni è stato ampiamente rivalutato da aziende e istituzioni nonché dalla letteratura accademica, che lo ha reso oggetto di studi approfonditi in svariati campi.

Sono molti gli interrogativi che sono emersi nei diversi ambiti di applicazione ed a cui non sempre è stata trovata una risposta univoca, sul come e perché tale fenomeno di portata mondiale sia riuscito a trovare una così ampia diffusione e partecipazione. La ricerca economica ha cercato di spiegare la percorribilità di un modello apparentemente assurdo dal punto di vista economico, provando a capire le motivazioni delle aziende che lo hanno scelto per le loro attività; la psicologia sociale si è chiesta invece quali siano le ragioni che spingono un così alto numero di programmatori a contribuire in forma volontaria ai progetti Open Source; alcuni autori hanno cercato di spiegare il fenomeno dal punto di vista giuridico, interrogandosi sul modo in cui esso ri-articola l'idea di proprietà intellettuale.

Il presente quaderno, vista la già copiosa letteratura disponibile sull'Open Source, si pone come scopo quello di rilevare, per i diversi aspetti tecnologici, giuridici, economico-sociali ed organizzativi, elementi e spunti di riflessione rappresentando sia un momento di diffusione culturale nel contesto di riferimento sia un supporto al lettore che intende adottare tale approccio per lo sviluppo di nuove iniziative.

Il primo capitolo descrive e analizza il fenomeno dell'Open Source ripercorrendo la sua evoluzione sin dalle origini, approfondendo il contesto sociale ed i diversi approcci filosofici sostenuti dai padri fondatori ed analizzando le modalità di estensione anche nei paesi in via di sviluppo.

Il secondo capitolo descrive e compara, da un punto di vista metodologico, i modelli di valutazione delle piattaforme software Open Source identificando gli elementi principali che occorre tenere presente in sede di analisi di un prodotto; vengono affrontati i temi dello sviluppo del software aperto sulla base delle metodologie disponibili tra le quali viene approfondita quella del patchwork prototype e della migrazione delle piattaforme software Desktop, comparando i modelli esistenti.

Il terzo capitolo effettua un breve inquadramento giuridico della fattispecie e delle sue caratteristiche più salienti che la contraddistinguono dal software di tipo proprietario, per procedere alla definizione di concetti quali proprietà intellettuale e paternità morale del software e soprattutto il problema delle garanzie. Viene inoltre fatta una breve disamina, senza dubbio non esaustiva, di alcune fra le più diffuse licenze di tipo Open Source, rinviando alla lettura dei rispettivi testi completi, per una maggiore conoscenza della disciplina che sorregge i rapporti con gli utilizzatori.

Il quarto capitolo esamina i modelli organizzativi e di business che si sono sviluppati e diffusi sul mercato a seguito della rapida diffusione dell'Open Source; in particolare viene analizzato il tema della community descrivendo i suoi aspetti organizzativi e sociali, entrando nel merito delle attività svolte ed analizzando gli elementi tecnici, le best practice ed i modelli che rendono economicamente profittevole un progetto di sviluppo di un progetto.

Il quinto capitolo affronta il tema del ruolo della Pubblica Amministrazione come attore del mercato, delle community e regolamentatore del sistema; viene analizzata la situazione europea dei progetti in essere, sulla base del ruolo svolto, delle scelte tecnologiche e delle motivazioni attese; sono descritte le iniziative correnti e future, le architetture tecnologiche ed applicative, e le motivazioni che sono alla base dell'adozione di software libero nell'ambito del contesto Consip/Mef.
L'ultimo capitolo da una visione prospettica del fenomeno Open Source sulla base di un'analisi generale del mercato. Sono delineati alcuni trend di diffusione di prodotti a codice aperto e di evoluzione dei modelli organizzativi presenti a livello mondiale nei diversi settori economici ed in particolare nel settore pubblico.

3. L'Open Source come fenomeno sociale

Il software è diventato un elemento vitale e imprescindibile nella società moderna; il software sostiene il business attraverso la gestione, la condivisione e la cooperazione di informazioni disponibili in tempo reale superando barriere e confini geografici, permettendo alle organizzazioni pubbliche e private di avere il controllo di molteplici attività e servizi e di analizzare fenomeni. L'importanza del software cresce nel tempo e tale crescita impone una serie di interrogativi a cui le organizzazioni ed i singoli sono chiamati a dare una risposta: che tipo di software utilizzare, quando utilizzarlo e quali motivazioni, economiche, tecnologiche, sociali e politiche devono essere alla base della decisione di utilizzare un software rispetto agli altri.
Nella valutazione delle scelte, in alternativa al tradizionale modello "commerciale" o "proprietario" si contrappone quello "Open Source". I modelli tecnicamente si contrappongono per l'accessibilità del codice sorgente che nel primo caso non è consentita a coloro che ne acquistano la licenza d'uso. Il modello Open Source invece pone il suo elemento caratterizzante proprio sul fatto che il codice sorgente del programma software è disponibile (appunto è open), modificabile e copiabile. Da fenomeno meramente tecnologico nato in ambito accademico, grazie allo sviluppo della rete internet, l'Open Source è divenuto un modello di partecipazione cooperativa worldwide organizzata in forme costitutive identificate con il termine di "Community" che, pur nella loro diversità di modelli, basano la loro identità sulla libertà di partecipazione quale driver di continua innovazione e di sviluppo.

3.1 Il Software Open Source come cultura: the hacker culture

Un valido punto di partenza dal quale procedere per vagliare il processo di produzione del sapere può essere fornito dall'analisi delle linee guida del **movimento hacker**, osservando come nasce la condivisione di conoscenza (inizialmente da intendersi prettamente informatica) in rete. È opportuno dedicare particolare attenzione alle origini di questo movimento ed al suo sviluppo, sia a livello storico sia a livello etico, in quanto gli hacker hanno il merito di aver inventato una vera e propria "pratica" ovvero un modo ben preciso e peculiare di fruizione della macchina informatica.
La storia del movimento ha inizio negli anni '50, negli Stati Uniti, attorno a due poli universitari fondamentali: l'Università di Berkeley e quella di Stanford, entrambe con sede in California. Altro punto nevralgico è da ricercarsi nel MIT (*Massachusett Institute of Technology*).

Il momento era particolarmente vivace dal punto di vista ideologico con la diffusione di idee a carattere libertario, il fiorire di intelligenze acute, interessate all'ambito informatico e un mutamento tecnologico sempre presente; tutti elementi indispensabili per costituire terreno fertile per la nascita di un nuovo movimento: giovane, un po' ribelle, tecnologicamente istruito.

Si può tentare una definizione di tale figura attingendo dagli studi di Manuel Castells. Il termine "hacker", come rilevato dall'autore, viene spesso confuso, a torto, con quello del "cracker", o del pirata informatico: «*Gli hacker non sono ciò che raccontano i media. Non sono esperti informatici irrequieti, ansiosi di crackare codici, penetrare illegalmente nei sistemi o portare il caos nel traffico informatico. Quelli che si comportano così sono chiamati "crackers"* ».

In realtà, l'hacker è colui che cerca di risolvere tendenzialmente un problema informatico che ha incontrato durante la sua "esplorazione" personale, un bug in cui si è imbattuto per caso. È colui che tende ad accrescere in maniera costante la propria conoscenza tecnologica per aver maggior padronanza di tecniche e linguaggi, in modo da essere in grado, poi, di ampliare o modificare a sua discrezione il software.

La storia degli hacker ebbe inizio dunque nelle università e, in particolare, tra i primi gruppi di studenti che si riunirono intorno all'Homebrew Computer Club di Berkeley, costituitosi nel 1973. Il loro sogno, più vicino a un'utopia forse, era quello di rendere l'informatica libera e aperta a tutti: a livello puramente tecnico, essi erano caratterizzati dal loro stesso scarso interesse per linguaggi di programmazione di alto livello e troppo distanti dal funzionamento di base. Questo ha dimostrato quanto l'ideologia fosse fortemente radicata in valori libertari e comunitari, in quanto si puntava ad un'informazione che doveva essere libera e condivisibile da tutti coloro che lo volessero, senza ostacoli o difficoltà tecniche.

Con la nascita e la diffusione di ARPANET, nella seconda metà degli anni '60, la Rete divenne il veicolo principale per la diffusione delle idee e delle prospettive hacker e non solo. Ci fu, in breve tempo, la creazione di una vera e propria comunità di rete, una tribù legata da ideali e comuni obiettivi, in grado di mettersi in comunicazione attraverso la nuova tecnologia delle BBS, come si diceva ideata proprio all'interno dell'Homebrew Computer Club.

Osservando attentamente il movimento hacker si può notare come in realtà abbia vissuto due grossi momenti fondamentali: una prima fase, di tipo "etico", e una seconda fase, per lo più "pragmatica". Per il primo passo è essenziale l'operato di Richard Marshall Stallman, mentre per la seconda si concentrerà l'attenzione su Eric Raymond.
Al di là del più generico desiderio di condividere informazioni e sapere, il movimento hacker si è contraddistinto per il progetto del "free software", in altre parole quel software il cui codice sorgente è liberamente distribuito in modo che chiunque possa ampliarlo o modificarlo e a sua volta ridistribuirlo. Nel tracciare le linee-guida del profilo "etico" di questo nuova idea di diffusione informatica, è centrale, come si è detto, la figura di Stallman.

3.2 Le interpretazioni di Stallman, Perens e Raymond

Richard M. Stallman definisce il concetto di software libero: «Il free software è libero, non gratuito: dovete pensare al concetto di libertà di parola, piuttosto che a quello di birra gratis». Ciò significa che il software può essere usato, copiato, modificato e distribuito da chiunque, gratis o a pagamento, a discrezione di colui che compie l'azione di diffonderlo.
Stallman e colleghi stavano lavorando ad un nuovo sistema operativo pensato perché tutti potessero accedere liberamente e sempre, quindi privo di qualunque forma di restrizione. Emerse così un grosso problema etico: il software proprietario era in quel momento indispensabile, ma ciò significava dover cedere alle leggi di un mercato che voleva prescindere dalla condivisione del software e più in generale dalle idee creative che avrebbero potuto liberamente contribuire a migliorarlo. Così Stallman si impegnò nel ricercare un'alternativa alla proprietà intellettuale sul software. Egli condusse questa ricerca impegnandosi in modo totalizzante, assumendola come una "missione".
Nel 1984, Stallman non volendo sottomettersi ai quei cambiamenti non in linea con la sua impostazione filosofica lasciò il MIT e si dedicò allo sviluppo di un nuovo sistema operativo compatibile con Unix, che era allora il più diffuso. Chiamò il sistema operativo GNU e lo rese completamente libero. Il codice sorgente era accessibile e disponibile in rete. La filosofia di Richard Stallman è sempre stata mantenuta dal suo ideatore senza cedimenti, grazie anche ai toni notevolmente estremistici e radicalmente ideologici della stessa.

La visione estrema della cooperazione come unica via possibile e auspicabile per la creazione di software libero ha condotto Stallman a diverse iniziative, tra cui la fondazione della FSF (Free Software Foundation) e l'ideazione di una nuova forma di diritto d'autore, definito **copyleft.**

Alla base di GNU e della sua più importante applicazione, Emacs, Stallman decise di porre un'organizzazione non-profit basata su contributi di tipo volontario, sotto forma di lavoro e quindi la Free Software Foundation si è fatta esponente e punto di riferimento del progetto "etico" cui Stallman ha dato vita, per l'appunto quello che ha alla base il software libero. La FSF, per Stallman, ha incarnato una nuova possibilità di ricreare le condizioni di lavoro che vigevano al MIT, dove si assisteva e si contribuiva ad un processo di creazione costantemente attivo intorno ad un prodotto passibile di modifica in ogni momento.

Altro fondamentale risultato dell'opera di teorizzazione compiuta da Stallman è il concetto di copyleft. Come egli stesso afferma, il copyleft è un termine che mira a «capovolgere il diritto d'autore». Il copyleft, tradotto liberamente dallo stesso suo creatore con "permesso d'autore", è ciò che sancisce il pubblico dominio del software libero e il diritto-dovere di distribuirlo. Il copyleft fa riferimento alla licenza che viene associata al software libero, ovvero la GNU GPL (GNU General Public License), che sancisce che chiunque può fare qualsiasi cosa col prodotto, salvo limitare la stessa libertà agli altri utenti.

Se da un lato Stallman rappresenta la parte filosofica del movimento che gira attorno a Linux e al software libero, Bruce Perens ed Eric Raymond rappresentano la capacità di conciliare la filosofia con le regole del mercato. Bruce Perens è l'ideatore della definizione **Open Source** ed è fondamentalmente il primo che ha fatto capire che anche il software a codice aperto può essere una fonte di business.

L'intuizione di Perens fu che se i clienti non avessero dovuto pagare per la versione di base del software, sarebbero stati disposti a pagare la sua personalizzazione; questo avrebbe potuto creare una diversa economia cosicché le aziende, anziché fare profitti sulle licenze, avrebbero potuto guadagnare sui servizi collegati alla manutenzione e sviluppo del software. Questo avrebbe motivato la creazione di imprese dove il capitale più importante è quello intellettuale, cioè la capacità di risolvere problemi, che nel mondo dei computer si chiama programmazione.

Il vero successo dell'Open Source è la rete che permette a sviluppatori sparsi nel mondo di cooperare costituendo una organizzazione anarchica di migliaia di sviluppatori che produce software.

Eric Raymond, rappresenta la faccia più controversa e discussa del mondo Open Source; è l'autore de **La Cattedrale e il Bazaar**, del 1997 testo che analizza, da un punto di vista personale, la modalità di sviluppo Open Source, attuale *maintainer* del **Jargon file**, autore di porzioni di fetchmail, Ncurses e Emacs.

Nel testo viene descritto un nuovo modello di sviluppo, il cui esempio più famoso ed efficace è la modalità di costruzione del *Kernel Linux*. L'autore per verificare le proprie ipotesi decide di utilizzare lo sviluppo collaborativo per il programma *fetchmail* e nel saggio viene descritta la genesi e lo sviluppo del progetto analizzando le interazioni con gli altri sviluppatori e i vantaggi rispetto al modello classico

Il saggio descrive in sostanza due contrapposte modalità di sviluppo del software libero:

- nel modello a **Cattedrale** il programma viene realizzato da un numero limitato di "esperti" che provvedono a scrivere il codice in quasi totale isolamento. Il progetto ha una suddivisione gerarchica molto stretta e ogni sviluppatore si preoccupa della sua piccola parte di codice. Le revisioni si susseguono con relativa lentezza e gli sviluppatori cercano di rilasciare programmi il più possibile completi e senza bug. Il programma *Emacs*, il *GCC* e molti altri programmi si basano su questo modello di sviluppo.
- Nel modello a **Bazaar** il codice sorgente della revisione in sviluppo è disponibile liberamente, gli utenti possono interagire con gli sviluppatori e se ne hanno le capacità, possono modificare e integrare il codice. Lo sviluppo è decentralizzato e non esiste una rigida suddivisione dei compiti; un programmatore di buona volontà può modificare e integrare qualsiasi parte del codice. In sostanza lo sviluppo è molto più anarchico e libero, da qui il nome di modello a Bazaar. Il *Kernel Linux* e molti programmi utilizzano questo nuovo modello di sviluppo associativo.

Il modello a Cattedrale è un modello tipico delle aziende commerciali. Queste normalmente non rilasciano il codice sorgente e una nuova revisione del programma può richiedere anni, mentre il modello a Bazaar è un modello che si è molto diffuso nell'ambiente del software libero dato che consente a ogni utente di essere un beta tester dei programmi. Lo stesso utente può modificare e integrare il programma se lo desidera.

Questo consente un rapporto stretto tra utilizzatori e programmatori, un rapporto paritario che ben si adatta alla filosofia del software libero. Consente inoltre un attento controllo del codice in modo da eliminare rapidamente la maggior parte dei bug, cosa impossibile per un software prodotto con la modalità a Cattedrale dove solo un numero limitato di persone lavora sul codice.
La modalità a Cattedrale è la stessa metodologia di sviluppo che viene utilizzata dagli sviluppatori delle enciclopedie commerciali dove un numero limitato di esperti si preoccupa di compilare tutte le voci. La modalità a Bazaar invece è quella utilizzata dalla **Wikipedia** dove ogni lettore se lo desidera può diventare anche scrittore di nuovi contenuti o integrare e migliorare quelli esistenti , e dove la verifica delle "voci" è gestita dagli stessi utenti e da alcuni amministratori.
La contrapposizione tra Stallman e Raymond è ben rappresentata attraverso i rispettivi termini proposti **Free Software**[1] dal primo e **Open Source**[2] dal secondo. Per Stallman, Free Software è un'azione politica che pone i principi della libertà sopra ogni altra cosa. Esso è completamente diverso dall'Open Source, che è puramente un modo pratico di scrivere il software e non solleva il punto della libertà degli utenti. In pratica, l'Open Source non è una ideologia ma pragmatismo.

Raymond ha fondato la **Open Source Iniziative** (OSI), sorella della FSF di Stallman, ove considera la disponibilità del codice sorgente dei programmi informatici, la base di un sostenibile sviluppo globale. L'OSI è una associazione *no profit* che si dedica all'utilizzo ed alla promozione della definizione Open Source (Codice Aperto) per il bene della comunità, in particolare attraverso la certificazione del marchio **OSI Certified Open Source Software** (**software a codice aperto certificato OSI**).[3]

[1] *Un software libero è un software rilasciato con una licenza che permette a chiunque di utilizzarlo e che ne incoraggia lo studio, le modifiche e la redistribuzione; per le sue caratteristiche, si contrappone al software proprietario (Wikipedia). Secondo Richard Stallman e la Free Software Foundation da lui fondata, un software per poter essere definito libero deve garantire quattro "libertà fondamentali": Libertà di eseguire il programma per qualsiasi scopo (chiamata "libertà 0"), Libertà di studiare il programma e modificarlo ("libertà 1"), Libertà di copiare il programma in modo da aiutare il prossimo ("libertà 2"), Libertà di migliorare il programma e di distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio ("libertà 3").*

[2] *Requisito fondamentale dell'Open Source è la disponibilità del codice sorgente (Wikipedia). Due elementi caratterizzanti sono il rispetto di alcune convenzioni di distribuzione (per esempio potrebbe essere vietata la redistribuzione o la modifica del codice sorgente e quindi un software distribuito sotto tale licenza non può essere software libero) e la discrezionalità dell'Open Source Initiative (OSI) sul riconoscimento di licenze "ufficiali" Open Source (che non rispondono alle quattro "libertà fondamentali" della FSF).*

[3] *Nel prosieguo del documento, il termine " Open Source" o "OS" è inteso nel significato più ampio del termine ovvero* **Free,** *Libre Open Source Software (FLOSS) cioè l'insieme dei due approcci freesoftware ed opensource.*

3.3 Diffusione, impatti ed opportunità dell'Open Source nei Paesi in via di Sviluppo

Nell'affrontare il problema della penetrazione informatica nei Paesi in via di Sviluppo non si può non introdurre il concetto, ormai attuale in ogni Paese, del **digital divide**. Con questo termine si fa riferimento al divario presente tra chi può e chi non può accedere (possedendone o meno le capacità) alle tecnologie informatiche. Questo è un fenomeno tipico non soltanto dei Paesi in via di Sviluppo, ma presente, anche se in forma meno accentuata, nei paesi che hanno visto una maggiore penetrazione delle tecnologie informatiche.

La questione del divario tecnologico può essere quindi suddivisa in due problematiche distinte:

- un ***digital divide interno***, riguardante le differenze tra individui all'interno dello stesso paese. Questo tipo di divide si articola in diversi fattori. Il maggiore o minore accesso alle tecnologie informatiche e alla rete Internet varia in base al reddito (ma il divario è più ampio nei Paesi in via di Sviluppo), alla dislocazione territoriale, soprattutto nei termini di un forte scarto tra zone urbane e rurali, al livello di istruzione, all'età (in merito alla quale gli anziani sono generalmente svantaggiati), al sesso (in questo caso il gap dipende più dalle tendenze socio-culturali di ciascun paese che da ragioni legate al reddito) e all'handicap (che pone il problema dell'accessibilità degli artefatti informatici);
- un ***digital divide internazionale***, riguardante le differenze in termini percentuali di penetrazione delle tecnologie informatiche e di alfabetizzazione informatica tra diversi paesi e generalizzabile, in maniera imprecisa, nel divario tra paesi del nord e del sud del mondo. Si distinguono a questo proposito tre tipologie di paesi: *paesi leader*, caratterizzati da una infrastruttura informatica matura e da alti livelli di interconnettività; *paesi adottanti*, caratterizzati da bassa interconnettività in termini di servizi web ma da un buon livello di interconnessione infrastrutturale; *paesi ritardatari*, in cui il livello di interconnessione è talmente basso da non permettere un incremento del livello di interconnettività.

In entrambi i casi si configura come un problema non meramente economico, ma riguardante anche gli ambiti della formazione e dell'insegnamento. Consapevoli del fatto che gran parte dei Paesi in via di sviluppo deve ancora affrontare il problema ben più urgente della soddisfazione dei bisogni primari (la fame, l'acqua, la sanità, l'istruzione, l'elettricità), la questione del divario tecnologico è di importanza cruciale in uno scenario globalmente interconnesso e in presenza di un grande cambiamento all'interno di molte società riguardante il campo produttivo e organizzativo, in buona parte indotto dall'introduzione delle tecnologie informatiche.

Parlare delle "nuove" tecnologie informatiche significa parlare della capacità di queste di immagazzinare, catalogare, selezionare, trattare enormi quantità di informazioni, con evidente influenza in qualsiasi campo dell'agire umano. Il divario tecnologico non si riassume semplicemente nell'accesso fisico alle tecnologie, in termini di presenza di computer e connettività disponibili per l'utilizzo da parte di tutti ed in tutte le aree geografiche, ma anche in termini di acquisizione di competenze nell'uso degli strumenti.

Negli ultimi anni, alcuni governi hanno dimostrato sempre maggiore interesse nei confronti dell'adozione di software Open Source; numerosi investimenti sono stati fatti dal settore privato, sia da operatori "puri" del software libero, quali le distribuzioni Red Hat, Suse e TurboLinux, sia da produttori di hardware e sviluppatori di DBMS, quali Hewlett-Packard, IBM, Sun, LG, Oracle e SAP.
In Brasile, su iniziativa del Presidente Luiz Ignacio Lula da Silva, è stata varata una raccomandazione che prevede l'adozione di software Open Source in sostituzione al software proprietario da parte delle dipendenze del settore pubblico. Nel 2003, a questo proposito, è stato pubblicato un rapporto dal titolo "Free Software Implementation Guidelines" mirante a fornire indicazioni sulle modalità della migrazione. Nella città di Sao Paulo, il *Telecenter Project* mira alla creazione all'interno della città di un centinaio di centri telematici, ognuno dei quali dovrebbe essere in grado di servire circa tremila utenti. La "Electronic Government Iniziative" alla quale sono affidati i compiti di creazione e gestione dei centri, persegue un abbassamento dei costi derivanti dall'uso di piattaforme Open Source (Debian GNU/Linux e Linux Terminal Server) in una infrastruttura hardware che prevede la presenza in ogni centro di un server

e di venti terminali privi di hard disk. In questo caso, l'uso di tecnologia Free/Open Source consente, oltre al taglio dei costi per l'acquisizione di licenze proprietarie, un abbassamento dei prezzi complessivi attraverso il taglio delle spese per l'hardware. Caratteristica interessante del progetto è l'attività di formazione e assistenza. Sono infatti previsti corsi di alfabetizzazione informatica di base, formazione nell'ambito della grafica computerizzata, nella creazione di sistemi web e l'istituzione di un help desk tecnico. È incoraggiata la produzione di articoli sulla vita della comunità di riferimento per il centro e la creazione di materiale di supporto per l'apprendimento e l'attivazione di nuove iniziative occupazionali.

Numerose iniziative si stanno affacciando anche nel sud-est asiatico.

In India, la *Linux India Initiative*, promossa dal governo, si sta occupando di organizzare gruppi di studio sul software libero/Open Source, centri di ricerca e gruppi di supporto all'adozione. Il Goa Schools Computer Project (GSCP), nato nel 1996, si è occupato della messa in funzione, all'interno delle scuole, di circa quattrocento computer riciclati dotandoli di software completamente libero/Open Source, risparmiando sull'acquisto delle licenze e potendo godere del supporto dei Linux User Group locali.

In Malesia è stato istituito un centro di riferimento per l'Open Source con compiti di assistenza e supporto all'adozione, alla certificazione del software e alla formazione. È inoltre in previsione lo stanziamento di fondi per compagnie informatiche indipendenti che si occupano di software a codice aperto.

Nelle Filippine il governo si è impegnato nella messa a punto di pacchetti software Open Source disegnati sulla base delle esigenze locali. Tra questi, una versione semplificata di Linux chiamata Bayanihan Linux. In Thailandia, il Ministro per l'ITC ha esplicitamente incoraggiato l'uso di software libero/Open Source nelle agenzie amministrative.
Cina, Giappone e Sud Corea hanno annunciato una collaborazione, da attuarsi insieme al settore privato, mirante allo sviluppo di sistemi software embedded basati su Linux per l'uso all'interno di cellulari, macchine digitali e sistemi di navigazione satellitare.

In Sudafrica, attraverso un approccio similare a quello adottato dall'Unione Europea, si raccomanda, mediante il rapporto governativo "Using Open Source Software in the South African Government", l'adozione di criteri economici e di merito nella scelta tra software libero e proprietario, suggerendo di preferire l'utilizzo di software libero quando il bilancio tra i vantaggi e gli svantaggi diretti tra software proprietario e aperto sia da considerarsi equiparabile. Si sottolineano inoltre i vantaggi del software Open Source in merito all'indipendenza dai fornitori e in relazione alla maggiore sicurezza.

In Tajikistan, l'assenza di traduzioni delle distribuzioni proprietarie nel linguaggio locale ha stimolato la nascita di un progetto, guidato da Khujand Computer Technologies, teso allo sviluppo di una traduzione della distribuzione Mandrake Linux che potrà in seguito essere utilizzata nelle scuole, nelle amministrazioni e nel settore privato.
In generale, i vantaggi derivanti dall'adozione di software Open Source nei paesi in via di sviluppo sono più consistenti che nei paesi ad alta informatizzazione. In particolare:

- il costo per i prodotti software è notevolmente più basso, consentendo una maggiore focalizzazione sulla personalizzazione ed adattamento alle esigenze locali, sulla formazione e sui servizi. Lo stesso vale nel caso di finanziamenti da parte di enti pubblici o privati nei confronti di questi paesi: il trasferimento tecnologico può essere effettuato a minore costo. Nei paesi a maggiore penetrazione informatica il TCO (Total Cost of Ownership) è costituito da un'alta percentuale di costi di formazione del personale; quindi nel caso di una migrazione verso una nuova tecnologia software (in questo caso verso un modello OS) spesso questo fattore è decisivo nel determinare la scelta di continuare a rivolgersi allo stesso fornitore proprietario e a non effettuare la migrazione. Nei paesi in via di sviluppo questo fattore ha meno peso nel rapporto tra software a codice aperto e software proprietario, in quanto i costi di formazione, nei casi diffusissimi di analfabetismo informatico, sarebbero uguali per entrambe le soluzioni.
- la tecnologia software è direttamente accessibile e non soltanto utilizzabile. L'accesso al software è molto vantaggioso, sia in termini educativi che, soprattutto, nel riutilizzo del software Open Source per la creazione di prodotti derivati senza il pagamento dei diritti intellettuali;

- è relativamente facile legarsi a dei circuiti internazionali di sviluppatori, beneficiando di collaborazioni su progetti specifici e garantendo una maggiore visibilità a livello mondiale delle proprie aziende ICT locali. All'interno, la nascita di "Linux User Groups" locali, caratteristici in presenza di una buona diffusione del software OS, costituirebbe un' ulteriore opportunità per creare dei legami di tipo locale/globale tra associazioni, singoli, privati e istituzioni e potrebbe rivelarsi importante a scopo formativo;
- è fattibile la creazione di prodotti studiati per le esigenze di quei territori dove, per questioni demografiche, ci sia scarsa scalabilità dei costi di sviluppo software. Questo fenomeno si verifica spesso nel caso delle traduzioni. Il caso Tajikistan ne è un esempio lampante;
- diventa più efficace l'azione contro l'obsolescenza dei programmi. Nel mondo del software proprietario le aziende tendono ad aggiornare frequentemente i prodotti smettendo di supportare le versioni precedenti e richiedendo maggiori ed aggiornate risorse hardware. Questo rappresenta generalmente un problema per molti paesi in via di sviluppo spesso dotati di macchine superate. Grazie all'apertura del codice sorgente, il porting di un programma su di una macchina datata è sempre possibile. In merito a questo, è importante citare il progetto RULE (Run Up to-date Linux Everywhere), che mira allo sviluppo di una distribuzione basata su Red Hat pensata per essere particolarmente leggera in modo di potersi adattare alle macchine con bassa CPU e poca RAM;
- viene data maggiore autonomia nel settore economico. Nel rapporto della Digital Opportunity Task Force [4] si sottolinea l'importanza del settore ICT nel creare opportunità di sviluppo, attraverso un migliorato accesso alle informazioni e più ampie capacità di comunicazione. Diversamente dal software proprietario, assai legato alle istanze del mercato, il software libero garantisce una maggiore indipendenza, stimolando maggiormente lo sviluppo di soluzioni più focalizzate sulla crescita locale, consentendo la nascita di un settore ICT autonomo e riducendo le spese di importazione di software dai paesi leader.

[4] La Digital Opportunity Task Force (DOT Force) è un team di lavoro istituito all'interno del "Okinawa Charter on Global Information Society " che fu emesso al "Kyushu-Okinawa Summit" in July 2000, al fine di inquadrare la cooperazione GS sul tema ICT in un contesto internazionale più ampio.

4. Sviluppo e valutazione delle piattaforme Open Source

La filosofia alla base dell'Open Source consente agli utenti il libero accesso e l'utilizzo del codice sorgente del software, che può essere adattato, modificato e ridistribuito. Tale metodologia è considerata rivoluzionaria rispetto a quella tradizionale del software commerciale coperto da copyright, in quanto coinvolge sviluppatori distribuiti in tutto il mondo che evolvono il software correggendo bug ed ottimizzando il codice. L'espansione costante di tale modello sta portando all'utilizzo di valide alternative Open Source a molti software proprietari soprattutto infrastrutturali quali ad esempio il sistema operativo Linux, il web server Apache o il mail server Sendmail.

Caratteristiche peculiari del processo di sviluppo Open Source sono lo sviluppo incrementale del software e la continuità dei progetti sul lungo periodo. Non è individuabile un unico modello di sviluppo; in alcuni casi si parte con un primo release di codice avente funzioni minime poi distribuito al fine di ulteriori miglioramenti, in altri casi si utilizza un nucleo di software già esistente fornito da università o da aziende commerciali (poi "donato" alla community). L'esistenza di una forte ed ampia community è una condizione necessaria ma non sufficiente per assicurare il successo di un progetto. È necessaria una schiera di sviluppatori dedicati e motivati; un esempio è Linux il cui progetto sta vedendo il coinvolgimento di moltissimi sviluppatori con specifico interesse a migliorare il sistema operativo per proprio utilizzo e per accrescere il proprio rating nella community nello sviluppo delle successive personalizzazioni.

Possiamo sintetizzare alcuni punti di forza e debolezza del software Open Source.

I vantaggi principali sono:

- **la libertà di utilizzo**, che consente di avere la massima flessibilità nella gestione del codice e forte indipendenza dai grandi fornitori nel processo di scelta dei pacchetti disponibili;
- **l'efficacia del processo di aggiornamento,** grazie alla contribuzione di più sviluppatori in parallelo ed alla continuità del processo per periodi medio-lunghi;
- **i minori costi e risorse necessarie,** per alcune tipologie di software, grazie alla forte riusabilità di componenti già sviluppate ed alla diminuzione dei costi di gestione ed ottimizzazione del codice;

- **la migliore qualità del software,** dato il coinvolgimento ed i feedback di molteplici utilizzatori worldwide e l'efficacia nello scoprire possibili vulnerabilità di sicurezza;
- **la gestione della conoscenza,** grazie alla condivisione di banche dati informative ed esperienze sui prodotti sviluppati sia all'interno della Community che verso altre istituzioni come le università.

I punti di debolezza sono:

- **la gestione dei progetti**, in quanto pianificare e distribuire prodotti Open Source in un contesto di massima de-localizzazione degli sviluppatori con pochi strumenti di controllo è molto complesso. Esiste un'estrema fluidità nel processo di appartenenza alla community; quindi non è prevedibile a priori la consistenza del parco degli sviluppatori;
- **la qualità e la sicurezza,** sono vantaggi ma anche criticità. Non sempre gli sviluppatori si dedicano ad aspetti primari come l'intuitività delle interfacce o la documentazione; inoltre la rapidità di sviluppo conduce alla mancanza di test sostanziali prima dell'immissione del codice in community con la conseguente presenza di molti bugs. Per quanto riguarda poi la sicurezza in senso stretto, certe applicazioni "sensibili" potrebbero risentire di problemi di riservatezza nell'utilizzo di kernel presi dalla community.

4.1 Criteri di valutazione generale

Il mercato del software Open Source presenta caratteristiche diverse da quello tradizionale; una di queste riguarda il numero estremamente elevato di informazioni riguardo il software ed il processo di valutazione; il mercato tradizionale non è altrettanto trasparente. La valutazione del software proprietario è normalmente focalizzata sulle funzionalità e sul costo della licenza, nell'Open Source la valutazione include altre informazioni provenienti da fonti diverse che forniscono un quadro completo del prodotto, del suo sviluppo e delle sue prospettive.

Dall'esperienza raccolta nello sviluppo del software nelle community, emergono due principali modelli per valutare la maturità del software Open Source.

Il primo modello è il **Capgemini Expert Letter Open Source maturity model**[5]. Tale modello consente di determinare se e quale prodotto Open Source corrisponde ai requisiti utilizzando un approccio basato su sette fasi. Il modello prevede inizialmente la ricerca e selezione del prodotto, utilizza degli indicatori di misura per classificare il prodotto, sviluppa delle scorecard ed infine elabora una valutazione finale.
Il secondo modello, il **Navica Open Source maturity model**[6], è basato su una logica di analisi di prodotto in tre fasi: assessment degli elementi del prodotto, assegnazione di un peso e calcolo del modello di maturità.

Sui criteri e modelli di valutazione del software Open Source esiste ampia letteratura[7] tendenzialmente focalizzata sui principali fattori che contribuiscono al successo dei pacchetti software OS. Nella tabella di seguito riportata vengono riassunti, in forma comparativa, i principali criteri di valutazione in relazione alle principali teorie sviluppate.

Tabella 1 – Matrice di criteri e modelli per la maturità dei prodotti Open Source

Criterio	**Capgemini Expert Letter Open Source maturity model** Duijnhouwer and Widdows (2003)	**Navica Open Source maturity model** Golden (2005)	Crowston et al. (2004)	Wheeler (2005)	Donham (2004)	Nijdam (2003)
Community	Sì	Sì	Dimensione del team e livello di attività	In fase di supporto	-	Attività dei gruppi
Rilasci	-	Livello di attività	Livello di attività	Correttiva	-	Attività dei gruppi
Longevità	Vetustà	Si	-	Si	Maturità	Versione
Licenze	Si	Aspetti di rischio	-	Si	Si	Si
Supporto	Si	Si	-	Si	Si	-
Documentazione	Facilità di distribuzione	Si	-	In fase di supporto	Si	-
Sicurezza	Si	Come rischio	-	Si	Si	-
Funzionalità	Tempi di sviluppo	Si	-	Si	Si	Si
Integrazione	Si	Si	-	Nelle funzionalità	Nella infrastruttura	-

[5] *Frans-Willem Duijnhouwer and Chris Widdows (Capgemini), Open Source Maturity Model 1.5.3, August 2003, (Expert Letter)*

[6] *Succeeding with Open Source (Addison-Wesley, 2005), authored by Navica's CEO, Bernard Golden.*

[7] *Ad es. vedi Wheeler, W. How to evaluate Open Source/free software programs (http://www.dwheeler.com/oss_fs_eval.html).*

Community

È uno degli aspetti più importanti e costituisce una delle principali fonti di informazioni. La community fornisce agli sviluppatori i feedback utente e le nuove idee, assicurando la qualità e la coerenza con i requisiti. Una community ben strutturata evidenzia l'interesse del team di progetto, permette all'utente di partecipare esprimendo i desiderata e sviluppando una interazione diretta con coloro che sviluppano. Elemento importante riguarda le attività di test che vengono svolte all'interno della community. A differenza del software tradizionale, dove gran parte del costo è sostenuto per eseguire modalità intensive di test e di verifica di qualità, nel caso Open Source le attività di test vengono svolte all'interno della community, che è la risorsa preposta al task, attraverso il libero coinvolgimento di tutti i suoi partecipanti che mostrano interesse verso il progetto.

Di seguito si riassumono alcun aspetti sostanziali nel processo di valutazione della community:

- **posts:** misura il numero di messaggi scambiati per unità di tempo ed il numero di argomenti proposti;
- **utenti:** misura il numero di utenti e il numero di sviluppatori partecipanti;
- **response time:** misura il numero di risposte alle domande-utenti ed il tempo che intercorre tra domanda e risposta;
- **qualità:** misura la qualità nei messaggi scambiati e nelle repliche; la qualità è espressa dai contenuti, dalla brevità e dalla discussione generata dai messaggi;
- **rapporti:** misura il grado di relazione nei rapporti tra i membri, soprattutto nei confronti dei nuovi iscritti.

Rilasci

Ogni progetto rilascia versioni successive di software. Il numero di rilasci in un certo periodo ed il loro contenuto in termini di cambiamenti apportati, esprimono lo stato d'avanzamento del software sviluppato e si rilevano spesso come un buon indicatore della serietà di chi lavora al progetto. Esistono repository (tra i più accreditati SourceForge[8] e FreshMeat[9]) ove si condividono informazioni tra i membri di progetti utili a valutare i rilasci.

[8] *http://sourceforge.net*

[9] *http://freshmeat.net/*

Esistono due tipologie di rilascio per un progetto Open Source:

- **stable releases:** si tratta delle versioni di software ritenute fondamentali per un utilizzo del software in produzione aventi il minimo rischio di errore;
- **development versions:** possono avere diverse forme, versioni beta, CVS[10] o build giornaliere. Si tratta di versioni che sono usabili con rischio d'errore in quanto non pensate per la produzione. Un progetto rilascia nuove versioni e pubblica una nota relativa ai rilasci contenente la lista dei cambiamenti apportati rispetto alla precedente. Il progetto può inoltre avere una road map che mostra gli obiettivi dello sviluppatore, quanti ne sono stati raggiunti e le dead-line per quelli da raggiungere. Dalla verifica del rispetto della road map è possibile fare una valutazione sulla modalità di pianificazione delle attività del team di progetto.

La tracciabilità dei cambiamenti di un progetto può dare ulteriori informazioni quali:

- **il numero di rilasci:** l'anno rappresenta una corretta unità di riferimento temporale per il conteggio del numero di rilasci;
- **il contenuto di un rilascio:** la tracciatura nei log dei cambiamenti o le note del rilascio spiegano il contenuto delle componenti modificate. Una distinzione viene effettuata se il rilascio contiene fix a problemi oppure evoluzioni di funzioni o nuove funzioni. Si predilige un minor numero di rilasci contenenti variazioni significative rispetto ad un elevato numero con minor contenuto. I due aspetti, numero e contenuti dei rilasci, dovrebbero essere coerentemente bilanciati.

Longevità

La longevità di un prodotto misura la stabilità del progetto e le possibilità di sopravvivenza. Chiaramente la vetustà di un software può anche implicitamente significare l'utilizzo di vecchie tecnologie di sviluppo e mantenerlo in essere può significare il suo completo rifacimento. Il superamento del ciclo di vita rappresenta comunque un segno di maturità. Età del software e attività di progetto sono correlate.

[10] *Il Concurrent Versions System (CVS), conosciuto anche come Concurrent Versioning System, implementa un sistema di controllo versione: mantiene aggiornato il lavoro ed i cambiamenti in un insieme di file*

Un progetto nuovo sviluppa un elevato numero di attività in quanto c'è un alto interesse di tutti i suoi utenti rispetto ad un progetto datato che ha raggiunto i suoi obiettivi e vive nella correzioni dei soli errori.

I criteri da considerare sulla longevità possono essere:

- **età del software:** calcolata dalla data del primo rilascio;
- **numero di versione:** una versione **0,x** significa che il software non è completo o pronto per andare in esercizio. Numeri di versioni molto rapide possono essere sintomo di una falsa evoluzione.

Licenze

Esistono diverse tipologie di licenze Open Source; la più diffusa è GNU GPL (General Public License) elaborata da Stallman, che introduce il concetto di *copyleft* il cui significato è quello di utilizzare il copyright per assicurare che il software e le sue derivazioni restino libere, obbligando chi ridistribuisce il software a mantenere lo stesso tipo di licenza. La GPL ha un approccio molto restrittivo e rigido nel copyleft per cui nel tempo è stata elaborata una versione più permissiva che è stata ribattezzata L-GPL (L sta per less o light) che permette l'utilizzo di librerie Open Source senza la necessità di dover distribuire il codice sorgente.
Altra tipologia di licenza non basata sul concetto di copyleft è quella BSD (Berkeley Source Distribution); è una licenza che è stata oggetto di molte controversie ed è utilizzata in molte componenti di software commerciale. Infine ci sono altri tipi di licenza che sono rimaste legate al prodotto stesso come nel caso di Mozilla, PHP e Apache. La scelta di utilizzare un tipo di licenza è legata all'obiettivo di utilizzo del software che si persegue.

Supporto

Ci sono due tipi di supporto per i prodotti software:

- **usage support:** è il supporto per problematiche di installazione e uso del software;
- **maintenance:** è il supporto alla risoluzione dei problemi del codice.

Le due tipologie spesso sono utilizzate in forma congiunta in quanto non sempre chi attiva la richiesta conosce il corretto uso del software. La modalità di gestione del supporto è un indice della serietà del lavoro svolto dagli sviluppatori, così come l'utilizzo di un prodotto di tracker è anch'esso indice di maturità dello sviluppo.
Oltre al supporto erogato direttamente dalle community, molti progetti di grandi dimensioni prevedono possibilità di supporto a pagamento. Anche se il software è gratuito, all'utente si offre la possibilità di attivare un supporto professionale in base ad un canone sottoscritto per un certo periodo. Le compagnie che offrono tali servizi su un prodotto Open Source sono identificate come terze parti. Tale servizio è un indice di serietà del prodotto.

Gli elementi da ricercare nella valutazione sono:

- esistenza di un forum o un gruppo dedicato per rispondere a domande sull'installazione ed uso del software;
- partecipazione degli sviluppatori al forum;
- adeguatezza delle risposte e della documentazione fornita.

L'organizzazione della community influenza l'efficacia del supporto. Un progetto di grandi dimensioni dovrebbe avere più aree per ciascuna parte di progetto. Ciascuna area non dovrebbe assumere una connotazione troppo specialistica e di dettaglio altrimenti non si genererebbe sufficiente interazione con una conseguente perdita d'interesse. Il supporto ai problemi di norma viene gestito attraverso strumenti di tracking; studi statistici hanno evidenziato che nei progetti di successo il numero degli sviluppatori che lavorano sulle fix è molto più alto di quelli che sviluppano nuovo codice.

Documentazione

Esistono due principali tipi di documentazione:

- **documentazione utente:** contiene tutti i documenti che descrivono come usare il software; l'articolazione della documentazione può essere legata alla complessità del software e alla tipologia di utenti previsti;

- **documentazione di sviluppo:** contiene tutti i documenti che descrivono come aggiungere o cambiare il codice. Senza tale documentazione la community, volontaria e decentralizzata, non potrebbe lavorare correttamente. Tale documentazione, sebbene esista anche per il software commerciale, però non è di norma pubblicata. Attraverso i commenti si fornisce spiegazione della logica applicativa adottata, di come usare e modificare il codice.

Un terzo tipo di documentazione, spesso disponibile per applicazioni di grandi dimensioni, è quella relativa alla manutenzione, che include le istruzioni per l'installazione e l'upgrade. Tale documentazione deve essere estremamente chiara: deve esprimere i requisiti di infrastruttura e la modalità d'installazione.
La redazione della documentazione è spesso ritardata rispetto allo sviluppo del software e viene prodotta successivamente allo sviluppo delle funzionalità, specialmente quella utente. Una valida metrica di misura è quanto spesso viene aggiornata la documentazione e quanto è coerente con il livello di sviluppo corrente del software.

Sicurezza

La sicurezza nel software, in special modo per quello Open Source, segue due approcci:

- quello per cui è consigliabile che la sicurezza sia oscurata all'interno del software. Tale approccio non è compatibile con il concetto di codice aperto;
- quello che vede nell'apertura del codice una salvaguardia perché le vulnerabilità vengono subito individuate.

La qualità del codice è strettamente connessa alla sicurezza e ciò è valido sia nel caso Open Source che per il software proprietario. Nel primo caso le vulnerabilità vengono individuate da hacker che provano a ledere l'integrità del software. Le vulnerabilità del software Open Source possono essere identificate anche dallo sviluppatore o dall'utente durante le attività di rivisitazione del codice.

In generale, la sicurezza deve essere gestita con serietà fornendo risposte veloci a qualunque vulnerabilità riportata. Esistono diversi tipi di schemi di controllo dei bug nelle componenti software potenzialmente vulnerabili. Tra i più diffusi si evidenziano quelli forniti da alcune società (http://www.securityfocus.com[11] e http://www.secunia.com[12]). Ad una maggiore diffusione e popolarità del software corrisponde una maggiore disponibilità dei report di vulnerabilità. Quindi la mancanza di tali report non è sinonimo di sicurezza del software.

Funzionalità

La comparazione delle funzionalità non è un elemento di valutazione specifico del solo software Open Source ma è largamente diffuso e utilizzato dai modelli di sviluppo applicativo. Il software Open Source utilizza spesso lo slogan di sviluppo "rilasci veloci e frequenti"; tale metodo è un fattore abilitante alla correzione degli errori del codice in quanto in continua revisione. Inoltre incoraggia la contribuzione degli sviluppatori perché il risultato del lavoro viene valorizzato nei frequenti rilasci anche se a spese di una certa incompletezza nella fase di sviluppo iniziale. Rispetto al software proprietario ove il fornitore può fornire ampia descrizione delle funzionalità, nel caso Open Source possono non essere da subito disponibili le informazioni e la documentazione.

Una lista dei requisiti funzionali può essere un elemento utile a verificare se le funzionalità richieste sono disponibili. Può essere opportuno fare una distinzione tra le funzionalità strettamente necessarie che possano far orientare la decisione verso altri software e quelle che rappresentano un plus e che innalzano la valutazione. Nella comparazione delle funzionalità, le caratteristiche che sono parte dei requisiti funzionali hanno maggiore priorità, altre di tipo addizionale possono trovare un utilizzo futuro.

[11] *SecurityFocus è una organizzazione che tratta information security su Internet; è indipendente dai fornitori e fornisce informazioni su obiettivi, tempi sulla sicurezza a membri della comunità (utenti finali, amministratori di rete e sicurezza, CIO,..) senza pagamento di canoni. Dispone di servizi di newsletter, mailing list, bug report, database delle vulnerabilità.*

[12] *Secunia è un sito web dedicato alla sicurezza del software. Il portale si occupa di analizzare più di 4000 programmi per scovarne i punti deboli e proporre reports dei risultati.*

Integrazione

La metodologia Cap Gemini fa riferimento a diversi criteri di integrazione; questi sono maggiormente significativi nel caso di software che ha forte interazioni con altri prodotti interni all'azienda e per coloro che prevedono una forte attività di personalizzazione, aggiungendo funzionalità al fine di renderlo il più aderente alle aspettative della propria organizzazione.

- **modularità:** significa che il software è composto di più moduli separati ognuno auto consistente. Tale struttura ha i seguenti vantaggi:
  – facilità di gestione dei singoli moduli;
  – facilità di aggiungere parti personalizzate senza alterare il cuore del software;
  – l'approccio modulare agevola la selezione delle funzionalità necessarie tralasciando quelle non previste dall'uso del software;
  – utilizzo dei singoli moduli nello sviluppo di software di software commerciale. È possibile operare una scelta diversificata sui moduli che possono essere rilasciati come Open Source e quelli venduti come software proprietario (tale modello è chiamato **Razor**);
- **standard:** nel mercato emergono molti standard aperti per sviluppare in modo semplice la cooperazione tra software diversi. L'uso di tali standard quindi agevola la comunicazione tra prodotti software. Gli standard nell'Open Source sono un segno di maturità del software. Nella lista delle caratteristiche normalmente vengono indicati gli standard a cui il software aderisce;
- **interazione con altri prodotti:** questa caratteristica è vicina alla tematiche degli standard; ad esempio l'uso dei formati di compatibilità di documenti tra prodotti quali Microsoft Word e Adobe PDF implementati nel software OpenOffice;
- **requisiti software:** i requisiti software indicano la compatibilità del prodotto verso uno specifico software e/o versione.

Uno studio[13] dell'Università di Padova – Dipartimento delle Scienze Economiche "Marco Fanno" – propone un modello di analisi finalizzato a convalidare una serie di assunti sulla validità dei progetti di sviluppo Open Source sulla base di un modello matematico, prendendo in considerazione quale campione i circa 90.000 progetti censiti nel 2004 sul repository SourceForge[14]. Per l'analisi dei progetti vengono considerate diverse metriche quali lo stato di sviluppo, data di registrazione, numero sviluppatori, numero di bugs, patch e feature richieste, licenze, linguaggio di programmazione e argomenti.

Vengono formulate quattro ipotesi di previsione che il modello dimostra poi come veritiere; in particolare:

1) i progetti distribuiti sotto la licenza GPL (più restrittiva) sono quelli che hanno un peggiore stato di sviluppo;
2) le applicazioni destinate ad utenti sofisticati[15] sono quelle che hanno un maggiore successo di sviluppo;
3) lo sviluppo di un progetto ottiene maggiori benefici se la community di sviluppatori è molto estesa. A tal proposito si fa presente che oltre una certa dimensione subentra un problema di complessità organizzativa;
4) i fattori critici che incidono sullo stato di sviluppo di un progetto si differenziano in base all'età del progetto.

[13] "From Planning to Mature: on the Determinants of Open Source Take Off" a cura di Stefano Comino, Fabio M. Manenti e MLaura Parisi, working paper n. 35 pubblicato nel gennaio 2007.

*[14] **SourceForge.net** è una locazione centralizzata per sviluppatori software che debbano controllare e gestire lo sviluppo di software Open Source; essa inoltre costituisce un source code repository. SourceForge.net è ospitata da VA Software e si avvale di una versione del software SourceForge. Questo sito ospita una grande mole di progetti Open Source: nel Febbraio 2007 se ne contavano 140.417 con 1.498.326 utenti, ma molti di questi sono progetti in letargo o che coinvolgono un solo interessato.*

[15] È da intendersi con "sofisticato" un utente che utilizza strumenti di sviluppo e di gestione software. Viceversa con "non sofisticato" si fa riferimento all'utilizzo di software office e gaming.

4.2 La prototipazione come modello di sviluppo

Il modello di sviluppo Open Source possiede un forte potenziale innovativo durante tutte le sue fasi implementative quali il disegno, lo sviluppo del codice e l'utilizzo. In particolare, tra le pratiche emergenti di sviluppo ed utilizzo si sta affermando una specifica modalità definita **patchwork prototype.**

Tale metodo enfatizza l'importanza centrale del diretto coinvolgimento dell'utente, riducendo i lunghi periodi di tempo che intercorrono tra il disegno dell'idea e la sua realizzazione. L'obiettivo è strutturare insieme le applicazioni Open Source al fine di creare prototipi significativamente aderenti a componenti critiche identificate dall'utente (prototipi ad alta fedeltà). Il patchwork prototype combina la rapidità e l'economicità dei prototipi su carta con l'ampiezza dei prototipi "orizzontali" e la profondità di quelli "verticali".

Il patchwork prototype nasce da esperienze su progetti cosiddetti "cyber-collaboratory" ossia progetti di sviluppo di sistemi per l'interazione di tipo collaborativo-virtuale. Con tale metodologia si cerca di rispondere ad alcune criticità che in genere emergono in fase disegno di sistemi collaborativi. La community di utenti per le quale si avvia il processo di sviluppo potrebbe ancora non essere perfettamente determinata e quindi non essere note le modalità di interazione tra gli utenti; in alternativa ci potrebbero essere casi in cui la community esiste ed è osservabile, ma non sono disponibili gli strumenti che consentono di praticare e gestire le attività di collaborazione. In tali contesti la raccolta dei requisiti utente può condurre a false interpretazioni.
Il patchwork prototype appare quindi come una metodologia guidata dall'utente (sotto forma di gruppi o community) che mira a sfruttare tecniche computazionali per migliorare le attività quotidiane e agevolare quelle future.

Il patchwork prototype possiede tre componenti chiave:

- rapida interazione di prototipi ad alta fedeltà;
- inserimento dei prototipi nelle attività di lavoro quotidiane dell'utente;
- ampia collezione di feedback, agevolati dalla presenza di un membro nella comunità degli utenti.

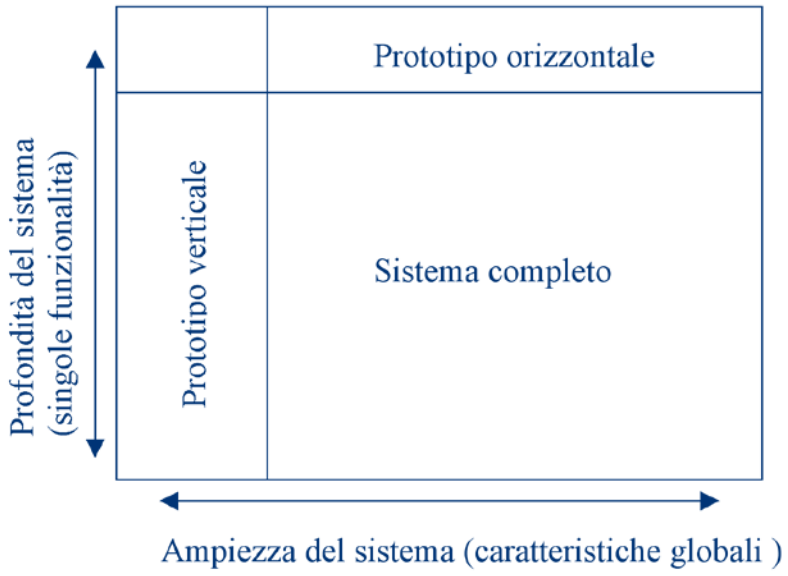
Il suo approccio mira a superare le difficoltà della classica metodologia di prototipazione rapida, su cui comunque occorre fare un breve excursus.

La prototipazione rapida è un metodo per la raccolta dei requisiti disegnato per incrementare ed agevolare la comunicazione tra utenti e sviluppatori, aiutando questi ultimi nel prevedere le conseguenze di un particolare disegno applicativo prima della costruzione dell'intero sistema. L'obiettivo della prototipazione rapida è quindi quello di sviluppare una serie di modelli iterativi che rappresentino le aree funzionali, che facilitino la creatività e forniscano feedback circa il valore dell'impostazione prima di investire tempo e risorse nell'implementare il totale delle funzionalità di sistema. Esistono due aspetti da considerare nello sviluppo di questi prototipi: il primo è il passaggio da un prototipo a bassa fedeltà ad uno ad alta fedeltà. I prototipi a bassa fedeltà hanno il vantaggio di essere veloci e poco costosi in termini di sviluppo e di iterazione. Un esempio è la prototipazione su carta. Tale modalità non consente però agli sviluppatori di osservare qualsiasi reale modalità di utilizzo del sistema o di capire le interazioni tra i componenti del sistema e l'utente; di conseguenza diventa difficile modellare in dettaglio le componenti necessarie alla realizzazione del sistema.
Al contrario, i prototipi ad alta fedeltà possono simulare la reale funzionalità. Sono dei programmi sviluppati in un ambiente di sviluppo semplice (es. Visual Basic) o con strumenti di prototipazione (es. CASE). Tali programmi consentono di osservare le interazioni con gli utenti e di capire le implementazioni più onerose e di maggior costo da sviluppare. Critiche a tale metodo sono relative ai costi di sviluppo e, nel caso di utilizzo successivo di tecniche RAD (rapid application development), nel limite di scalabilità del software finale.

Un secondo aspetto da considerare nella prototipazione rapida è il range di azione. Il software può esser visto come una pila di strati che interagiscono nella parte più alta con l'utente e in quella più bassa con il sistema operativo. I prototipi orizzontali sono ad ampia portata ovvero misurano l'ampiezza del sistema ma solo con un strato software (l'interfaccia utente). L'utente può interagire con una maggiore numero di funzioni senza poter però scendere in dettaglio.

I prototipi verticali invece esplorano il sistema in profondità attraverso tutti gli strati del software ma per poche funzioni.

Figura 1. protipi orizzontali e verticali



Da un punto di vista economico l'approccio di sviluppo per prototipi ad alta fedeltà, verticali e orizzontali, è spesso un problema. Anche se gli sviluppatori realizzassero, ad esempio, una serie di prototipi verticali, sarebbe poi necessario completarli in un'ottica orizzontale al fine di avere funzionalità generali sufficienti. Ciò comporta impegno economico, tempo di sviluppo in presenza di funzionalità e robustezza minimali.

La diffusione della produzione in scala di sistemi Open Source ha fornito ampie possibilità per lo sviluppo di codice affidabile, usabile e ricco di funzioni. Tali programmi sono facilmente integrabili insieme perché il codice è aperto e modificabile. Questo rappresenta una grossa opportunità per gli sviluppatori che rapidamente possono costruire e valutare prototipi ad alta fedeltà per gli ambienti collaborativi utilizzando una forma di "patchworking" di molteplici componenti Open Source. Dall'esperienza derivata dalla diffusione di tale approccio prototipale si possono riassumere alcuni aspetti chiave.

Sviluppo di una rappresentazione iniziale del sistema target; è il passo più difficile perché richiede che il team di disegno sintetizzi la conoscenza e la comprensione del problema in un disegno coerente. Nella prima iterazione del processo è spesso utile l'uso di prototipi su carta e le funzioni richieste servono principalmente come forma di brainstorming. La cosa importante è avere un punto concreto di partenza.

Selezione e integrazione degli strumenti; il metodo patchworking prototype non richiede teoricamente software Open Source; può essere utilizzato anche software commerciale purché fornisca delle API[16] di interfaccia. L'utilizzo però di software Open Source garantisce importanti vantaggi. Infatti senza l'accesso al codice sorgente gli sviluppatori sono limitati nell'assemblaggio dei diversi moduli e nel capire quali feature possono essere abilitate nel processo di integrazione visuale dei moduli con il resto del sistema. La natura web-based della metodologia permette diverse modalità di integrazione del software nel prototipo a seconda dei livelli di profondità scelti. Perché risulti conveniente fare il prototipo, l'effort deve essere minimo. I programmatori devono essere liberi di aggiungere e togliere componenti software con minimo sforzo. Il fatto che il software Open Source sia disponibile, inoltre, è importante per motivi di budget perché consente agli sviluppatori di evitare complesse negoziazioni sulle licenze. Infine, molti software Open Source hanno community di riferimento con membri che sono spesso attivi nel rispondere alle domande degli sviluppatori. Tutto questo facilita la raccolta dei requisiti in quanto può essere rapidamente attuata l'iterazione del prototipo con ricche funzionalità a costo minimo da parte degli sviluppatori.

Diffusione, ritorno e iterazione; durante la fase di utilizzo del prototipo nuovi utenti si integrano nella "cyber-infrastruttura" con proprie best practice ed esplorano ambiti di collaborazione. L'insieme dei feedback dati dall'esperienza degli utenti contribuisce alla raccolta dei requisiti. L'esistenza di un prototipo crea una infrastruttura tecnologica che influenza la negoziazione delle pratiche quotidiane essendo sviluppate dagli utenti attraverso le loro attività. Dall'interazione con il prototipo per un certo tempo, raccogliendo feedback basati sull'esperienza, emerge una più ricca comprensione

[16] *API sta per **Application Programming Interface** (Interfaccia di Programmazione di un'Applicazione), sono ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per un determinato compito (fonte Wiki).*

del sistema nei suoi aspetti tecnico-sociali. L'iterazione è essenziale per un approccio rapido. Innanzitutto permette l'esplorazione di più feature alternative, può scoprire aspetti trascurati, può rafforzare la necessità di particolari funzionalità e requisiti; infine permette agli utenti di avere un flusso costante di nuove possibilità di disegno. A seguito del processo di iterazione può emergere un disegno più reale ed aderente, grazie all'identificazione di una serie di requisiti chiave. Il disegno non è ancora completo ma il lavoro su un sistema prodotto in scala può iniziare.

La metodologia patchworking prototype indirizza due dei maggiori problemi che gli sviluppatori incontrano nella realizzazione di un sistema tecnico-sociale:

- consente al team di disegno di ricevere feedback sul prototipo in situazione di utilizzo reale. L'interazione quotidiana degli utenti focalizza i loro feedback intorno a specifici problemi;
- riduce lo spazio interpretativo spostando quelle che sembrano supposizioni verso funzionalità e requisiti veri.

L'utilizzo in un contesto reale è un elemento che differenzia tale approccio dagli altri metodi. I prototipi su carta sono di norma dati ad utenti in laboratorio così che ogni task è artificiale. Altri metodi più onerosi quali il disegno partecipatorio e etnografico provano ad incorporare l'uso reale nel processo di disegno facendo inizialmente partecipare l'utente nel team e successivamente, osservando l'utente nel suo naturale ambiente di lavoro. Al momento in cui si introduce la tecnologia cambiano inevitabilmente le pratiche e le strutture sociali presenti nell'ambiente di lavoro in modo non prevedibile. Il patchworking prototype supera questi limiti essendo poco oneroso e consentendo in tempo reale feedback sui problemi dell'utente con il software e gli effetti che il software ha nel contesto di lavoro.

L'uso del patchworking prototype è ancora in una fase di immaturità. La relativa facilità con cui il prototipo può essere costruito sta a significare che il metodo permette la diffusione in nuovi contesti di utilizzo. Offre un economico ed efficace metodo per esplorare uno spazio di rappresentazione e valutazione delle feature di prodotto.

Ciò consente di sapere quando acquistare software dai fornitori e di valutare quanto è efficace una soluzione di mercato per l'organizzazione. Con l'aumentare degli sviluppatori e delle organizzazioni di supporto all'Open Source, cresce la disponibilità di prodotti Open Source e quindi di componenti disponibili abbassando così il costo per lo sviluppo di prototipi. In questo senso il patchworking prototype è un eccellente esempio di come il software Open Source possa indurre innovazione (di processo oltre che, eventualmente, di prodotto).

4.3 Modelli di migrazione di software Open Source in ambiente Desktop

La crescita di mercato del software Open Source può essere attribuita a vari fattori quali il pricing elevato dei prodotti commerciali (ad es. Microsoft), la disponibilità e la qualità crescente del software e l'adozione di standard aperti nelle organizzazioni pubbliche e private. Nonostante sia sempre più riconosciuto il software Open Source come alternativa al software di mercato nell'area dei server, l'uso di software aperto per i desktop resta ancora limitato. L'impatto sull'organizzazione, rispetto ad una migrazione di server, è maggiore e richiede il coinvolgimento di un alto numero di utenti. In tal senso è piuttosto elevato l'interesse nello sviluppo di linee guida che assistano durante la fase di migrazione.

Il software Open Source desktop include il software di produttività individuale (es. OpenOffice), client e-mail (es. Mozilla thunderbird), browser Internet (es. Mozilla firefox) e altre utility.

La migrazione alla piattaforma desktop Open Source è una decisione strategica. Potenziali vantaggi sono associati al basso costo di licenze, l'accessibilità del codice, la disponibilità e stabilità del software, il supporto attraverso le community, la scalabilità e la sicurezza. Al contrario, si evidenziano possibili svantaggi quali la perdita del supporto dei fornitori, le difficoltà tecniche di installazione, la mancanza di integrazione, gli skill tecnici richiesti, la compatibilità con l'hardware e le resistenze degli utenti.

Una migrazione richiede una analisi del ROI (Return On Investment), in termini di attuale ed atteso TCO (Total Cost of Ownership) a cui associare i costi di migrazione.

La migrazione non deve necessariamente essere un cambiamento *in toto* del software della postazione. Infatti, in base alla tipologia di utenti il desktop Linux può dimostrarsi più appropriato solo per alcuni a causa dell'uso di molte applicazioni proprietarie non sempre compatibili con il sistema Linux.

Da una analisi Gartner[17] risulta che i costi di migrazione sono significativi se comparati con i risparmi. Un break-even point è stimabile in 1.3 anni nel miglior caso di migrazione da Windows 95 a Linux desktop. La valutazione cambia a secondo se si tratta di user data entry o business user per i quali si prevede che l'utilizzo di desktop Linux maturerà nei prossimi 3 o 5 anni.

Esistono diverse metodologie per la migrazione del software Open Source in ambiente Desktop di cui si analizzano gli aspetti più salienti.

- Lachniet Framework [18]
- Wild Open Source Migration [19]
- NetProject IDA[20]

Lachniet Framework è un set di linee guida che pone il focus sulle attività preliminari che vanno condotte prima della migrazione in un ambiente aziendale. Il framework si compone di tre tipologie di attività: amministrative, di sviluppo applicativo e tecnologiche.
Le attività amministrative sono volte ad ottenere dal Top Management il necessario supporto alla migrazione mediante:

- lo sviluppo di politiche di alto livello anche attraverso il ricorso ad incentivi;
- l'adozione di nuove politiche di acquisto del software;
- l'adozione di politiche di assunzione basate su skill tecnici (es. Linux);

[17] *Prentice & Gammage 2005*

[18] *Lachniet, M. (2004). Desktop Linux Feasibility Study Overview, August 20, http://lachniet.com.desktoplinux*

[19] *Wild Open Source Inc. (2004). Steps to Take when Considering a Linux Migration, [Online], http://www.wildopensource.com/technology_center/steps_linux_migration.php (25 August 2005*

[20] *Netproject. (2003). IDA OSS Migration Guidelines, [Online], Version 1.0, Available: http://www.netproject.com/docs/migoss/v1.0/methodology.html (29 August 2005)*

- la creazione di un Team Linux che continui l'analisi e l'implementazione di software Open Source;
- l'assunzione di un Project Manager esperto in Open Source;
- l'identificazione di un budget di spesa.

Le attività di sviluppo applicativo aiutano ad assicurare che le future migrazioni del software Open Source siano fattibili sulla base dell'esperienza derivante dalla migrazione del software Desktop attraverso:

- l'identificazione di una piattaforma di sviluppo portabile, per gli sviluppi futuri, verso molteplici architetture, come java;
- l'interruzione di tutti gli sviluppi non portabili;
- l'ottenimento del training per gli sviluppatori;
- l'identificazione di una strategia di migrazione per le applicazioni già sviluppate.

Le attività tecnologiche assicurano che l'infrastruttura di back-end (serventi, rete,…) sia correttamente dimensionata prima della migrazione e assicura compatibilità con le applicazioni mediante le attività di test attraverso:
l'identificazione e la migrazione delle applicazioni di back-end;
l'ottenimento di training per le risorse IT;
la spinta del software Open Source nelle community.

Tale framework è dettagliato nel rappresentare i task che precedono la migrazione; non sono invece ben specificate le attività da sviluppare durante la fase operativa di migrazione.

Wild Open Source Migration è una metodologia composta di tre fasi: pianificazione, disegno e implementazione.

La prima fase – la pianificazione - identifica gli obiettivi della migrazione in base agli obiettivi del committente. Viene prodotto un assessment di dettaglio delle funzionalità del client e dei requisiti architetturali per poi elaborare il disegno di una soluzione di alto livello.

La seconda fase - il disegno - comporta la creazione di un disegno dettagliato con le specifiche tecnologiche. Tutti i requisiti hardware e software necessari alla migrazione vengono specificati.
La terza fase - l'implementazione - prevede la formulazione di una strategia di migrazione ed il piano di attività. I risultati vengono documentati e successivamente viene organizzato un audit ed una fase di formazione per gli utenti.

Tale metodologia prevede unicamente il coinvolgimento degli utenti nella fase finale del processo di migrazione e non specifica i passi realizzativi ma solo pianificazione e documentazione.

NetProject IDA è una metodologia basata sulle attività preliminari alla migrazione e si compone di cinque passi:

- descrizione del software e hardware in uso, dei requisiti funzionali, del piano relativo alla raccolta requisiti e delle altre attività del progetto;
- definizione delle motivazione della migrazione e della stima dei costi;
- verifica del piano di implementazione e test di una fase pilota;
- migrazione del software Open Source sui server e client;
- controllo dei risultati rispetto al piano.

Tale metodologia, di alto livello e focalizzata sulle attività tecniche, non specifica i punti relativi all'accettazione del management o degli utenti e la fase di training.

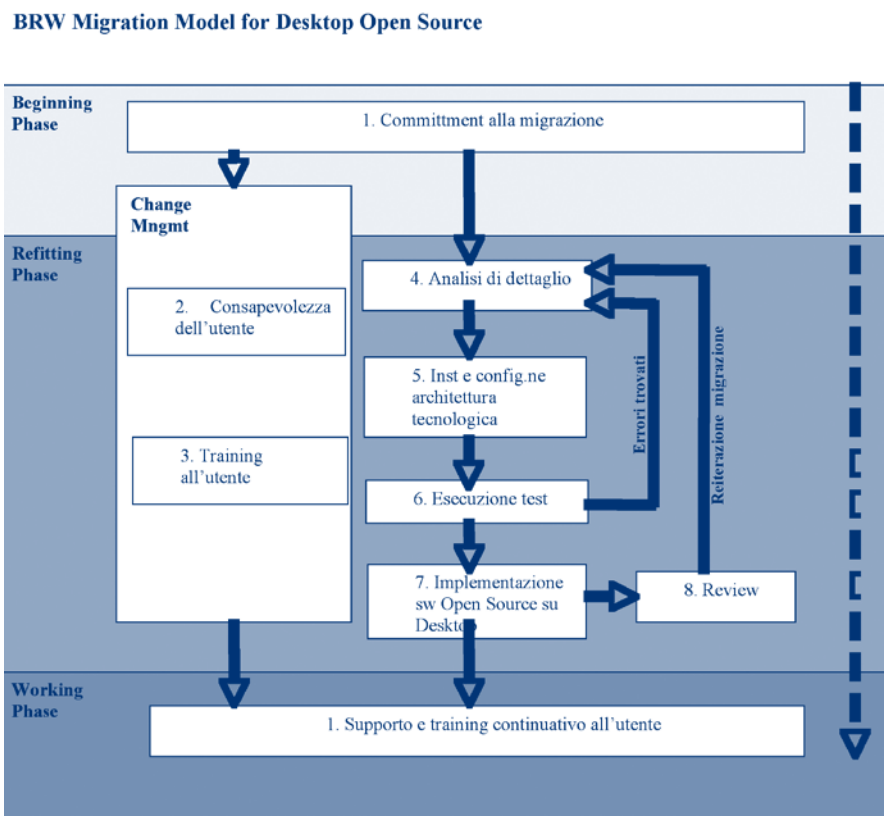
In Sud Africa, a seguito dell'analisi di alcuni casi di studio relativi a migrazioni del software Open Source desktop in diversi settori di attività (settore pubblico, education e privato), alcuni autori[21] hanno individuato gli elementi fondamentali di caratterizzazione delle attività di migrazione.
Innanzitutto, coerentemente con la letteratura, il fattore principale che sottende alla migrazione sembrerebbe essere la promessa di benefici economici, quali la riduzione dei costi di licenza e la ridistribuzione del revenue su altre aree.

[21] *Van Belle, Brink, Roos & Weller - 2006*

Benefici secondari addotti sono la riduzione del lock-in dai produttori di software, la sicurezza ed il risparmio per l'aggiornamento hardware dovuto alla riduzione dei vincoli nel caso di ambienti non necessariamente di recente tecnologia. Altro aspetto comune rilevato nei casi di studio è la resistenza degli utenti al cambiamento, unita alla percezione durante le attività di training, di scarsa usabilità; tali tematiche non sempre sono oggetto di approfondimento nelle metodologie.

Sulla base di tali esperienze, è stata sviluppata una nuova metodologia denominata **BRW** focalizzata sui fattori di successo della migrazione del software in ambiente desktop Open Source ed adattabile a qualunque tipo di migrazione Open Source a seconda dell'ambiente tecnologico o del contesto.
Nella seguente tabella si riporta lo schema delle fasi del modello BRW di cui si fornisce una sintetica descrizione.

Figura 2: il modello BRW per la migrazione del Desktop OpenSource



Committment alla migrazione: qualunque progetto di change management necessita in modo vitale del supporto e del commitment del Top Management dell'organizzazione. Il commitment deve pervadere l'organizzazione fino ai livelli operativi e viene facilitato dall'uso di un progetto campione. Durante la fase iniziale della migrazione è necessaria l'acquisizione di tutte le risorse previste; questo richiede la predisposizione di un budget per acquisire tutti gli skill necessari. Inoltre è opportuno documentare la nascita del progetto, le motivazioni per la migrazione, gli obiettivi del progetto, i costi ed i tempi.

Consapevolezza dell'utente: per ridurre le resistenze degli utenti al cambiamento è opportuno il loro coinvolgimento nell'intero processo di migrazione. Ciò avviene attraverso la comunicazione biunivoca con l'utenza spiegando i razionali della migrazione, le aspettative e gli oggetti del cambiamento.

Training all'utente: un adeguata formazione on the job è necessaria per rendere efficiente l'utilizzo dei prodotti del nuovo desktop Open Source. Il miglior modo per imparare ad usare il nuovo corredo software è attraverso la sperimentazione e l'aiuto reciproco in alternativa alla formazione tradizionale.

Analisi di dettaglio: rappresenta il task più complesso della intera migrazione ed è la chiave del successo del progetto. Obiettivo è assicurare che il software Open Source venga implementato solo laddove può produrre efficienza e dove può incontrare le necessità dell'utenza. Le principali sottoattività di tale fase sono:

- sviluppare un assessment del parco software e hardware esistente;
- identificare i requisisti funzionali e tecnologici;
- identificare la strategia di business;
- identificare gli utenti da migrare;
- identificare i migliori prodotti Open Source in coerenza con le necessità utente;
- creare un disegno del sistema;
- definire la migliore strategia di migrazione.

Installazione e configurazione architettura tecnica: l'architettura di back-end (serventi, rete,…) deve essere aggiornata prima di avviare la migrazione dei desktop.

Esecuzione test: la progettazione ed esecuzione di test è vitale per assicurare che il nuovo software Open Source soddisfi le necessità utente. Il test viene fatto attraverso un pilota ovvero la verifica di un certo numero di prodotti software da parte di un insieme di utenti in un contesto di operatività reale. Tale test deve poter evidenziare qualsiasi incompatibilità hardware e/o software o qualunque scopertura funzionale che andrà risolta prima della diffusione del nuovo desktop agli utenti finali.

Implementazione software Open Source su desktop: la migrazione effettiva del software Open Source sui desktop va condotta a chiusura del test e dopo aver risolto tutti i problemi emersi.

Review: è una fase successiva alla migrazione fisica; assiste l'organizzazione attraverso l'esperienza maturata, utile per future migrazioni. È una fase della quale frequentemente non si tiene conto. La review è realizzata attraverso la documentazione dell'intero processo di migrazione e dovrebbe indicare i cambiamenti portati a termine, le analisi svolte, le variazioni dell'architettura, i problemi incontrati e come sono stati risolti.

Supporto e training continuativo all'utente: è essenziale l'attività di supporto e training all'utente in quanto non tutti i problemi vengono rilevati in fase di test; inoltre durante l'uso dei nuovi software l'utente si aspetta di poter svolgere con la stessa efficienza le attività quotidiane. Durante tale fase inoltre viene mantenuto un forte commitment al cambiamento.

Il modello BRW non ha trovato ancora una diffusione su larga scala nelle grandi organizzazioni; richiede inoltre un lavoro di affinamento successivo per renderlo ancora più usabile nei casi di migrazioni complesse.

5. Aspetti legali e contrattualistici

5.1 Inquadramento giuridico

La licenza di software libero / Open Source è un negozio giuridico attraverso il quale il titolare dei diritti esclusivi di utilizzazione economica su un programma concede, a titolo gratuito, alcune facoltà che la legge sul diritto d'autore gli riserva[22]. Si ricorda che il software rientra nell'ampia categoria delle opere di ingegno la cui caratteristica peculiare è quella di essere un bene immateriale. Per tale ragione il legislatore ha disciplinato attraverso le norme che vanno dall'art. 2575 all'art. 2583 del codice civile, ma soprattutto dalla legge n. 633/41 (Legge sul diritto d'autore)[23].

Non esiste una definizione giuridica di software libero ma piuttosto un'etichetta costruita sui valori condivisi dai ricercatori e formalizzata con due movimenti che dominano il mondo del software libero: la Fondazione sul Software libero (FSF), che riconosce che una parte del software è libera e divisa in quattro libertà che vengono allegate alla relativa licenza, e l'Iniziativa Open Source (OSI), che certifica la natura libera di una licenza in conformità con dieci criteri (certificato OSI).

Anche al software si applicano i diritti di esclusiva, il cui contenuto si avvicina al concetto di proprietà, tanto da essere definiti diritti di **proprietà intellettuale** (ove si ritenga tutelabile il software sulla base del diritto di autore) o di **proprietà industriale** (nei paesi in cui il software può essere coperto anche da esclusive di tipo brevettuale).

In linea generale, questi diritti di privativa si possono riassumere nell'esclusiva per il detentore dei diritti:

- di usare,
- di riprodurre e duplicare,
- di tradurre.
- di modificare,
- di commercializzare (c.d. **diritti di sfruttamento economico**) il programma su cui si vanta l'esclusiva, a titolo originario per esserne l'autore, o a titolo derivato per averne acquisito i relativi diritti dal titolare originario.

[22] *La licenza proprietaria consente invece solo l'uso del programma nei limiti consentiti dalla licenza stessa, riservando tutte le attività in capo solo al titolare dei diritti.*

[23] *Come novellato dal D.lgs. 518 del 29 (tutela delle opere d'ingegno, che include anche i programmi per elaboratore - equiparate ad opere letterarie ai sensi della convenzione di Berna) l'autore di software, con la licenza Open Source esprime semplicemente la propria volontà circa l'esercizio dei suoi diritti di esclusiva sul software.*

L'elemento qualificante del modello di licenza Open Source può senz'altro ritenersi la concessione di un diritto, non esclusivo e non soggetto a limitazioni temporali o territoriali, di utilizzare, distribuire, modificare un programma per elaboratore. L'autorizzazione all'utilizzo delle facoltà esclusive sul programma, nel caso in cui l'utente decida di ridistribuire a terzi il programma per elaboratore, si sostanzia nell'obbligo di consegnare, unitamente al software, anche una copia dello schema contrattuale della licenza e di permettere che il codice sorgente del programma sia liberamente accessibile a chiunque.

Altro aspetto peculiare è quello relativo agli obblighi nel caso in cui il software sia elaborato e prevede l'impegno, da parte dell'utente licenziante, di trattare i diritti esclusivi sulle modifiche da lui apportate negli stessi termini contrattuali previsti sull'opera base.

Gli elementi qualificanti i modelli contrattuali Open Source sono la concessione di un diritto, non esclusivo e non soggetto a limitazioni temporali o territoriali, di utilizzare, distribuire, riprodurre, modificare, ridistribuire e rappresentare in pubblico il programma per elaboratore[24]. Tali diritti sono riconosciuti direttamente all'utente nel momento in cui lo stesso ha la disponibilità del software; nel caso di una sublicenza, tutte le facoltà sono automaticamente attribuite al sublicenziatario e ciò senza alcuna restrizione ed il successivo utilizzo viene regolato dalle condizioni e dai termini espressi dalla licenza originaria.

È possibile pertanto individuare talune caratteristiche di massima che ci riconducono concettualmente al tipo di licenza di un software open-source[25], deve soddisfare talune caratteristiche come ad esempio: 1) la libertà di redistribuzione, nel senso che la licenza non può limitare alcuno dal vendere o donare il software che ne è oggetto, come componente di una distribuzione aggregata, contenente programmi di varia origine; 2) diffusione del codice sorgente, poiché il programma deve includere il codice sorgente e ne deve essere permessa la distribuzione sia come codice sorgente che in forma compilata.

[24] *Nel caso in cui il programma per elaboratore in qualche paese sia protetto come invenzione industriale e non come oggetto del diritto d'autore, viene concesso il diritto di utilizzo, di distribuzione, di riproduzione, di vendita e di offrire in vendita il software.*

[25] *Le licenze di software libero / Open Source non possono essere accomunate alle licenze utilizzate nella commercializzazione del software c.d. pacchettizzato, cioè software standardizzato destinato al mercato di massa.*
In quest'ultimo caso, infatti, i produttori adottano dei contratti c.d. di licenza d'uso, così definiti proprio perché attraverso essi il titolare dei diritti sul software concede il solo uso del programma.
Questa prassi risponde all'esigenza di produttori e distributori di garantirsi diritti maggiori rispetto a quelli concessi dalla legge sul diritto d'autore per la vendita di copie di software e di evitare nel contempo il verificarsi dell'esaurimento della privativa.

Laddove alcune forme di un prodotto non siano distribuite con il relativo codice sorgente, deve essere chiaramente indicato il modo per ottenerlo; 3) modificabilità del prodotti derivati, nel senso che la licenza deve permettere modifiche e prodotti derivati, e deve permetterne la distribuzione sotto le stesse condizioni della licenza del software originale; 4) integrità del codice sorgente originale, poiché la licenza può impedire la distribuzione del codice sorgente in forma modificata, a patto che venga consentita la distribuzione dell'originale accompagnato da "patch", ovvero file che permettono di applicare modifiche automatiche al codice sorgente in fase di compilazione. La licenza deve esplicitamente permettere la distribuzione del software prodotto con un codice sorgente modificato. La licenza può richiedere che i prodotti derivati portino un nome o una versione diversa dal software originale; 5) distribuzione della licenza e cioè che diritti allegati a un programma devono essere applicabili a tutti coloro a cui il programma è redistribuito, senza che sia necessaria l'emissione di ulteriori licenze; 6) assenza di vincoli su altro software, poiché la licenza non deve porre restrizioni su altro software distribuito insieme al software licenziato. Per esempio, la licenza non deve richiedere che tutti gli altri programmi distribuiti sugli stessi supporti siano software Open Source; 7) neutralità rispetto alle tecnologie: la licenza non deve contenere clausole che dipendano o si basino su particolari tecnologie o tipi di interfacce.

Dal punto di vista della causa contrattuale, la concessione delle autorizzazioni anche a titolo gratuito propria delle licenze di software libero/Open Source, trova giustificazione nella soddisfazione di interessi diversi rispetto a quelli legati ad un corrispettivo economico, quali la diffusione del programma e l'affermazione del suo autore in seno alla comunità, oltre che allo sviluppo del movimento del software libero/Open Source e del modello di condivisione della conoscenza che ne costituisce il fondamento.

La questione della qualificazione delle licenze di software libero/Open Source risponde certamente ad esigenze pratiche quali l'identificazione della normativa applicabile sul piano internazionalprivatistico. Aspetto, quest'ultimo, che assume molta importanza in un contesto caratterizzato da un forte carattere transnazionale quale quello del software in questione.

Non va inoltre sottaciuto che un aspetto piuttosto controverso è rappresentato dalla qualificazione giuridica di tali licenze: negozi giuridici unilaterali o contratti. Secondo una parte degli autori che si sono occupati dell'argomento, le licenze sarebbero negozi unilaterali. Gli utenti del programma sarebbero dunque autorizzati a compiere le attività concesse dalla licenza solo nei limiti di quest'ultima, commettendo altrimenti una violazione dei diritti dell'autore.
Altri autori ritengono invece che alcune licenze, le quali contengono clausole che pongono delle limitazioni nel caso di modifica e ridistribuzione di software libero / Open Source (c.d. clausole di copyleft), necessitino di una forma di accordo tra licenziante e licenziatario.
Le clausole di copyleft creerebbero infatti delle obbligazioni a carico del licenziatario, imponendo la ridistribuzione del programma o di opere derivate da esso alle stesse condizioni della licenza originaria[26].

Altra questione collegata alla qualificazione, nel caso si propenda per la natura contrattuale delle licenze, è quella della valida formazione dell'accordo e di conseguenza dell'applicabilità delle disposizioni civilistiche in materia di contratti standard (art. 1341 I comma cod. civ.), clausole vessatorie (art. 1341 II comma cod. civ.) e tutela del consumatore (art. 1469-bis e seg. cod. civ.).
Caratteristica comune alle licenze di software libero / Open Source è quella di contenere clausole volte a escludere la fornitura di qualsiasi garanzia per il corretto funzionamento del programma e l'assunzione di responsabilità per i danni eventualmente provocati da quest'ultimo[27].
L'individuazione della legge applicabile alla licenza, quindi, è soltanto in grado di rassicurare l'autore iniziale ed i successivi creatori ed utenti garantendo la conformità dell'accordo con quella precisa legislazione. Questo tipo di sicurezza può favorire la più ampia divulgazione del software in questione[28].

[26] *In questo caso l'autore di un software derivato dal programma originario si troverebbe nella condizione di vedere limitato lo sfruttamento dei propri diritti di utilizzazione economica sull'opera derivata proprio a causa del copyleft sul programma originario.*
Tale limitazione dovrebbe perciò essere oggetto di consenso da parte del licenziatario, consenso che è uno degli elementi fondamentali del contratto.

[27] *A questo proposito una prima considerazione riguarda il divieto (ex art. 1229 cod. civ.) di inserire clausole che escludano o limitino la responsabilità per dolo o colpa grave o per violazione di norme di ordine pubblico.*

[28] *Nella Causa tedesca "Netfilter"7 , sebbene il Tribunale di Monaco abbia dichiarato valida la licenza GNU GPL, sono comunque state formulate copiose riflessioni dirette a verificare la conformità di tale licenza al diritto tedesco. La decisione non garantisce ovviamente che un'altra giurisdizione, che analizzi la GPL in funzione di un'altra legislazione nazionale, disciplinerà la questione allo stesso modo.*

Come già anticipato, nell'ambito dell'ordinamento giuridico italiano, il software è tutelato dalla legge sul diritto d'autore n. 633 del 1941 come forma di protezione giuridica delle opere creative. Alcuni esempi di opere creative sono le opere letterarie, musicali e cinematografiche, i disegni e le fotografie. Ciò che si protegge quindi non è un'idea, bensì l'espressione creativa di un'idea o di un'arte.

L'autore acquista sulla propria opera il diritto esclusivo di riproduzione, di esecuzione, di diffusione, di noleggio, di prestito, di elaborazione e di trasformazione. La normativa tutela, pertanto, anche i programmi per elaboratore come opere letterarie anche ai sensi della convenzione di Berna sulle opere letterarie ed artistiche.

In particolare tenendo conto di quanto previsto nella L. 633/'41 e s.m.i. come novellato dal D.lgs. 518 del 29, l'autore di software, con la licenza Open Source esprime semplicemente la propria volontà circa l'esercizio dei suoi diritti di esclusiva sul software. Questa "cessione" però, è accompagnata da clausole che impongono certe regole di utilizzo del programma; queste sia pure di un contenuto più permissivo rispetto alle tradizionali licenze d'uso, costituiscono, anche se non tutte, pur sempre l'esercizio dell'esclusiva d'autore.

In tale ambito è possibile, pertanto, distinguere la tutela garantita dal diritto d'autore e quella assicurata dalla licenza, in quanto contratto normativo unilaterale. La prima salvaguarda il licenziante, in quanto autore di software titolare di diritti esclusivi, nei confronti sia del licenziatario che disconosce la paternità dell'opera in capo al licenziante e l'integrità della stessa, sia nei confronti dei terzi, sub-licenziatari non rispettosi della licenza, poiché sconosciuta ad essi.

Ciò nonostante questa tutela è insufficiente per realizzare gli obiettivi dell'autore, poiché non consente un controllo diretto ed efficace sulla redistribuzione del software compiuta dal licenziatario.

A tale proposito interviene però la seconda tutela, apportata invece dalla licenza come contratto normativo unilaterale; questa infatti protegge l'autore, in quanto licenziante, nei confronti del licenziatario inadempiente, solitamente lo strumento utilizzato è la cd. Clausola di copyleft: che permette al licenziante di obbligare il licenziatario ad inserire il regime di licenza originaria e dunque le modalità di redistribuzione del software nei contratti con i terzi (ossia le sub-licenze[29]).

[29] *È quello che accade ad es. con la licenza GPL.*

Le clausole di copyleft creerebbero delle obbligazioni a carico del licenziatario, imponendo la ridistribuzione del programma o di opere derivate da esso alle stesse condizioni della licenza originaria.
In questo caso l'autore di un software derivato dal programma originario potrebbe trovarsi nella condizione di vedere limitato lo sfruttamento dei propri diritti di utilizzazione economica sull'opera derivata proprio a causa del copyleft sul programma originario.
Pertanto l'autore originario si può avvalere dell'azione prevista ai sensi dell'art. 156 della predetta Legge sul diritto d'autore in caso di situazioni che ledono i suoi diritti esclusivi di utilizzazione economica.
La clausola di copyleft consente, quindi, che il software rimanga libero ed il codice sorgente sia sempre aperto (essa è imposta ai sub licenziatari del software). La possibilità di imporre tale clausola deriva proprio dal copyright di cui è titolare l'autore dell'opera. Perciò le due clausole sono complementari[30].
Altro aspetto di particolare rilevanza, come già sopra accennato, è che il modello di licenza Open Source non vede limitato il suo ambito naturale di applicazione ai confini geografici, il che lascia aperte talune problematiche afferenti il coinvolgimento di più ordinamenti nazionali, in particolare riguardo al sistema di proprietà intellettuale.
Ad ogni modo è bene ribadire che non esiste per l'utente una libertà illimitata ed assoluta di disporre di questo particolare programma per elaboratore. Per poter compiere delle attività che ne presuppongono l'impiego, occorre pur sempre sottostare a delle previsioni contrattuali che contemplano obblighi.

[30] *Perciò la clausola di copyleft si attua in questo modo:*

- *l'autore crea l'opera;*
- *tutela del software dalle norme sul diritto d'autore;*
- *l'autore distribuisce il sw con licenza libera con clausole di copyleft;*
- *il software rimane libero anche se modificato e ridistribuito.*

5.2 Proprietà intellettuale e paternità morale dell'opera

La partecipazione numerosa da parte di una pluralità di soggetti, ricercatori, sviluppatori ecc. ai progetti creativi quali quelli afferenti il software, richiede una particolare riflessione soprattutto in considerazione dell'importanza di individuare i soggetti cui attribuire i diritti patrimoniali e morali di quanto realizzato, ciò anche al fine di verificare, in un secondo momento, quali facoltà ed in che limiti, vengono concretamente esercitati.

In tale ambito, peraltro, non sempre è facile individuare il contributo di ogni singolo partecipante all'opera tale da qualificarsi come opera autonoma, tuttavia all'interno delle comunità dei partecipanti ai progetti si sviluppo dei programmi per elaboratore, solo coloro che scrivono parti di codice che producono un'evoluzione migliorativa dell'opera dovrebbero vedere sorgere in loro favore la costituzione di diritti patrimoniali e morali d'autore. Difatti, ad esempio, la semplice segnalazione di errori non è considerata un'operazione rilevante, sebbene sia difficile tracciare una vera e propria linea di confine per identificare gli interventi partecipativi che contribuiscano al lavoro finale.

Come è noto il modello sottostante all'attività creativa che presiede alla nascita dei programmi per elaboratori è di tipo aggregativo, collettivo, collaborativo; esso vede la partecipazione di un numero molto elevato di soggetti impegnati nella modifica e nell'elaborazione del codice sorgente dei software a più livelli operativi. In prima battuta, si può certamente affermare che il vincolo che viene imposto ai partecipanti è quello di permettere l'accesso al codice sorgente anche in caso di redistribuzione delle modifiche sull'opera.

L'attività creativa è quasi sempre frutto di un'attività di elaborazione in quanto deriva sempre dal codice originale e si sviluppa secondo vari schemi organizzativi più o meno strutturati gerarchicamente, dove il ruolo di guida è assunto da pochi soggetti; vi sono però anche situazioni in cui non esiste alcun ruolo gerarchico ma si autoalimentano attraverso l'individuazione, di volta in volta, delle direttrici degli sviluppi. Va da sé che nell'ambito della partecipazione ad un'opera collettiva, tutti i soggetti hanno un forte interesse alla diffusione dell'opera sia nella versione "originale" che in quella futura, rielaborata e migliorata; ed è per tale ragione che l'ulteriore accordo sottostante riguarda proprio la destinazione dei singoli contributi diretti al fine creativo dell'opera finale.

Il modello di licenza Open Source impone, quindi, un regime circolatorio dei singoli contributi non diverso da quello a cui è sottoposta l'opera collettiva e quindi permette un libero accesso al codice sorgente anche sulla parte di programma isolatamente considerata.
Vi è da dire, inoltre, che il modello di licenza Open Source sembra compatibile con il principio secondo cui ciascun coautore può cedere ad altri la propria quota di partecipazione alla comunione, salva, comunque, l'indisponibilità delle facoltà morali. A tale ultimo riguardo è opportuno considerare anche un altro importante aspetto afferente i diritti morali che costituiscono quelle facoltà riconosciute all'autore per la tutela della sua personalità e per la protezione dell'apprezzamento sociale del suo lavoro creativo. I diritti morali si contraddistinguono per essere inalienabili, irrinunciabili ed imprescrittibili e sono, pertanto, sottratti, nei limiti che verranno esposti, alla disponibilità contrattuale.

Si può certamente osservare che i movimenti Open Source promuovono l'esatta attribuzione di paternità sui singoli elementi del codice, pertanto, sotto il profilo della rivendicazione di paternità della propria opera, le norme patrizie Open Source non sono in contrasto con la disciplina che tutela tale profilo ed anche le modalità attraverso le quali il nome dell'autore viene reso noto, in particolare se si considera che, secondo le varie forme d'uso, le indicazioni di tutti coloro che partecipano o che hanno partecipato al progetto figurano nei file che vengono inseriti nei programmi.
Nelle opere di tipo Open Source tutti gli sviluppatori del software sono consapevoli di operare in un determinato contesto, accettandone le regole di base. In sostanza, ognuno sa che il proprio lavoro è in continua evoluzione e che in qualunque momento altri programmatori possono cambiare il codice nei modi più svariati. In tal senso è facile comprendere come nelle ipotesi di concessione della facoltà di modifica, il campo di applicazione di eventuali lesioni derivanti da elaborazioni o alterazioni è piuttosto ristretto; peraltro, stando alle regole sul diritto d'autore, le modifiche che non hanno rilievo sono solo quelle accettate e conosciute dall'autore.

Per tale ragione, ad esempio, è certamente possibile affermare che una delle più concrete ipotesi di violazione delle regole delle licenze Open Source potrebbe derivare da una distribuzione di programmi senza libero accesso al codice sorgente, che potrebbe essere considerata una fattispecie di lesione dei diritti morali dell'autore[31].
Possono, pertanto, prevedersi soluzioni contrattuali che tendono a preservare il lavoro creativo che è alla base del software Open Source, garantendo che il codice sorgente non venga alterato o distribuito dai terzi che utilizzano il programma in forma diversa da quella originaria. Infatti, qualora l'autore del programma intenda salvaguardare i propri diritti morali sull'opera da interventi modificativi esterni che ne potrebbero intaccare la sua paternità originaria, pregiudicandola, potrà assicurarsi che il codice sorgente non venga alterato da terzi che si propongano di sviluppare programmi derivati. In questo caso, affinché il suo software possa considerarsi Open Source, l'autore dovrà mettere a disposizione di chiunque dei patch files, cioè delle sequenze di istruzioni di programma che consentono la modifica del software o la sua integrazione con altri diversi programmi per elaboratore.

I diritti patrimoniali e morali sull'elaborazione creativa appartengono all'autore di quest'ultima "nei limiti del suo lavoro" ex art. 7 della legge sul diritto d'autore e la sua utilizzazione economica è sottoposta al consenso del creatore dell'opera base ex art. 18, della medesima legge. L'autore dell'opera, pertanto, esprime il proprio consenso per determinare i termini delle successive modifiche o elaborazioni sull'opera base, che darebbero vita alle c.d. elaborazioni creative, nel senso che traggono i propri elementi espressivi da un'opera precedente. In questo ambito è pertanto necessario considerare in che modo le licenze Open Source interpretano il rapporto di derivazione tra opere creative e conseguentemente in che limiti tale derivazione garantisca l'interoperabilità tra programmi[32]. I programmi per elaboratore, infatti, sono sistemi "modulari" ed indipendenti, essendo offerti sul mercato in modo autonomo la cui naturale destinazione è quella di essere abbinati dall'utente finale, il che rende necessario che la produzione dei beni informatici sia agevolata da un'ampia compatibilità tra software.

[31] *Va da sé che ridistribuire in modo proprietario un programma Open Source è una contraddizione in termini, una negazione stessa della natura insita dell'opera e del sistema partecipativo sottostante.*

[32] *I modelli di sviluppo dei programmi proprietari non permettono l'accesso al codice sorgente, ragion per cui è il legislatore stesso ad offrire agli utenti la facoltà di modificare il codice sorgente al fine di poter permettere l'interoperabilità.*

È importante considerare che in linea generale il modello di licenza Open Source permette la rielaborazione, l'incorporazione e l'interconnessione di pezzi di codice da esso disciplinato con altri ed il più delle volte a condizione che l'intero programma derivato, ovvero l'altro software con il quale interagisca, adotti le medesime condizioni contrattuali.

5.3 Cenni sull'opera derivata ed il problema delle garanzie

Di non secondaria importanza diventa, quindi, la considerazione del concetto di opera derivata, argomento basilare per accertare le concrete possibilità di diffusione del modello di licenza Open Source, non solo per stabilire il regime giuridico delle opere, ma anche per definire come dovrebbe operare la garanzia dell'accesso all'interoperabilità tra software di diverso genere, quando siano regolati da tale modello di licenza[33]. Attraverso l'Open Source si incentiva la disponibilità per chiunque di conoscere il codice sorgente, poiché solo attraverso di esso è possibile modificare il programma per elaboratore e creare opere derivate che contribuiscono all'evoluzione informatica, culturale della collettività.
Va ricordato ancora che nel caso in cui il software Open Source sia incluso in un altro software per dar vita ad un prodotto diverso da distribuire al pubblico, la parte di software Open Source, di norma, dovrà essere sempre regolata dalla licenza originaria.
Inoltre, nel caso in cui l'utente includa il software Open Source all'interno di un altro programma non Open Source e decida di commercializzare il prodotto ottenuto, egli dovrebbe essere tenuto a manlevare e garantire tutti i soggetti che hanno contribuito alla realizzazione del software Open Source dagli eventuali danni causati a terzi derivanti dall'impiego di tale nuovo programma.
A tale ultimo riguardo, per quanto attiene le garanzie, i modelli di licenza Open Source sono privi di obblighi per il soggetto concedente. Il software viene concesso in uso nello stato di fatto in cui si trova (*"as it is"*) ed il licenziante non garantisce che possa creare delle disfunzioni o dei malfunzionamenti con le apparecchiature o con altri programmi preesistenti; il programma viene, dunque, rimesso nella libera disponibilità dei terzi che si assumono ogni rischio derivante dal suo impiego.

[33] *Secondo alcuni sostenitori, l'interpretazione di opera derivata accolta dalle licenze Open Source è illegittima poiché non è sorretta dalle regole sul diritto d'autore in quanto estende la protezione monopolistica stabilita dalla legge su un'opera di ingegno ed impedisce l'interoperabilità tra programmi per elaboratore, il che lascia concludere che in materia di software non sarebbe corretto estendere contrattualmente un criterio di opera derivata che vada oltre l'ambito legislativamente stabilito.*

Allo stesso modo, secondo lo schema ricorrente delle licenze, la responsabilità, per qualunque titolo, derivanti dall'utilizzo di software Open Source, non potranno essere attribuite al soggetto che ha concesso al pubblico il diritto d'uso del programma.
Pertanto, in linea di massima, la licenza Open Source esclude ogni forma di garanzia e di ipotesi di responsabilità risarcitorie che a qualunque titolo dovessero essere avanzate dagli utenti. Attraverso questa clausola l'originario titolare del programma viene esonerato da ogni tipo di responsabilità derivante dall'uso del bene, precisando, come già anticipato, che tutti i rischi derivanti da un eventuale malfunzionamento del programma dovrebbero gravare esclusivamente sul suo utilizzatore, il quale non potrà, secondo quanto stabilito nello schema negoziale, rivalersi nei confronti del soggetto concedente.
Per ciò che concerne il nostro ordinamento giuridico, e soprattutto secondo quanto previsto dall'art. 1229 del codice civile, tale esonero che definirei totale, come sopra descritto, non dovrebbe ritenersi efficace e dunque in grado di assicurare la certezza dell'esenzione da responsabilità per il soggetto concedente il software Open Source; si tratterebbe comunque di valutare se, vertendo nell'ambito di uno schema negoziale a titolo gratuito, la responsabilità del licenziante per eventi collegati all'uso del programma, possa ritenersi affievolita e valutata con minor rigore[34].
Diverso è il caso invece in cui l'utente del programma Open Source decida, in violazione delle disposizioni statuite nella licenza del programma che ha tacitamente accettato utilizzando il software concesso, di apportare delle modifiche al programma stesso e di distribuirlo commercialmente al pubblico senza rivelare ai terzi il codice sorgente, tradendo così i principi basilari della filosofia Open Source.
Ovvio che una tale condotta configura un inadempimento che dovrebbe essere sanzionato con la risoluzione del contratto che produce la retrocessione delle ampie facoltà concesse all'utilizzatore, il quale non avrà più alcun diritto di usare il programma per elaboratore.
Tuttavia ci si chiede quali siano le conseguenze per i terzi che hanno acquistato i diritti dall'utente nei cui confronti è stato revocato il diritto di utilizzo da parte dell'originario titolare dei diritti e cioè capire se anch'essi versino in una situazione di illegittimità che travolge anche il loro contratto. In tal caso si può dire comunque che il bene trasferito è stato realizzato in violazione de diritti esclusivi riconosciuti dalla legge al suo autore.

[34] *Naturalmente dovrebbero esulare da tali considerazioni le ipotesi in cui il soggetto concedente il programma Open Source non abbia agito in buona fede e cioè abbia deliberatamente prodotto il risultato dannoso con dolo.*

Di regola il licenziatario può duplicare e distribuire - anche dietro compenso - il software a patto che garantisca ai terzi l'accesso al medesimo codice sorgente;
oppure che può modificare il sorgente ed ottenere così una c.d. "opera derivata" che può vendere a condizione che rilasci il sorgente anche delle porzioni modificate od aggiunte.
Se il licenziatario non si conforma a queste condizioni, egli sta travalicando i limiti della licenza concessagli e potrebbe, pertanto, essere perseguibile dall'autore/detentore dei diritti/licenziante per violazione dei diritti di esclusiva garantitigli dalla legge sul diritto di autore.

5.4 Concetto di licenza d'uso e licenze Open Source

È utile ribadire che per quanto riguarda i diritti di proprietà sul software è il caso di distinguere fra proprietà del software in quanto bene materiale (supporto) e proprietà del software in quanto opera dell'ingegno. La seconda rileva per il diritto d'autore (è detta infatti 'proprietà intellettuale) e si fonda sul meccanismo dei diritti esclusivi licenziati; la prima invece rimane nella sfera d'influenza del diritto privato, risolvendosi in un normale contratto di compra-vendita.
La piena proprietà del supporto e degli "accessori" del bene è indiscussa; ma come abbiamo visto, a creare maggiori perplessità è invece la portata dei diritti di proprietà intellettuale derivanti dai diritti esclusivi di sfruttamento economico in capo all'autore dell'opera.

Come più volte detto, la formula contrattuale maggiormente adottata nella prassi per consentire il trasferimento delle facoltà di utilizzo del software è la licenza d'uso, strumento attraverso il quale una parte concede ad un'altra il diritto di utilizzare il software per un certo periodo di tempo e verso un determinato corrispettivo.

Il licenziatario, quindi, non diventa proprietario del bene ma ne acquista la facoltà di utilizzo o sfruttamento secondo i termini e le condizioni stabilite nel contratto. Al di là di ogni considerazione sulla qualificazione giuridica del contratto di licenza, in linea di massima e per ciò che riguarda il software Open Source, per effetto di una precisa volontà del soggetto originario titolare dei diritti, il bene è svincolato dal regime di protezione e dai correlati diritti di privativa accordatagli dalla normativa a tutela del diritto d'autore.

Tuttavia, le licenze denominate "libere" possiedono ognuna le proprie caratteristiche da adattarsi in funzione delle caratteristiche specifiche del progetto.
In questa sede ci limiteremo ad analizzare brevemente taluni aspetti peculiari di alcune delle licenze più diffuse con particolare riferimento alla GPL, individuando differenze più o meno evidenti che modificano i diritti e le obbligazioni concesse al licenziatario. Resta fermo che la trattazione non è affatto esaustiva in primo luogo perché deriva da testi tradotti e secondariamente poiché trattasi di una breve disamina degli aspetti principali. Pertanto, per una lettura completa delle licenze si rimanda ai singoli testi pubblicati.

La GPL (General Public License)

Già dalle prime pagine di tale licenza si può notare che, a conferma di quanto detto sul fondamento giuridico del copyleft, si tratta di un documento che basa tutta la sua essenza ed efficacia sul copyright, a differenza di ciò che si può pensare comunemente. Lo si coglie pure dal fatto che lo stesso testo della GPL viene dichiarato inequivocabilmente tutelato da copyright da parte della Free Software Foundation Inc. L'essenza del concetto di copyleft è inteso solamente come 'permesso di copia': "chiunque può copiare e distribuire copie letterali di questo documento, ma non ne è permessa la modifica." Quest'annotazione preliminare è ovviamente resa necessaria al fine di evitare che il destinatario della licenza ne possa stravolgere la funzione primaria cambiandone i contenuti e alterandone la disciplina a suo piacimento; quindi, sotto questo aspetto, unico soggetto autorizzato a rilasciare modifiche o addirittura nuove versioni è la FSF, in posizione di detentrice dei diritti di copyright.

In tale licenza viene inoltre delimitato l'ambito d'azione, precisando quali attività relative al programma essa copra: l'esecuzione, la copiatura, la distribuzione e la modifica. Essa regola inoltre le condizioni che rendono lecita la distribuzione e la copia del programma: si ribadiscono in pratica i paradigmi del copyleft relativi all'obbligo di trasmettere (mediante "un'appropriata nota sul copyright") gli stessi diritti ricevuti.
Vi è poi una specifica sezione che scende nello specifico del concetto di modifica e delle sue delicate implicazioni giuridiche; si entra infatti nel terreno accidentato che porta alla configurazione del software nato da modifica di altro software come 'opera derivata ai sensi della legge sul copyright'.

Si rammenta infatti che è proprio la possibilità di modifica che distingue un software libero e Open Source da un normale software commerciale distribuito gratuitamente; ne è assicurata quindi la disponibilità del codice sorgente e stabilisce le condizioni della sua diffusione. Inoltre vi è una parte in cui è contemplata la validità della licenza e si presenta come una sorta di "norma di chiusura" con la quale si escludono dalla copertura della GPL (con un meccanismo di decadenza automatica dai diritti licenziati) le attività diverse da quelle esplicitamente previste dalla licenza.
Di interesse sono inoltre alcuni aspetti pratici della distribuzione di prodotti sotto GPL: l'acquirente di un programma di questo tipo è libero di non accettare i termini della licenza (e quindi di non usare quel prodotto), ma qualora lo modificasse o lo distribuisse ciò significherebbe l'accettazione tacita e automatica della licenza, e di conseguenza l'assunzione degli obblighi di trasferimento dei diritti licenziati. In pratica chi non vuole accettare le condizioni della GPL, può benissimo lavorare su un altro tipo di software; qualora però volesse un prodotto di 'software libero', deve accettarne in toto le peculiarità giuridiche e di distribuzione.

Molto interessanti risultano invece due sezioni che prevedono alcune cautele per ovviare al rischio di eventuali modifiche ai termini della licenza imposte da statuizioni di autorità giudiziarie o da peculiari regimi legislativi. Nel caso tali obblighi "istituzionali" dovessero contrastare con i dettami fondamentali della GPL, allora il prodotto in questione non potrebbe più essere distribuito.

La normativa afferente i diritti di copyright sul testo stesso della licenza (vedi poco sopra) si configura come una norma di apertura, dato che con essa la Free Software Foundation si riserva la possibilità di pubblicare revisioni o nuove versioni della GPL (identiche nello spirito ma più precise nei dettagli), "al fine di coprire nuovi problemi e nuove situazioni." Importante è inoltre considerare che la licenza prevede la possibilità di armonizzare i differenti regimi giuridici nel caso in cui un programma coperto da GPL sia incorporato in altri programmi liberi coperti però da altri tipi di licenze.

Infine vi sono altre due sezioni che rappresentano invece la parte più determinante dal punto di vista del diritto civile in generale, trattandosi di una "manleva da responsabilità" per coloro che hanno partecipato allo sviluppo e alla distribuzione del software libero. Infatti si leggerebbe che "l'intero rischio concernente la qualità e le prestazioni del programma è dell'acquirente"; altra sezione contempla invece l'assenza di responsabilità per danni ("generici, speciali o incidentali"), menzionando alcune fattispecie esemplari di "danni che conseguono dall'uso o dall'impossibilità di usare il programma" (perdita dei dati o difficoltà nell'interazione con altri programmi).
È opportuno quindi che chiunque decida di usare un software libero sia informato chiaramente di tale sua natura.

Sembrerebbe comunque che le prime applicazioni della GPL abbiano fatto emergere un problema di tipo pratico, derivante proprio da una caratteristica specifica e peculiare della licenza: la sua 'viralità'. Cioè la sua capacità di trasmettere presso un numero indefinito di utenti una certa serie di diritti e responsabilità; "in pratica tutto il codice sviluppato per funzionare in associazione con software GPL deve essere a sua volta coperto da GPL", pena la caducazione dell'intera licenza: "I programmi coperti da questa Licenza Pubblica Generica non possono essere incorporati all'interno di programmi non liberi.

Bisogna comunque tenere presente che il software è solitamente un insieme fittissimo di altri software. Perciò è possibile che nel modello "a cascata" di ridistribuzione del software libero, grazie alla disponibilità del codice sorgente completo e alla possibilità di modifica garantite dalla GPL, qualche sviluppatore voglia scorporare il pacchetto software e rielaborarne solo una parte, oppure decida di fare interagire un software di derivazione libera con un software proprietario.

È il tipico caso delle librerie: esse sono delle "raccolte di funzioni" precompilate" a disposizione del computer, di cui possono "servirsi" i programmi veri e propri che vengono eseguiti, pur non facendo realmente parte dei programmi stessi.

In questo modo può facilmente verificarsi che un software libero si appoggi su un libreria di origine proprietaria e viceversa. Quindi un'eccessiva rigidità (come per certi versi è quella della GPL) può risultare controproducente allo sviluppo di software libero: uno sviluppatore GNU che volesse creare una libreria sotto i parametri del copyleft saprebbe che poi molti utenti di software proprietario potrebbero avere dei problemi ad usarla. Fu per questo che fu stilata una seconda licenza, chiamata LGPL, Library General Public License.

La LGPL (Lesser General Public License)

Essa è dunque una licenza apposita per il caso delle librerie, che si presenta come una versione della GPL alleggerita però di alcune restrizioni (per questo infatti è stata recentemente rinominata 'Lesser GPL', ovvero 'GPL minore). Ovviamente la FSF "diffida" chiunque ad usare questa licenza per software che non siano librerie, pena l'esclusione del programma così distribuito dalla "famiglia" del software libero.

La BSD (Berkeley Software Distribution) license

Si tratta di una licenza piuttosto scarna, senza componenti ideologiche o programmatiche e parti esemplificative, i cui termini risultano sintetici e essenziali e la cui applicazione si risolve nell'inserimento di una breve nota standard da inserire nei file che s'intendono tutelare con la licenza. La nota (detta 'template) deve riportare il nome di chi detiene il copyright, l'organizzazione a cui egli appartiene e l'anno di realizzazione. Il testo della licenza prosegue poi specificando che, del software così tutelato, sono permesse la ridistribuzione e l'utilizzo in forma sorgente o binaria, con o senza modifiche, ma solo se vengono rispettate tre condizioni:

- le ridistribuzioni del codice sorgente devono mantenere la nota sul copyright;
- le ridistribuzioni in forma binaria devono riprodurre la nota sul copyright, l'elenco delle condizioni e la successiva avvertenza nella documentazione e nell'altro materiale fornito con la distribuzione;
- il nome dell'autore non potrà essere utilizzato per sostenere o promuovere prodotti derivati dal software licenziato, senza un apposito permesso scritto dell'autore.

In fine si trovano gli avvertimenti (disclaimer) sull'assenza di garanzia e sulla manleva da responsabilità di, che ricalcano grossomodo lo schema dei loro corrispondenti all'interno della GPL.
È il caso invece di mettere in luce la caratteristica più problematica di tale licenza, ovvero la possibilità di usare il codice da essa tutelato per sviluppare software proprietari: aspetto che la pone al di fuori del paradigma di 'copyleft', poiché non è garantita la trasmissione all'infinito dei diritti da essa concessi, i quali si interrompono nel momento in cui il codice diventa proprietario.

La MPL (Mozilla Public License)

Brevemente, in essa più che in altre licenze anteriori, sono presenti alcune rilevanti clausole mirate a tutelare gli sviluppatori del progetto Mozilla dalla malafede di coloro che avrebbero potuto approfittare della elasticità della licenza, inserendo tacitamente parti di codice coperte da brevetto, per poi rivendicarne per vie legali il pagamento. "La licenza […] prescrive che l'azienda o l'individuo che contribuisca con codice al progetto rinunci ad ogni possibile pretesa a diritti di brevetto a cui il codice potrebbe dare adito." La licenza Mozilla tuttora usata per i browser Netscape, resta incompatibile con la GPL - ma può essere qualificata come licenza Open Source.

6. Modelli organizzativi e di business

6.1 Il ruolo della community Open Source

Aspetti motivazionali

Nel periodo successivo alla creazione della Free Software Foundation di Stallman ed all'applicazione dei principi del software libero nel mercato del business, ci si è cominciati a porre domande sul come e sul perché il modello organizzativo Open Source funzioni; ad oggi, a questo quesito non è stata data ancora una spiegazione esauriente.

Un primo tentativo di analisi è stato fatto analizzando l'approccio motivazionale.

La motivazione comprende diversi fattori ognuno dei quali condiziona l'atteggiamento ed il comportamento dell'individuo verso task specifici. Esistono tre diversi tipi di motivazione: **intrinseca**, **estrinseca** e la **demotivazione**. La motivazione intrinseca è il fare un'attività solo per la soddisfazione che ne deriva; la motivazione estrinseca è il fare perché c'è una qualche forma di obbligo. La demotivazione invece emerge nel momento in cui si percepisce di non essere capace o, più in generale, di non volere svolgere un'attività per svariate ragioni.

Quindi, mentre una comunità di individui è intrinsecamente motivata attraverso l'esplorazione, la sfida e l'interesse per la creatività, altri sono estrinsecamente motivati da tangibili o intangibili benefici che possono derivare da interventi esterni. Non è comunque detto che un individuo sia unicamente motivato in una direzione intrinseca o estrinseca. Task particolari, infatti, possono scatenare entrambi i meccanismi psicologici.

Nel mondo reale del business, fattori esterni come ad esempio gli incentivi sono ritenuti fondamentali nel processo motivazionale delle persone. Al contrario, dal punto di vista psicologico, troppa influenza esterna è vista come pericolosa per lo sviluppo della motivazione. Infatti gli interventi esterni in termini di "ricompensa" tenderebbero ad abbassare la motivazione intrinseca e quindi a ridurre l'efficacia nelle attività.
Ciò dipende anche da modi e modalità dell'intervento esterno. Ad esempio un intervento in termini non pecuniari ma culturali (ad esempio uno stage) può avere effetti benefici.

Al contrario ricompensare qualcuno per il completamento di un task può risultare dannoso in quanto in certi individui può far percepire una senso di controllo esterno che minerebbe la propria creatività. In questo senso la modalità di erogazione della ricompensa dovrebbe essere normata in maniera adeguata per ridurre aspetti legati alla standardizzazione ed al controllo.[35]

Nell'ambito del movimento Open Source, è ormai prassi consolidata considerare la **motivazione intrinseca** come il più forte driver di partecipazione. Ancora più spesso l'interesse specifico nello sviluppo e programmazione di nuovo codice è di per sé sufficiente ad attirare contribuitori intorno ad un progetto. L'attività di realizzazione del codice determina uno stato di "esperienza fluida" che è un misto di gioia, creatività e sfida. Una forma di auto-regolamentazione di queste comunità è applicata tramite meccanismi meritocratici: più un membro ha già contribuito ad un progetto, più egli può determinare quali caratteristiche saranno integrate nell'applicazione e controllare l'evoluzione del progetto. Inoltre, l'interazione e l'altruismo verso i membri può essere un fattore di partecipazione senza alcun apparente vantaggio per se stessi. Del resto, l'ideologia Open Source gioca un forte ruolo per la comunità ovvero mantenere e condividere il codice sorgente per metterlo a disposizione di chiunque. Strettamente collegato a questo aspetto è il concetto di reciprocità fra i diversi membri ovvero la contribuzione come un "dono" alla comunità che qualcuno sente di elargire perché altri possano estendere e sfruttare il proprio software.

I membri della comunità Open Source possono essere motivati anche da **fattori estrinseci**. Ad esempio "la necessità" da parte di qualcuno può indurre lo sviluppo di una patch per un bug esistente o per la mancanza di una feature. Gli sviluppatori possono essere guidati da una forma di guadagno di reputazione o dal desiderio di dimostrare il loro talento a propri collaboratori o ancora per far decollare la propria carriera professionale. Per evitare tempo perso, gli sviluppatori estrinsecamente motivati selezionano unicamente progetti di ampia risonanza, in grado di condurli velocemente verso i propri obiettivi.

[35] *Bénabou, R. & Tirole, J. 2003. Intrinsic and Extrinsic Motivation. Review of Economic Studies, 70(244): 489-500. Bitzer, J., Schrettl, W., & Schröder, P. J. H. 2007. Intrinsic Motivation in Open Source Software Development. Journal of Comparative Economics, 35(1): 160-169.*

Nei progetti Open Source la collaborazione è fatta generalmente su base volontaria senza ricevere un pagamento diretto. In questo senso, molti esperti si stanno domandando se la ricompensa e l'incentivo pecuniario hanno un effetto sulla motivazione degli sviluppatori e qual è il ruolo delle regole esistenti di sovvenzionamento della comunità. La tematica è attualmente molto dibattuta. Stanno comunque emergendo delle linee di pensiero piuttosto precise[36], avallate anche da ricerche fatte su campioni statisticamente significativi di comunità Open Source esistenti. In particolare:

1. gli sviluppatori a cui viene offerta una ricompensa per il lavoro sul progetto Open Source sviluppano una minore autostima ed interesse rispetto agli sviluppatori che lavorano su base motivazionale intrinseca. Quindi offrire un premio in denaro per il completamento di un task può avere un effetto controproducente sulle motivazioni intrinseche;
2. questo effetto è ridotto nel momento in cui gli sviluppatori appartengono a comunità in cui già esistono delle norme di remunerazione come ad esempio la sponsorship globale da parte di una azienda commerciale. In questo caso l'esistenza di adeguate politiche "strutturali" di incentivazione all'interno della comunità esercita un effetto molto positivo sulle motivazioni estrinseche che compensa l'effetto sulle motivazioni intrinseche.

Riassumendo, se la community ha già condiviso delle politiche di sovvenzionamento economico, una ricompensa esterna non sarà percepita come un intervento di controllo e quindi non eroderà la motivazione intrinseca; i contributori, in linea con le loro forti motivazioni estrinseche, selezioneranno maggiormente progetti con ampie sponsorship e la cultura meritocratica Open Source potrà associare ricompense adeguate a membri il cui contributo sia particolarmente significativo ed innovativo. Queste ricompense ad hoc in progetti specifici potrà attrarre ulteriori sviluppatori che avranno forti stimoli a partecipare alla community.

[36] *Alexy, Oliver; Leitner, Martin. Norms, Rewards, and Their Effect on the Motivation of Open Source Software Developers. 2007.*

Quindi, secondo questa impostazione, le aziende dovrebbero trattenersi dal partecipare a progetti Open Source i cui contributori condividono volutamente, secondo impostazioni prettamente ideologiche, politiche di NON remunerazione/incentivazione.

Aspetti strutturali

Un **community customer**[37] è un individuo o un' organizzazione che vuole diffondere un prodotto Open Source senza avere un impegno diretto nei futuri sviluppi del prodotto, e che promuove la partecipazione alla community stessa per assicurarsi che l'evoluzione del prodotto sia coerente con i propri obiettivi. Il community customer non è uno sviluppatore. Il termine **customer** individua il ruolo di chi offre risorse, economiche e non, e riceve servizi o prodotti in contropartita. In genere qualunque organizzazione che opera la diffusione di software Open Source dovrebbe essere pienamente informata sulle regole ed i processi che operano nella community. È un errore di valutazione considerare il software Open Source come una scatola chiusa identificata attraverso un canale di help-desk.

Riguardo **il processo di sviluppo**, la partecipazione attiva alle community può essere una seria opportunità per le organizzazioni che hanno necessità di sviluppo di componenti software operative; i vantaggi indiretti possono essere tali che persino investimenti considerevoli nello sviluppo possono essere ripagati. Ad esempio IBM e Oracle sono i principali esempi di aziende che esplicitamente sostengono finanziariamente progetti di sviluppo di infrastrutture IT Open Source per raccogliere i benefici di migliori infrastrutture su cui costruire prodotti e servizi proprietari.
Le organizzazioni sostenute dai contribuenti, come l'olandese SurfNet, richiede che qualunque software sviluppato con i loro fondi sia di tipo Open Source[38]. Questo rende etico l'utilizzo del fondo ed evita che il prodotto vada "end of life " quando il progetto originario di sviluppo termina. Esistono, infatti, esempi di prodotti proprietari sviluppati da aziende commerciali su concessione governativa che sono stati abbandonati subito dopo la conclusione del progetto, indipendentemente dal successo riscontrato.

[37] *Community Customers, pag. 514-519, Handbook of research on OSS – 2007 – Information Science Reference*

[38] *http://www.surfnet.nl/*

Quindi, la partecipazione attiva nel processo di sviluppo può essere un modo efficace per le organizzazioni, per assumere un ruolo di customer della community.

Riguardo **il processo di procurement,** le differenze di approccio rispetto al software proprietario sono molteplici. Alcune di queste differenze risiedono nel fatto che il software Open Source permette di disporre pubblicamente di molte più informazioni rispetto al software proprietario; altre nella mancanza nel caso Open Source di organizzazioni di marketing o di vendita che vadano alla ricerca di nuovi clienti.
Sebbene non ci siano costi di licenza associati ai prodotti Open Source, l'attività di procurement non è comunque a costo zero. Infatti la ricerca e la selezione di prodotti Open Source per un cliente può essere economicamente più onerosa rispetto al software proprietario, con un completo spostamento dell'effort dal fornitore al cliente. Attività quali la redazione di proposte, dimostrazioni, sviluppi di demo, coinvolgimento di specialisti, sono oneri che in un assessment di prodotti Open Source sono assunti dal cliente (anche se parte di queste attività può derivare dalle esperienze documentate della community). Al contrario, per il software proprietario, i costi di procurement vengono spesso ribaltati sul cliente solo in caso di acquisto del prodotto (no-purchase-no-pay option).
La community inoltre dovrebbe assumere un ruolo importante nel prendere decisioni in ambito procurement. Vi sono diverse teorie in proposito[39] che suggeriscono quali informazioni e metriche dovrebbero essere rese disponibili per supportare il processo decisionale di scelta dei prodotti. Ad esempio la tracciatura dell'attività della community dovrebbe indirizzare delle liste ristrette di prodotti mentre l'analisi dei risultati dei test uniti all'esperienza utente, dovrebbero costituire un fattore importante nella decisione finale. È auspicabile che in un processo standard di procurement Open Source, i risultati delle analisi svolte dai singoli clienti siano donate alla community indipendentemente dall'adozione del prodotto. In questo modo si arriverebbe ad un modello economico sostenibile ed equilibrato in cui le informazioni di mercato siano disponibili a tutto il mercato evitando i consueti comportamenti da parte dei vendor di soluzioni proprietarie che utilizzano grossi budget marketing per superare concorrenti minori sfruttando l'indisponibilità sul mercato di analisi di prodotto libere ed oggettive. Tali "donazioni"

[39] *Van de Berg, K.(2005) Finding Opne Options. Master's thesis, Tilburg University. http://www.karinvandenberg.nl/Thesis.pdf*

non possono essere mandatorie e proprio l'impostazione culturale dell'Open Source sarà di guida e di stimolo.
Riassumendo, nel caso Open Source un'organizzazione inizialmente ha la necessità di investire molto del suo tempo e delle sue risorse anche se la community sarà incline per sua natura nel fornire supporto, ancor più se il cliente condividerà con la community l'esperienza accumulata durante il processo di procurement. Se verrà deciso l'utilizzo del prodotto Open Source, l'organizzazione verrà subito gratificata per la sua donazione non dovendo pagare per l'impegno di risorse di marketing o di vendita a spese di un investimento nella reputazione della community (la reputazione è un elemento fondamentale per la community perché consente di essere via-via contattati da altri potenziali clienti).

Anche per il **processo di diffusione** le regole della community seguono il medesimo approccio del processo di procurement. Esperienze documentate, best practice, linee guida sono risorse utili per qualunque progetto di diffusione. La disponibilità di tali informazioni è tanto maggiore quanto è maturo il prodotto Open Source. Altre informazioni di supporto possono derivare dalla stessa rete dei partecipanti; è un fatto culturale più che di processo. A fronte di domande o problemi emersi in un progetto di diffusione, la community rende disponibili metodi rapidi per ottenere supporto grazie all'esperienza di persone ed organizzazioni che hanno già affrontato situazioni analoghe. Il modello funziona tanto più se l'organizzazione che implementa il progetto di diffusione ha ottenuto un alto livello di visibilità nella community in quanto si genera un comportamento virtuoso incline all'aiuto reciproco. Ovviamente non è un problema di esperienza ma di attitudine. Se qualcuno aiuta un altro che viene dopo di lui, questo qualcuno sarà in genere aiutato da chi è venuto prima di lui.
L'organizzazione, nel dimensionare il budget per un progetto di diffusione, dovrebbe prevedere di assegnare parte delle risorse per documentare l'esperienza, per poi riportarla nella community.

Per quanto riguarda **il processo di manutenzione,** in genere si assiste ad un supporto efficiente con risposte rapide e concrete a domande urgenti.

La manutenzione software Open Source non è molto diversa da quella relativa a software proprietario. In entrambi i casi c'è la necessità di disporre in tempi brevi di fix per malfunzioni e scoperture di sicurezza. È compito della community fornire le patch mentre le organizzazioni che utilizzano i prodotti Open Source devono impegnarsi nel mantenere aggiornato l'ambiente operativo al pari dei prodotti proprietari.

Un elemento specifico dei prodotti Open Source è la tendenza ad un più vivace ciclo di vita delle patch. Nei prodotti proprietari intercorrono in genere diversi mesi tra i rilasci di nuove release ed il servizio di aggiornamento viene contrattualizzato attraverso la struttura di vendita; l'approccio adottato è quello relativo alla percezione dell'urgenza in base alle problematiche ed esigenze dei clienti. I prodotti Open Source possono offrire lo stesso approccio ma non direttamente verso il cliente. La community attua un processo di "active self monitoring" tendenzialmente on-line con disponibilità di update automatica. I prodotti Open Source tendono ad avere una maggiore rapidità nel responso ad una segnalazione, grazie all'attività di vigilanza della community, specialmente per i problemi inerenti la sicurezza in cui si cerca di minimizzare i ritardi. Questo fatto determina un interesse primario da parte delle organizzazioni che utilizzano i prodotti Open Source a tenere alta l'attenzione sui problemi riportati e sulle patch disponibili e contribuisce attivamente nel riportare i problemi percepiti alla community.

Per il software proprietario spesso si tende a guidare tale processo sulla base dei rapporti con alcuni grandi clienti mentre quelli minori non sono altrettanto considerati. Nelle community Open Source tale situazione non si manifesta in quanto chiunque può stabilire contatti diretti con gli sviluppatori. Non si può però negare che anche la dimensione delle organizzazioni che utilizzano il prodotto e che contribuiscono alla community possa influire sulle priorità di sviluppo assegnate. Questa però è più un eccezione che una regola in quanto non è detto che i maggiori contributi in genere arrivino a clienti di grosse dimensioni.

In conclusione, qualunque organizzazione che prevede di utilizzare un prodotto Open Source deve investire significativamente nella relativa community.

Non esistono indicazioni certe sull'entità dell'investimento sebbene il costo nel partecipare alla community dovrebbe ridursi nel tempo e mantenersi al di sotto del costo totale delle licenze di un prodotto proprietario. Ciò che appare comunque evidente è che qualunque approccio che miri a considerare il software Open Source come un affare in quanto a costo zero, si dimostra sempre sostanzialmente errato.

6.2 Strategie di costo e profittabilità di un progetto Open Source

L'evoluzione dei modelli di business relativi al software Open Source va ricercata nella storia del movimento stesso; a partire dagli anni '80 si è cominciato ad assistere ad una graduale ma costante conflittualità fra i due approcci legati alla proprietà del software ed allo sviluppo cooperativo. La Free Software Foundation formalizzava lo sviluppo cooperativo attraverso il progetto GNU di realizzazione di un sistema operativo libero. I concetti alla base del software free sono riassunti nelle quattro libertà di Stallman, che per comodità riportiamo:

- libertà di eseguire il software;
- libertà di modificare il codice sorgente;
- libertà di distribuire il software;
- libertà di distribuire versioni modificate.

Il software da un lato non doveva essere soggetto a regime di copyright altrimenti sarebbe venuto meno lo sviluppo cooperativo, dall'altro non poteva non essere regolamentato per evitare che venisse in regime di libertà incluso all'interno di software proprietario. Venne così sviluppata l'idea del *copyleft* ossia proteggere le liberta del software; (i termini del copyleft di un programma vengono mantenuti anche per le versioni modificate dello stesso). Per realizzare tale idea fu sviluppata la GPL (General Public License).

Alternativamente al movimento Open Source di Stallman si crearono movimenti che miravano a diffondere il software Open Source in maniera meno radicale; venne fondato l'OSI che aveva come idea di base la promozione dell'Open Source Definition, un insieme di termini legali per la licenza più assimilabili ad un utilizzo commerciale. OSI ha registrato un marchio e sviluppato una serie di tipologie di license (OSI license) aprendo di fatto l'interesse del mercato.

Il nuovo modello di sviluppo (introdotto dal progetto Linux) è stato considerato il miglior modo per guidare un rapido e qualitativo avanzamento del software. Lo sviluppo cooperativo non era più solo un fatto ideologico ma piuttosto un qualcosa da utilizzarsi in progetti di tipo commerciale.

Una delle maggiori critiche verso il modello di business Open Source sta nella libera distribuzione, consentita dai termini di licenza. La richiesta di un canone non è generalmente fattibile in quanto il compratore potrebbe rivendere a sua volta il software creando un meccanismo perverso che porterebbe alla diminuzione dei tassi di utilizzo del prodotto anche da parte degli sviluppatori. Non è pertanto pensabile basare la logica di business sui canoni di licenza. È possibile però utilizzare il software Open Source come parte di un altro prodotto commerciale quale ad esempio un pacchetto, hardware o servizi; questo approccio non è esente da rischi per le caratteristiche di unicità della licenza Open Source.
Molte aziende che operano nel mercato con prodotti Open Source sono in qualche modo legate alle community per lo sviluppo del software che costituisce la loro offerta e per il supporto. Comunque, le community sono fuori da un loro possibile in quanto libere da accordi contrattuali.

In letteratura[40] sono proposti tre possibili approcci che le aziende possono adottare nel rapporto con le community:

- **approccio parassita:** l'azienda si focalizza sui propri interessi anche a discapito dei possibili danni prodotti alla community; comunque è esclusa una diretta influenza sulla attività di sviluppo poiché l'azienda non condivide norme, valori o regole con la community;
- **approccio commensalistico:** l'azienda ricerca benefici dalla community senza assumere una posizione di ostilità; in questo caso è possibile anche se difficile una qualunque forma d influenza;
- **approccio simbiotico:** l'azienda partecipa in modo cooperativo allo sviluppo condividendo norme e valori, cercando però di influenzare lo sviluppo verso i propri obiettivi.

[40] *Dahlander e Magnusson (2005): "Relationships between Open Source software companies and communities".*

Il modello di business

Recenti studi[41] identificano un numero limitato di elementi caratterizzanti i diversi modelli di business. Tali elementi includono l'offerta, le risorse necessarie a sviluppare ed implementare un modello e le relazioni con altri attori del mercato. Tutti questi elementi sono interconnessi con il modello di reddito includendo le fonti, le quotazioni di prezzo e le strutture di costo.
I primi tre elementi sono fattori chiave su cui le aziende devono focalizzarsi dopo che si è deciso di partecipare al business Open Source.

L'offerta: il modello di business Open Source è coerente con le definizioni e le caratteristiche di altri modelli di "business e management" già teorizzati e disponibili in letteratura.[42]
I concetti di prodotto e di offerta sono sempre presenti all'interno del modello di business ed influenzano direttamente il fattore reddito. Generalmente, il mercato target, l'orientamento tra prodotti e servizi, il modello di licensing sono aspetti legati alla strategia di prodotto. Diversamente l'offerta include aspetti quali ad esempio l'efficacia del prodotto ovvero il reale beneficio che il cliente sta ottenendo, le caratteristiche del prodotto, lo stile, la qualità, il marchio ed il confezionamento del prodotto in vendita.
Dal punto di vista del modello di business, una caratteristica che distingue il prodotto Open Source (come in genere il software) dagli altri è la sua non fisicità ovvero la caratteristica informativa del prodotto. Le informazioni sottese al prodotto hanno caratteristiche univoche che divergono largamente dalle caratteristiche fisiche.

Oltre che la tipologia di prodotto, anche la tipologia di licenza è inclusa nella definizione di offerta essendo un fattore determinante nella scelta del modello di reddito.

[41] *Rajala, Rossi,&Tuunainen, 2003; Osterwalder, 2004; Morris, Schindeutte, & Allen, 2005)*

[42] *Nel seguito ripiloghiamo brevemente per comodità i tre concetti, Offerta, Risorse e Relazioni, che sono alla base di un modello di business software, anche non Open Source.*

Le risorse: lo sviluppo delle risorse secondo una prospettiva industriale è legata alle strategie di business e di prodotto. Le risorse ed il know-how di una azienda sono fra gli elementi centrali nell'analisi e comprensione del proprio business. Questo accentua l'importanza delle risorse come elemento cruciale nello sviluppo delle competenze core dell'azienda.

Nell'analisi delle risorse nel business software, si tende a definire gli asset tangibili e intangibili come risorse fisiche e non fisiche, mentre le capacità, gli skill ed il know-how intangibili risorse di conoscenza.

Le relazioni: l'interazione dell'azienda con altri attori è considerata una parte essenziale del modello di business al pari dell'offerta e delle risorse. In particolare nel contesto dei modelli di business si sposta il focus dalla creazione di valore attraverso le risorse interne alla creazione di valore attraverso le relazioni esterne. Tali relazioni creano una rete di valori che sono un importante elemento nello sviluppo e distribuzione dell'offerta. Inoltre, essendo la rete di relazioni un importante asset intangibile, si favorisce una interconnessione con risorse di altre aziende.

L'analisi del **modello di sviluppo del reddito** nel contesto Open Source ha incontrato difficoltà e problematiche sin da quando il movimento Open Source enfatizzava la distribuzione libera della proprietà intellettuale. Comunque, se una azienda è coinvolta nel business Open Source può scegliere[43] di licenziare il proprio software come software Open Source o trarre beneficio da prodotti Open Source già esistenti. In entrambi i casi le modalità per trarre profitto si attuano attraverso la vendita di servizi connessi all'uso del software, la vendita di hardware connesso al software o la vendita di prodotti software licenziati da utilizzare insieme al software Open Source.
Sono stati identificati otto possibili modelli di reddito[44], riportati nella tabella sottostante, applicabili all'Open Source. Tale classificazione resta una delle più comprensive classificazioni ancora in essere.

[43] *De Laat, P.B. (2005) Copyright or Copyleft? An analysis of property regimes for software development. Reasearch Policy, 34(10), 1511-1532*

[44] *F. Hecker (1999): "Setting up shop: the business of Open Source software".*

Tabella 2: sommario dei modelli di reddito dell'Open Source [45]

Modello di reddito	Descrizione	Tipo licenza	Fonti di reddito
Support selling	Per aziende di profitto che effettuano supporto ad un software distribuito senza canone	Tutte	Il reddito proviene dai mezzi di distribuzione, dal marchio, servizi di training, consulenza, sviluppo personalizzato
Loss-leader	Un prodotto Open Source senza canone è usato come prodotto sottocosto nei confronti di un proprio software commerciale (per stimolare la domanda verso una specifica offerta)	Varie	Offerte complementari
Widget frosting	Aziende che primariamente vendono hardware usano tale modello. Ad esempio condividendo il codice dei driver al fine di un debugging evoluto	Tutte	Il maggiore reddito proviene dall'hardware
Accessorizing	Per aziende che distribuiscono manuali, PC e altri articoli quale supporto a software Open Source	Tutte	Offerte supplementari
Service enabler	Il software Open Source è creato e distribuito primariamente per accedere a fonti di reddito da servizi di consulenza e on-line	Tutte	Servizi a canone
Brand licensing	Una azienda richiede canoni da altre aziende per i diritti d'uso del proprio marchio e nome di prodotto nella creazione di prodotti derivati	Forte reciprocità	Compensazione del copyright
Sell it, Free it	I prodotti software di una azienda iniziano il loro ciclo di vita come prodotti commerciali e dopo sono convertiti in Open Source	Alterazione di licenza d'uso	Il reddito iniziale viene da offerte di prodotto convertite poi in altri modelli
Sw franchising	Insieme di diversi modelli (brand licensing e support seller) in cui una azienda autorizza altri ad usare il proprio marchio e nome di prodotto per creare organizzazioni associate che sviluppano software per clienti (per aree geografiche o mercati verticali)	Forte reciprocità	L'utilizzatore riceve training e servizi in cambio di canoni

A meri fini esemplificativi dell'interconnessione tra gli elementi principali del modello di business (offerta, risorse e relazioni) e il modello di reddito si riportano due "case study"[46] relativi ad aziende che operano nel mercato Open Source, **MySQL** e **RedHat,** di cui vengono messi a confronto i modelli di Business.

MySql è il marchio di un prodotto software Open Source proprio dell'azienda svedese MySQL AB. La compagnia sviluppa e mantiene un offerta basata sul prodotto chiave Open Source MySql (Database relazionale) sotto GPL license sponsorizzando una community.

RedHat è una delle maggiori aziende distributrici di Linux. L'offerta è nel riassemblare il software di sistema, nel supporto allo sviluppo ed in prodotti aggiuntivi. Aspetto univoco è che Red Hat non ha mai sviluppato il software offerto né pagato lo sviluppo a terzi.

[45] *Source: Modified from Hecker,1999; Valimaki, 2005*

[46] *Handbook of research on Open Source Software (2007), Kirk St. Amant & Brian Still pp. 547-551*

Le regola di RedHat è nel suo valore di rete legato alla maggiore attività di pacchettizzazione del software.

Nella tabella successiva si confrontano gli elementi del modello di business e il modello di revenue per le due aziende.

Tabella 3: comparazione tra elementi del modello di business per MySql e RedHat

Modello di business	MySql	RedHat
Offerta	Il cuore dell'offerta accorpa lo sviluppo in-house del database con i servizi.	Manutenzione del software di sistema e altri servizi
Risorse	Risorse interne per lo sviluppo e la gestione	Risorse impegnate sulla gestione del marchio (gestione del marketing e del business)
Relazioni	Bilanciamento tra Open Source community e rete commerciale per necessità sporadiche; la user community dipende dalla software community	La community è lo sviluppatore del software
Modello di reddito	La principale fonte di reddito deriva da licenze proprietarie e in piccola parte da servizi. Il modello di reddito include la vendita di supporto, feature e licenze duali quindi assimilabile al modello Loss-leader	La principale fonte di reddito è la "sottoscrizione" che consente alle aziende di sviluppare e esercire la propria tecnologia e fornisce supporto ai clienti. Il modello di reddito applicato è il Support-selling

6.3 La Governance di un progetto Open Source

La velocità di cambiamento del business è sempre maggiore ed è causa di un'accelerazione del ciclo di vita dei sistemi informatici i quali, spesso, non riescono ad assicurare la flessibilità e la dinamicità necessarie. L'implementazione di un'architettura di governance IT aiuta le grandi aziende a rispondere efficacemente alle richieste del business.

Uno strumento usato per determinare il percorso della governance è il Modello di Maturità. Attraverso tale modello è possibile valutare la maturità di alcuni processi chiave aziendali. In particolare il **Capability Maturity Model** (CMM)[47] è un metodo che mira ad una valutazione dei produttori di software mediante livelli successivi; tali livelli descrivono il percorso che un software provider deve intraprendere per arrivare ad un sempre maggiore grado di maturità. Il livello è una collezione di attività

[47] *Nel 1986 l'US Air force richiese al Software Engineering Institute di creare un modello sistematico per la valutazione dei contratti software; insieme a MITRE Corporation un gruppo di studio sviluppò un questionario che permise al committente di valutare un fornitore in base alle sue capabilities.*

chiave. La maturità è intesa nell'accezione di potenzialità e capacità di crescere e indica la ricchezza dei processi di una software company e la consistenza con cui vengono applicati. Produttività e qualità risultanti possono essere migliorati nel tempo attraverso considerevoli ritorni nella disciplina acquisita usando i propri processi. Tale modello è stato adattato da diverse discipline tra cui il knowledge management, project management e sviluppo di prodotti.

Partendo dal CMM, è stato sviluppato[48] per l' Open Source uno specifico "Maturity Model" definito OSMM. Attraverso tale modello si analizza il livello di maturità dei prodotti Open Source. Ogni prodotto viene valutato sulla base di sei elementi ovvero le caratteristiche, il supporto, la documentazione, il training, le integrazioni ed i servizi. Sono state apportate successivamente ulteriori variazioni al modello[49] introducendo elementi ulteriori quali l'età del prodotto, le piattaforme supportate, la popolarità, la qualità del disegno ed i costi. Non sono invece gestite le informazioni sul livello di maturità dell'organizzazione, delle architetture e del supporto al cliente dopo che il prodotto è stato installato.
Nell'applicazione del modello gioca un ruolo chiave il repository. Il repository è un database applicativo contenente informazioni strutturate e non, richieste ai più alti livelli di maturità; è fondamentalmente un'applicazione database che contiene informazioni sull'asset, con profonde interazioni a fonti documentali non strutturate. Il repository può contenere sia i meta-dati che gli oggetti stessi.

Implementare un repository Open Source come base informativa del proprio asset, consente di espandere in maniera ottimale ed efficace il proprio modello di business, superando l'approccio iniziale spesso caotico per orientarsi verso un "service point" unico e centralizzato. In questo senso l'applicazione del OSMM prevede una fase iniziale di raccolta dello stato delle componenti Open Source presenti in azienda per poi aggiungere altri bacini informativi come il tracking delle versioni, analisi di impatto specifiche, servizi di abbonamento, contesti informativi ulteriori e metriche di riutilizzo.

[48] *Golden, B.(2005). Suceeding with Open Source. Reading, MA: Addison-Wesley*

[49] *Giuliani, G.,Woods,D. (2005). Open source for the enterprise: Managing risks, reaping rewards. Cambridge, MA: O'Reilly Media.*

Il modello di governance[50]

I componenti Open Source sono normalmente descritti e pacchettizzati in ambienti dedicati alle community di sviluppo. Tali community Open Source, come SourceForge, rendono disponibili meta-dati come ad esempio l'amministratore, gli sviluppatori, le istruzioni di installazione e lo stato di sviluppo. Come per gran parte dei repository definiti all'esterno, le organizzazioni non sono però abilitate ad aggiungere applicazioni o funzionalità a meno di replicare i dati. Quindi l'integrazione e l'aggiornamento spesso viene a mancare e molte fonti di software Open Source si dimostrano non affidabili.

Ovviamente, il repository non risolve da solo tutti i problemi che ruotano intorno alla governance IT; però consente all'azienda di indirizzare le diverse criticità secondo un modello di maturità e di aggiungere ulteriori funzionalità in linea con quanto richiesto dai processi chiave individuati. In particolare, una politica efficace di governo dell'ambiente Open Source dovrebbe preoccuparsi dei seguenti aspetti:

- Gestione dell'inventario

Il primo e più importante tema da prendere in esame è l'analisi dell'ambiente e delle diverse tecnologie che compongono il portafoglio applicativo. Le applicazioni Open Source sono di norma una piccola parte dell'infrastruttura e ogni componente deve essere soggetta a regole e processi di gestione dell'inventario con informazioni relative a chi utilizza le applicazioni, dove sono installate e quali sono i prodotti Open Source. Non è pensabile, anche al più basso livello operativo, gestire o governare l'architettura senza conoscere quali elementi compongono l'ambiente operativo.

- Analisi d'impatto

Mentre la gestione dell'inventario si occupa di quali prodotti sono installati, l'analisi d'impatto si focalizza sulle relazioni tra applicazioni Open Source e altri asset esistenti.

[50] *"Con IT governance, o governo dei sistemi informativi, si intende quella parte della più ampia corporate governance che si occupa della gestione dei sistemi IT (Information Technology: sistemi informativi) in azienda; il punto di vista della IT governance è rivolto alla gestione dei rischi informatici ed all'allineamento dei sistemi alle finalità di business. La corporate governance si è molto sviluppata in seguito ai recenti sviluppi normativi in USA (Sarbanes-Oxley) ed Europa (Basilea II) che hanno avuto notevoli ripercussioni anche sulla gestione dei sistemi informativi. L'attività di analisi attraverso cui si perseguono questi obiettivi è l'IT Audit" tratto da http://it.wikipedia.org.*

L'analisi d'impatto risponde a domande relative ai sistemi che usano componenti Open Source, alla presenza di competenze sulla tematica, alle interazioni con altri prodotti, agli impatti nel caso di sostituzione di un'applicazione e ai team di supporto. Tali domande diventano più critiche quando l'ambiente tecnologico è visto come vantaggio competitivo per il business; oggi infatti non è più ammessa l'indisponibilità di ambienti per cause ad es. di integrazione o di sicurezza, e solo conoscendo l'impatto di un cambiamento è possibile evitare interruzioni del servizio.

- Riuso e analisi del dominio

L'analisi di dominio è il processo di rivisitazione degli ambienti di business ed IT che mira ad identificare elementi variabili comuni. Ciò costituisce il requisito più importante nello sviluppo o nell'utilizzo di componenti già presenti. Attraverso il controllo della mappatura tra funzionalità di business e lo specifico asset tecnologico, l'architettura di governance assicura che solo una soluzione venga implementata per uno specifico problema. Ad esempio la funzionalità generalizzata di web server è associata all'asset Apache: quindi per la richiesta di funzioni base si utilizza la stessa applicazione. Il riuso diventa una tecnologia critica e un requisito per qualunque progetto di sviluppo. Il riuso della conoscenza, dei processi e del software incrementano la produttività e la qualità delle organizzazioni IT.

- Acquisizioni e controllo delle versioni

L'adozione e l'effettivo utilizzo di software Open Source in un'organizzazione costituisce una criticità in termini di governance. Il software Open Source, per sua natura libero e disponibile, implica che non c'è un'unica origine del codice relativo ad una certa applicazione. Ad esempio Linux può essere scaricato da diversi siti ognuno con una diversità di versione, di supporto e documentazione. Idealmente il repository Open Source dovrebbe avere una sola versione che coincide con il modello di dominio e per qualsiasi implementazione si dovrebbe utilizzare il medesimo codice. Tramite meccanismi di tracciatura sugli accessi alle banche dati per il download del codice, l'architettura di governance garantisce sicurezza e gestibilità dell'ambiente operativo.

- Metriche e misure

Al fine di salire nei livelli di maturità dell'OSMM, si ha la necessità di integrare metriche nel processo di governance. Le metriche di base riguardano le installazioni, le versioni di downloads, la documentazione. Man mano che l'organizzazione matura nel processo di riutilizzo del software, possono emergere altre metriche come ad esempio il volume di transazioni relative alle applicazioni Open Source, ciò anche al fine di determinare l'evoluzione delle piattaforme. Altre metriche sono il ROI, le metriche di capacità e di performance. Su base periodica occorrerebbe poi collezionare le misure ed analizzarle tramite strumenti di trend analysis.

Il repository Open Source

L'idea diffusa di repository Open Source è quella di una applicazione che consente di raccogliere un limitato set di elementi significativi per l'organizzazione. In realtà il repository è parte di un framework molto più ampio costituito da prodotti, servizi, strumenti e processi.

Il repository Open Source può essere distinto in cinque componenti base:

- Portale Open Source

È lo strumento che fornisce agli utenti l'accesso a diverse tipologie di informazioni ed applicazioni tramite una interfaccia standard. Rappresenta un singolo punto di accesso per tutti i prodotti e servizi Open Source. La modalità di presentazione dei risultati delle richieste avviene attraverso un collezionamento di asset, ovvero un metodo di raggruppamento di asset basati su un contesto di ricerca. La pagina risultante presenta i meta-dati che descrivono il componente stesso. La presentazione degli oggetti in modo usabile e funzionale è sinonimo di successo del repository.

- Data loader

È uno strumento che consente il caricamento dei meta-dati nel meta-modello da diverse fonti (database e asset). Oltre alle utilità di caricamento automatico molte applicazioni dispongono di librerie per il versioning, qualità dei dati, integrazione e inserimento dati.

- Processi di business

Attraverso i processi di business il repository può divenire da magazzino passivo di informazioni a soluzione integrata di governo dell'ambiente. Ogni componente Open Source deve essere collezionata e catalogata nel repository attraverso procedure di caricamento automatico (es. web services). Nel repository dovrebbe essere gestito lo stato del componente, che risulterà pending quando deve essere ancora rivisto e valutato dalla struttura di governance (la valutazione e il punteggio sono espressione del supporto, maturità, funzionalità e rischio); una volta che il componente è approvato allora la funzionalità viene resa disponibile all'organizzazione.

- Processi applicativi

S'intendono prodotti e servizi che operano sulle informazioni contenute nel meta-dato. Mentre i processi di business sono focalizzati sul flusso informativo nell'ambiente, i processi applicativi si focalizzano sul valore aggiunto dei dati del repository. Un tipo di processo applicativo ad esempio è la misura dell'ammontare dei contenuti del repository Open Source che include componenti, meta-dati e documentazione. Altro tipo è l'uso delle informazioni Open Source che è critico per la formulazione di programmi a lungo termine. Altri esempi sono la sottoscrizione di servizi e le ricerche fallite.

- Ambiente di supporto al cliente

I componenti Open Source necessitano di un ambiente per interfacciare il cliente.
In genere il gruppo di supporto crea un ambiente di interfaccia tra la community e l'organizzazione. Tale ambiente nel repository è una complessa collezione di comunicazioni di diversa natura: mono direzionale, collaborativa e interattiva. L'aggiunta di utility del prodotto di supporto al cliente permette di creare nel tempo esperienza per il cliente stesso. Alcune componenti base sono: guida utente, help-online, FAQ e programmi di training.

Il repository Open Source appena descritto, nella sua accezione più ampia, sta entrando all'interno delle community. L'evoluzione di alcune tecnologie quali la SOA ed i Web services[51] sta portando sul tavolo di architetti e sviluppatori la necessità di una gestione strutturata di dati e processi in linea con il framework proposto. Sulla stessa linea si sta muovendo l'ITIL[52], che identifica, all'interno delle sue best practices per la gestione dei servizi IT, il repository come risorsa centrale per la governance di un'azienda. Quindi il repository sta diventando il centro informativo di controllo per l'azienda. Ovviamente, nel caso di piccole organizzazioni che utilizzano pochi componenti Open Source, il repository non è strettamente necessario; al contrario, le grandi organizzazioni tendono a sfruttarne in modo progressivo le sue funzionalità quale elemento di vantaggio competitivo.

La convergenza a livello enterprise delle architetture e della governance aumenterà la disponibilità di strumenti, standard e team di supporto; elementi questi che contribuiranno significativamente allo sviluppo di repository Open Source. Il sempre maggiore sviluppo di software in outsourcing, l'integrazione di componenti Open Source e l'implementazione di architetture "di servizio" renderà quindi il repository indispensabile all'interno delle aziende medio-grandi.

[51] Service-Oriented Architecture (SOA) viene indicata un'architettura software atta a supportare l'uso di servizi Web per soddisfare le richieste degli utenti così da consentire l'utilizzo delle singole applicazioni come componenti del processo di business. Un Web Service (servizio web) è un sistema software progettato per supportare l'interoperabilità tra diversi elaboratori su di una medesima rete; caratteristica fondamentale di un Web Service è quella di offrire un'interfaccia software utilizzando la quale altri sistemi possono interagire con il Web Service stesso attivando le operazioni descritte nell'interfaccia tramite appositi "messaggi" XML inclusi in una "busta" SOAP (wiki).

[52] Information Technology Infrastructure Library (ITIL) è un insieme di linee guida ispirate dalla pratica (Best Practices) nella gestione dei servizi IT (IT Service Management) e consiste in una serie di pubblicazioni che forniscono indicazioni sull'erogazione di servizi IT di qualità e sui processi e mezzi necessari a supportarli (wiki).

7. Utilizzo ed adozione Open Source nel Settore Pubblico

7.1 Il ruolo pubblico nel supporto del software Open Source

L'interesse del settore pubblico nell'Open Source è in continua crescita; in Europa i primi segnali d'interesse si sono manifestati con la conferenza di Lisbona[53] nel cui piano strategico 2000-2005 veniva chiaramente affermato che l'Open Source era lo strumento per consentire la partecipazione attiva dei cittadini allo sviluppo di una società globale basata sull'informazione[54]. Nel mercato del software il ruolo giocato dal settore pubblico è considerato di primaria importanza sia come regolamentatore, tramite autorità, norme e leggi, sia come primario cliente[55].

L'adozione di software Open Source nel settore pubblico, tralasciando motivazioni puramente ideologiche, è ricercata nella propria intrinseca natura di superiorità rispetto al software proprietario in termini di immediata disponibilità, sicurezza, flessibilità e manutenibilità del codice; tali caratteristiche derivano in prima istanza dalle modalità organizzative delle community e in seconda istanza dall'efficienza dei costi. Quest'ultimo è un elemento strategico in contesti in cui c'è pressione nei budget, esplicitati dal ridotto o totale abbattimento dei costi di licenza e dall'opportunità di instaurare liberamente rapporti contrattuali con terze parti per la manutenzione di software senza subire lock-in di fornitori esterni, o infine di usufruire delle economie di scala legate al riuso del software.

Dal punto di vista del Paese Italia, l'interesse del settore pubblico nell'Open Source è considerato un modo per supportare lo sviluppo dell'industria locale, limitando le dipendenze dai grandi fornitori di mercato e stimolando nel contempo la competitività del mercato.

[53] *Il Consiglio europeo ha tenuto una sessione straordinaria il 23 e 24 marzo 2000 a Lisbona per concordare un nuovo obiettivo strategico per l'Unione al fine di sostenere l'occupazione, le riforme economiche e la coesione sociale nel contesto di un'economia basata sulla conoscenza.*

[54] *eEurope, COM(1999) 687 def*

[55] *Bertini, L. (et al.), Appalti pubblici e innovazione: il procurement come strumento di supporto allo sviluppo della "Società della Conoscenza", Quaderni Consip-MEF, 2007*

A tali argomentazioni, nella diffusa e copiosa letteratura sull'argomento, vi sono anche correnti di pensiero che contraddicono quanto esposto; non si ritiene infatti determinante il ruolo del settore pubblico nello sviluppo dell' Open Source, adducendo al fatto che questo ha avuto successo indipendentemente da tale ruolo e che il mercato del software proprietario sembra essere un esempio di mercato ben funzionante; anzi l'intervento di stimolo del settore pubblico può rappresentare una minaccia verso le grandi aziende nell'investire nella qualità del loro software.

Di contro al tema dell'efficienza dei costi, diversi autori indicano che il risparmio ottenibile dall'adozione di software Open Source è molto minore rispetto a quello legato al software proprietario. I canoni di licenze rappresentano una piccola parte dei costi del software e la comparazione corretta del costi va effettuata sulla base del TCO che include altre voci quali training, supporto tecnico e manutenzione.

7.2 La situazione europea

A livello Europeo gli interventi dei governi e delle agenzie, a partire dal 2003, sono registrati e accessibili nel sito Open Source Observatory[56] a cura della Commissione Europea nell'ambito del programma IDABC.
L'analisi che segue[57] deriva dal set di informazioni registrate, che ovviamente non rappresentano l'intero parco di tutte le iniziative/interventi prese a livello comunitario dai paesi membri. Il campione considerato comprende oltre 100 interventi distribuiti su 14 Paesi. L'analisi ha raggruppato i dati nelle seguenti tipologie di policy:

- **tipo di software coinvolto dall'intervento** (custom, pacchetto, uso di Open Standard);
- **livello politico/amministrativo a cui viene deciso l'intervento** (PA/Agenzie[58] Centrali e PA/Agenzie Locali);

[56] *http://ec.europa.eu/idabc/en/chapter/452*

[57] *S. Comino, F. Manenti , A. Rossi (2007) . Handbook of Research on Open Source Software. (pp. 412- 427) . Information Science Reference*

[58] *Per Agenzie (Locali e/o Centrali) si intendono Enti che erogano servizi pubblici su base territoriale (es servizi postali)*

- **tipo di intervento** (*adozione* nel casi di decisione di utilizzo di software Open Source, *consulenziale* nel caso di spinta all'uso di software Open Source e di *sviluppo* nel caso di sviluppo di software Open Source);
- **razionali dell'intervento** suddivisi in sette categorie:
  – *efficienza dei costi* (motivazioni legate al risparmio di canoni di licenza, economie di scala dal riuso, risparmi dallo sviluppo cooperativo, impiego di risorse ottimizzato);
  – *disponibilità del codice* (motivazioni connesse alla qualità del software, robustezza e sicurezza);
  – *interoperabilità* (motivazione connessa allo stimolo della diffusione e promozione dell' Open Source nel mercato del software);
  – *flessibilità* (motivazione legata al vantaggio connesso alla possibilità di personalizzare il codice alle specifiche necessità, all'integrazione e compatibilità con i sistemi);
  – *vantaggio competitivo* (motivazione legata alla possibilità di creare alternative alle industrie proprietarie supportando l'industria locale e stimolando l'indipendenza tecnica dai fornitori);
  – *efficienza nel settore pubblico* (motivazione legata alla diffusione di best practice nella PA);
  – *diffusione dell'informazione* (motivazione legata allo sforzo di aumentare le informazioni disponibili e sviluppare la consapevolezza della PA sull' Open Source).

Nella tabella sottostante si riporta la distribuzione degli interventi suddivisi per livello amministrativo e tipologia.

Tabella 4: distribuzione degli interventi suddivisi per livello amministrativo e tipologia di intervento

	Interventi			
Livello	**Sviluppo**	**Adozione**	**Consulenziale**	**Totale**
PA Centrale	8 (17,8%)	9 (20%)	28 (62,2%)	45 (100%)
Agenzie Centrali	1 (8,3%)	9 (75%)	2 (16,7%)	12 (100%)
PA Locale	9 (21,4%)	24 (57,1%)	9 (21,4%)	42 (100%)
Agenzie Locali	1 (16,7%)	5 (83,3%)	0 (0%)	6 (100%)
Totale	19 (18,1%)	47 (44,8%)	39 (37,1%)	105 (100%)

Oltre l'80% degli interventi sono presi a livello PA Centrale e PA Locale, mentre le Agenzie hanno un minore peso. Le policy "Consulenziale" e "Adozione" sono quelle più diffuse indipendentemente dai livelli amministrativi. Emerge che i governi a livello centrale si pongono come linee guida, mentre decisioni operative sono proprie dei governi a livello locale.
Nella tabella sottostante gli interventi sono raggruppati in base alla tipologia di software; spesso uno stesso intervento può interessare più tipologie di software per cui le percentuali superano il 100%.

Tabella 5: distribuzione degli interventi suddivisi per livello amministrativo e tipologia di software

	Software		
Livello	Custom	Pacchetti	Open Std
PA Centrale	69%	73%	0%
Agenzie Centrali	33%	66%	17%
PA Locale	38%	78%	5%
Agenzie Locali	83%	33%	0%
Totale	53%	72%	8%

Ciò che emerge è che i livelli di governo locale sono molto attivi verso i pacchetti mentre quello centrale non sembra seguire particolari modelli.
Restringendo l'attenzione sul livello amministrativo centrale, nella tabella sottostante vengono analizzati i razionali degli interventi in matrice con gli interventi stessi. Anche in tal caso uno stesso intervento può avere più razionali per cui le percentuali superano il 100%.

Tabella 6: matrice tra il tipo di interventi e le motivazioni

	Razionale intervento							
Intervento	**Totale**	**Effic.za costi**	**Dispon. codice**	**Interoper.**	**Flessib.**	**Vantaggio Compet.**	**Effic.za PA**	**Diffusione inform.**
Sviluppo	6 (100%)	1 (17%)	4 (67%)	1 (17%)	1 (17%)	1 (17%)	0 (0%)	1 (17%)
Adozione	11 (100%)	7 (64%)	2 (18%)	6 (55%)	3 (27%)	2 (18%)	1 (9%)	1 (9%)
Consulenziale	20 (100%)	11 (55%)	7 (35%)	11 (55%)	2 (10%)	8 (40%)	5 (25%)	7 (35%)
Totale	37 (100%)	19 (51%)	13 (35%)	18 (49%)	6 (16%)	11 (30%)	6 (16%)	9 (24%)

Dalla distribuzione emerge che la motivazione più diffusa è quella legata alla efficienza dei costi, seguita dall'interoperabilità e disponibilità del codice. In relazione agli interventi, nel caso specifico di adozione del software, le motivazioni più rilevanti sono anche legate all'interoperabilità , mentre sono meno significativi il vantaggio competitivo, la disponibilità del codice e la flessibilità. Per gli interventi consulenziali le motivazioni interoperabilità ed efficienza costi sono le più diffuse.
In un'ottica più generale emerge, in Europa, una situazione in cui gli interventi pubblici tendono a stimolare piattaforme ICT e sviluppi applicativi aperti e standard al fine di favorire lo sviluppo di un mercato maggiormente competitivo.

7.3 Le esperienze Consip/Mef

L'adozione dell' Open Source ha seguito nel Sistema Informativo del Ministero dell'Economia delle Finanze un approccio graduale, che lo ha preferito prima per scelte tattiche e poi, al maturare della disponibilità di strumenti e competenze, anche per componenti strategiche.
Il primo ambito di adozione è stato quello degli Standard Open Source per lo sviluppo delle applicazioni. Infatti se è vero che il fenomeno dell'Open Source è esploso grazie alla disponibilità di codice di varia natura e qualità è anche vero che un certo numero di comunità si sono orientate alla redazione e sviluppo di standard riguardanti tutti gli ambiti dell'ecosistema ICT dai protocolli di trasporto TCP/IP ai linguaggi di programmazione Java alle specifiche architetturali della Java 2 Enterprise Edition (J2EE).

Le prime scelte di Consip in tal senso risalgono al 2000 quando fu necessario indirizzare gli sviluppi di nuove applicazioni in scenari non legacy.
Si preferì adottare Java come linguaggio di programmazione e l'architettura J2EE come ambiente di esecuzione, in un momento in cui la predominanza Microsoft sembrava ineluttabile. Fu una scelta inizialmente tattica dettata dalla volontà di valutare le capacità del nuovo linguaggio ed architettura e di verificare come il mercato avrebbe reagito alla disponibilità di questi standard.

Oggi con più di cinquanta applicazioni del Sistema Informativo del MEF sviluppate ed eseguite in ambiente Java, e tra queste alcune con decine di migliaia di utenti, si può affermare che la scelta fu corretta. Il mercato dello sviluppo software e dei prodotti ha adottato pienamente questi standard aperti. Non solo esistono prodotti Open Source pienamente affidabili ma tutti i maggiori produttori, oltre a supportare pienamente questi standard, sono presenti in queste comunità: secondo il detto "se non puoi batterli unisciti a loro". Questa adesione dei produttori ha comunque rallentato la dinamicità di queste comunità.

L'approccio adottato da Consip si è andato consolidando ed arricchendo, man mano che strumenti Open per lo sviluppo delle applicazioni maturavano, con nuove indicazioni per l'utilizzo sia di moduli specializzati (per es. scrittura dei log o per stampe) sia dei framework[59] di programmazione. Questo ha permesso, pur operando in un ambiente di sviluppo multifornitore, di avere software applicativo omogeneo.
Il secondo ambito, più tecnologico, è stato quello dei sistemi operativi Open Source. Anche qui l'approccio è stato prima tattico, utilizzando Linux per macchine dedicate ad attività di infrastruttura e quindi con livelli di servizio non critici.
Oggi Linux, nelle due distribuzioni Red Hat e Suse, viene utilizzato negli ambienti di produzione. Le condizioni a che ciò si verifichi è la disponibilità della sua certificazione rispetto al middleware che ospita e che ne sia garantito il servizio di supporto. Tale supporto, spesso, come nel caso di Oracle, è garantito dal fornitore del middleware stesso.
Con il progressivo maturare dei prodotti Open Source, Consip si è trovata sempre più frequentemente a dover scegliere tra prodotti di mercato e prodotti Open Source o anche solo tra questi. Ha quindi ritenuto, nel corso del 2005, necessario dotarsi di un metodo formale di "valutazione" della maturità di un prodotto scegliendo il "Open Source Maturity Model"[60].

La metodologia è stata applicata ad esempio per la scelta del prodotto per la gestione della configurazione del software applicativo.

[59] *Framework, definiti anche pattern, sono strutture di supporto secondo le quali un software può essere organizzato e progettato, utilizzando modelli predefiniti per specifici scopi.*

[60] *Sviluppato da Bernard Golden, CEO di Navica www.navicasoft.com. Vedi anche Cap. 4.1*

Modello di adozione/supporto - Attitudine Consip verso l'Open Source

Il modello di supporto del Open Source adottato in Consip ne riconosce le opportunità e opera per mitigarne i rischi connessi.

L'approccio è articolato nella:

- valutazione e approvazione dell'utilizzo di specifici prodotti o classi di prodotti per determinati utilizzi;
- nell'accertamento, in fase di approvvigionamento, della disponibilità di documentazione e supporto;
- nella definizione del processo per la manutenzione ed aggiornamento del prodotto.

In termini pratici ciò si traduce in:

- il codice sorgente dei prodotti software Open Source non viene modificato;
- le necessità specifiche di personalizzazione, profilatura e sicurezza sono sviluppate all'interno dei nostri contratti di sviluppo e costituiscono codice proprietario dell'Amministrazione.

In tutte le soluzioni indicate, MEF-Consip utilizza il software Open Source nella stessa modalità con cui fruisce di un software proprietario, cioè riceve gli eventuali aggiornamenti attraverso le release del software mantenute dal Fornitore all'interno del quale sono cablati i moduli Open Source.

Con un'unica rilevante eccezione relativa al prodotto CVS, per la gestione della configurazione del software applicativo, per il quale, Consip, attraverso le risorse di assistenza del Fornitore, acquisisce in modo continuativo gli aggiornamenti resi disponibili nella Community.

Più recentemente, sotto l'egida CNIPA, Consip sta valutando l'ipotesi di rendere il software sviluppato internamente, liberamente disponibile sotto una licenza Open Source aderendo/creando una community. L'opportunità potrebbe essere la riduzione dei costi per mantenere un software aggiornato.

Rendendolo liberamente disponibile in una community aperta, le modifiche possono essere proposte e realizzate diffusamente e riducendo quindi i costi dello sviluppo. Questo scenario non escluderebbe anche la possibilità per un venditore di offrire servizi commerciali a supporto del prodotto.

Si tratta comunque di una strategia non senza rischi che vanno dalla definizione delle proprietà intellettuale, all'aiuto di potenziali competitori e, non ultima, la possibilità di costi addizionali senza alcuna garanzia di successo.

Lo stack Open Source Consip in dettaglio

Per tutto quanto detto ancora oggi la presenza di Open Source nel Sistema Informativo del MEF è più consistente nei layer di infrastruttura e negli ambienti di sviluppo/programmazione mentre la presenza è ancora limitata nei middleware (dbms, application server etc.) dove la carenza di funzionalità di gestione e monitoraggio costituisce un fattore penalizzante.

Consip impiega soluzioni basate in tutto o in parte su software Open Source nei seguenti settori:

Infrastruttura:

- Server (Sistemi Operativi Linux);
- Web Server (Apache);
- Application Server (Tomcat).

Strumenti per lo sviluppo di applicazioni:

- Linguaggio Java EE;
- Compilatori GCC;
- Framework per lo sviluppo (Struts).

Strumenti per lo sviluppo di moduli applicativi verticali:

- Prodotti Open Source;
- CVS per la gestione della configurazione delle applicazioni;
- Tool/strumenti di supporto.

Infrastruttura – Server

Linux è il sistema operativo libero basato sul kernel Linux e sul sistema GNU.
In ambito MEF Consip utilizza la distribuzione **Red Hat Enterprise Linux 4** con servizi di supporto nell'ambito del contratto Oracle.
I principali ambienti in esercizio ospitati su piattaforma Linux sono:

- sito Internet Consip (www.consip.it);
- sito Intranet RGS;
- nuovo Portale Igrue;
- sistema di SSO (ambienti di collaudo).

La distribuzione **Suse** viene utilizzata sul mainframe con il supporto offerto da IBM, come uno dei nodi della piattaforma di System & Network Management IBM Tivoli.

Infrastruttura – Web Server

Apache è il nome dato alla piattaforma Web Server più diffusa al mondo (il nome deriva dalla Community Apache Software Foundation source che gestisce il software). È in grado di operare sui sistemi operativi UNIX-Linux e Microsoft.

In ambito MEF, Apache è il Web Server più utilizzato, ad esempio:

- tutte le applicazioni gestionali Java (circa 50 per 30.000 utenti nominali);
- il sito Internet Consip;
- le Intranet DAG e RGS;
- tutti i servizi SPT.

Infrastruttura – Application Server

Java EE è la versione *enterprise* della piattaforma java. Essa è costituita da un insieme di specifiche che definiscono le caratteristiche e le interfacce di un insieme di tecnologie pensate per la realizzazione di applicazioni di tipo *enterprise* e *mission critical*. Chiunque può realizzare una implementazione di tali specifiche e produrre application server compatibili con le specifiche Java EE.
Nel giugno 2005 Sun Microsystem ha rilasciato il codice delle componenti della piattaforma Java EE che da quel momento sono state sviluppate dalla Community "GlassFish", con il contributo di Sun, Oracle, TmaxSoft ed altri.

La community ha rilasciato la prima versione definitiva della piattaforma Java EE 5 nel maggio 2006 ed sta lavorando alla seconda versione. Il codice è oggi disponibile sotto la Common Development and Distribution License (CDDL) e sarà presto disponibile come licenza GLP v2 completa di ClassPath Exception.
Consip ha scelto di utilizzare l'implementazione di mercato IBM WebSphere Application Server (WAS) ma sono presenti diverse istanze di Tomcat utilizzato come ambiente di esercizio di prodotti proprietari.

Strumenti per lo sviluppo di applicazioni e tool di supporto

Tutte le principali applicazioni gestionali Java (Sico, Entrate, Athena, GECO) sono sviluppate a partire da framework OpenSource.

Un **framework** è una struttura di supporto su cui un software può essere organizzato e progettato, allo scopo di risparmiare allo sviluppatore la ri-scrittura di codice già steso in precedenza per compiti simili.

I principali framework utilizzati sono:

- **Spring** (http://www.springframework.org/);
- **Struts** (http://struts.apache.org/);
- **Shale** (http://shale.apache.org/).

Molte applicazioni gestionali, oltre ad essere sviluppate sulla base di Framework Open Source, utilizzano moduli Open Source per gestire specifiche aree funzionali. Alcuni esempi importanti sono:

- **Maven 2.0** come strumento per la compilazione in formato aperto di applicazioni Java EE quali Area Igrue, Entrate, PointRGS, etc. (http://maven.apache.org/);
- La gestione delle funzionalità di logging tramite il prodotto **log4j** della Apache Foundation (http://logging.apache.org/log4j/docs/);
- La gestione delle stampe e della reportistica di lavoro del Sottosistema Igrue, che utilizza il prodotto **JasperReports** (http://jasperforge.org/sf/projects/jasperreports);

- La produzione di stampe **PDF** in molte applicazioni (SPT, Contabilità Economica, etc.) viene effettuata utilizzando librerie Open Source;
- **LAMPP** (Apache, Mysql, PHP, Perl) utilizzato in piccole applicazioni di supporto alle attività dell'area Entrate;

e centinaia di altre librerie open scaricate da www.ibiblio.org ed integrate nelle applicazioni Java EE

Prodotti Open Source

Consip ha sviluppato un sistema interno di gestione della configurazione software che utilizza tutti componenti OSS:

- sistema Operativo: Linux distribuzione Red Hat;
- linguaggio di programmazione: php;
- database: Mysql;
- prodotto: CVS, CVSmonitor e Viewcvs.

In modo del tutto analogo prodotti OSS sono utilizzati come strumenti di supporto per attività di Workgroup, archiviazione, pubblicazione e accesso alla documentazione di progetto e alla documentazione operativa tra gli eesempi più significativi:

- tools : owl, samba, proftpd, sshd, MTRG, filezilla, 7-zip, sql-tool, Mozilla Firefox, open-office;
- per il Blog dell'Amministratore Delegato: Dblog;
- per il Forum TFR: Snitz Forum 2000.

Il Configuration Management Applicativo (CMA)

Il CMA è un sistema realizzato nel 2006 allo scopo di dotare le aree applicative di uno strumento per la gestione della configurazione del software applicativo in ambienti dipartimentali.

La gestione della configurazione delle versioni software, che si avvicendano nel tempo o che provengono da attività parallele di sviluppo e correzione di uno stesso parco software, è fondamentale per garantire la qualità delle applicazioni informatiche utilizzate. Tale necessità è stata sempre più sentita negli anni, man mano che si è passati dai sistemi legacy (in ambiente mainframe) ai sistemi open (in ambiente distribuito).

Nel 2006, a fronte di una sperimentazione interna svolta in Consip, si è avviata la realizzazione di un intero sistema integrato di configuration management basato su software Open Source.

Tutte le componenti utilizzate sono Open Source:

- il sistema operativo è *Linux* (distribuzione Red Hat);
- il web server è *Apache;*
- i prodotti utilizzati sono *CVS*, *ViewCVS*, *CVSMonitor* e *Maven*.

Le esigenze di sicurezza e di profilatura degli utenti hanno richiesto la realizzazione di un "Sistema integrato di gestione della configurazione" che ha affiancato ai prodotti Open Source alcune componenti sviluppate internamente. Rimanendo fedeli alla filosofia dell'intero progetto però anche per queste integrazioni si è ricorso esclusivamente a prodotti Open Source: linguaggio di programmazione *PHP* e data base *Mysql.*

Una caratteristiche peculiare del sistema è stata la scelta effettuata di utilizzo del codice sorgente Open Source nella versione di volta in volta fornita dalla Community senza modifiche e della raccolta dal Web delle patch correttive messe a disposizione dalla molteplice comunità degli sviluppatori dei prodotti software Open Source, che vengono testate dal team di assistenza e certificate come compatibili con l'intero sistema, prima di essere avviate nell'ambiente di produzione.

Il codice sviluppato internamente invece, è codice proprietario dell'amministrazione e come tale supportato contrattualmente per le attività di manutenzione correttiva.

La realizzazione è terminata nell'ottobre del 2006 e da allora sono in corso le attività evolutive, dettate sia dalle richieste delle singole aree applicative, che di volta in volta vengono inserite nel sistema, sia dagli adeguamenti evolutivi delle infrastrutture conseguenti all'aumento del numero di utenti.

I nuovi contratti di sviluppo e gestione del sistema informativo del MEF, in corso di definizione, prevedono il CMA come strumento interno di gestione della configurazione applicativa al quale si dovranno adeguare gli aggiudicatari delle prossime gare.

Le maggiori complessità che incontra un progetto integrato basato sia sull'utilizzo di componenti Open Source che di sviluppo interno, sono legate alla definizione delle attività di assistenza e gestione successive alla messa in produzione del sistema.

Il sistema CMA è supportato da un centro di competenza specifico formato da sviluppatori del software e personale di assistenza in grado di operare su sistemi innovativi e la cui conoscenza non è sempre facilmente reperibile sul mercato IT (pensiamo al Sistema Operativo Linux, al data base Mysql, al linguaggio di programmazione PHP), inoltre è diversa la concezione stessa del servizio di assistenza : si basa su una continua attività di ricerca sul web per il prelievo di "news" dal mondo degli sviluppatori Open Source, sulla valutazione caso per caso dell'utilità di determinate informazioni e sul collaudo delle nuove versioni a garanzia del corretto funzionamento nel tempo del sistema.

Ciò, come è facile comprendere, comporta un costante impegno e attenzione sia da parte di Consip per il governo dei servizi di assistenza e manutenzione sia da parte del fornitore che è tenuto contrattualmente ad erogare quegli stessi servizi.

Le esperienze conseguenti alla fase progettuale del sistema e alla successiva fase gestionale ci consentono di elencare alcuni punti di riflessione sull'utilizzo nella pubblica amministrazione di software Open Source.

Tra i vantaggi si possono sicuramente evidenziare:

- i risparmi sulle licenze, rispetto ad una soluzione basata su prodotti proprietari;
- la non dipendenza dalle politiche commerciali degli specifici fornitori di prodotti di mercato;
- la maggiore qualità del software in quanto vi sono intere comunità formate da migliaia di utenti che giornalmente utilizzano e di conseguenza "collaudano" i prodotti;
- per lo stesso motivo evidenziato sopra, in genere si ha una più veloce risoluzione degli errori.

Di contro però va sottolineato come punto di attenzione il maggior costo gestionale (stimato per il CMA in circa il 20%) rispetto ad altre soluzioni, a causa della necessità di utilizzare risorse con competenze specifiche.
Infine un discorso più ampio sul tema , in particolare per la pubblica amministrazione, deve tener conto dell'esistenza di aspetti legali ancora non completamente normati nel sistema italiano e in genere in Europa sull' Open Source in grado di creare confusione e contenziosi.

LOGAN

Il Sistema Informativo del MEF comprende una molteplicità di infrastrutture, centralizzate (mainframe) e distribuite, fortemente integrate, su cui cooperano attraverso lo scambio dati, un elevato numero di applicazioni on-line e batch.

Il controllo della qualità delle elaborazioni batch è evoluto da strumento verticale utilizzato esclusivamente in ambiente mainframe a strumento che centralizza e coordina le operazioni su più macchine.

Prodotti differenti, differenti formati delle diagnostiche, elevata numerosità dei file di log e molteplicità di macchine ospitanti i vari passi delle procedure batch hanno reso difficoltosa la ricerca di eventuali anomalie o l'evidenziazione di un eventuale degrado nelle prestazioni delle procedure.

Si è dunque dato l'avvio ad un progetto sperimentale, denominato "LOGAN" (l'acronimo di LOG ANalyzer) integralmente basato su piattaforma Open Source, dedicato ad effettuare il monitoraggio complessivo ed integrato di tutte le operazioni in differita dei sistemi.

LOGAN è un sistema dotato di interfaccia web realizzata in linguaggio PHP versione 5.2.0 su Web Server Apache 2.0 ed integrata con database MySQL versione 5.0.27. Il sistema è inoltre residente su piattaforma Linux 2.6.10.

La scelta della piattaforma Open Source è stata dettata principalmente dalle seguenti esigenze:

- larga disponibilità di librerie e componenti facilmente integrabili per la realizzazione dei moduli di controllo;
- elevata efficienza;
- semplicità di realizzazione di procedure batch;
- elevata affidabilità e stabilità delle versioni;
- indipendenza dai vendor;
- contenimento dei costi.

LOGAN consente di:

- controllare e monitorare lo stato di esecuzione di procedure batch residenti su sistemi distribuiti realizzando un vero e proprio cruscotto centralizzato per il monitoraggio del batch;
- individuare immediatamente gli errori di esecuzione semplificando l'analisi delle anomalie;
- evidenziare incongruenze e/o ritardi nella schedulazione delle operazioni;
- rappresentare in maniera omogenea le risultanze di batch ETL, batch MVS, batch Oracle e script indipendentemente dagli ambienti di esecuzione;
- evidenziare le prestazioni generali delle catene di batch.

Il sistema è dotato di una interfaccia web grafica che rappresenta con immediatezza:

- la composizione della catena di operazioni con la segnalazione delle tecnologie di esecuzione impiegate;
- lo stato di completamento di ciascun passo del batch mediante l'uso di opportuni colori;
- il contenuto dei file di log con l'evidenziazione delle linee contenenti le segnalazioni di errore;
- la durata di ogni passo della procedura;
- i vincoli all'esecuzione dei singoli componenti;
- la descrizione.

L'adozione di questo strumento quale presidio delle attività di monitoraggio proprie della gestione applicativa ha comportato:

- una notevole riduzione dei tempi dedicati al monitoraggio giornaliero del batch;
- l'individuazione praticamente immediata delle anomalie e delle loro cause;
- la valutazione costante delle prestazioni del batch permettendo l'adozione di misure atte a prevenire i malfunzionamenti stessi.

Non è infine trascurabile il risultato insito nel progetto stesso che ha consentito di sperimentare, su una applicazione non "mission critical", l'elevata affidabilità ed il grado di efficienza delle componenti Open Source adottate.

Personalizzazione Mantis e e-Groupware

Fin dalle prime fasi di approccio dell'iniziativa di realizzazione della Unità Locale della Sicurezza (ULS) di Consip (struttura deputata alla prevenzione e gestione degli incidenti informatici), è emersa in maniera chiara la necessità di mantenere traccia delle segnalazioni e della loro evoluzione nel tempo. Tale necessità si è così tradotta nell'esigenza di reperire un applicativo che potesse assolvere alle funzioni di repository delle segnalazioni permettendo al contempo di strutturare un workflow che partisse dall'inserimento di una segnalazione e terminasse con la chiusura della stessa dopo che le attività di contenimento/prevenzione degli incidenti informatici erano state eseguite.

Data la volontà di contenere la spesa e la necessità di plasmare l'applicativo sulle specifiche procedure operative adottate si è deciso di individuare di un applicativo Open Source e procedere quindi con la necessaria personalizzazione.

A seguito di una ricerca sul panorama Open Source inerente alle tematiche in oggetto, sono stati scelti due applicativi Web:

- Mantis;
- e-Groupware.

Come precedentemente anticipato, i software utilizzati sono soggetti a licenze d'uso di tipo Open Source e precisamente licenze di tipo GPL (General Public License) i cui termini contemplano espressamente la possibilità di modificare il codice, di copiarlo e di ridistribuirlo con o senza modifiche, sia gratuitamente che per scopi commerciali.

Mantis è un applicativo ideato e sviluppato per fornire supporto alle attività di individuazione e risoluzione di bachi software, tramite l'inserimento e la tracciatura delle segnalazioni; e-Groupware è una suite di applicazioni che permettono a diversi gruppi di utenti di condividere strumenti e informazioni.

Entrambe queste applicazioni Web sono state modificate per renderle maggiormente compatibili con la gestione delle segnalazioni e, a seguito di una procedura di collaudo, sono state rilasciate nell'ambito degli ambienti di produzione.
In particolare su Mantis, che costituisce il repository delle segnalazioni e delle attività messe i campo dal team, sono state introdotte le seguenti modifiche:

- realizzazione di profili utente specifici per le attività dell'IRT;
- introduzione della terminologia specifica mediante cambiamento dei file linguistici;
- introduzione di campi a scelta multipla necessari per l'inserimento di una segnalazione di prevenzione/reazione integrati dalla specifica della tipologia dell'evento stesso;
- personalizzazione del workflow tramite la configurazione delle funzionalità di invio automatico di mail ai vari utenti coinvolti nella varie fasi di processo.

Su e-Groupware, invece, che costituisce lo strumento di collaborazione applicativa e condivisione della conoscenza, le modifiche sono state meno invasive e sono consistite nella:

- creazione di profili utente specifici per le attività dell'IRT;
- scelta delle applicazioni che sono state ritenute funzionali alle attività dei diversi profili utente;
- configurazione delle applicazioni scelte.

Successivamente, nell'ambito di un progetto pilota di collaborazione Consip/CNIPA, si è pensato di affinare e mettere a fattor comune il lavoro svolto dalla ULS Consip/MEF realizzando una suite di strumenti, denominata "Risorse per l'Incident Management" (RIM). È stato disposto che questo lavoro potesse essere messo a disposizione di tutte le amministrazioni aderenti al Sistema Pubblico di Connettività al fine di consentire una effettiva gestione delle informazioni prodotte dal processo di monitoraggio evoluto erogato dal CNIPA nell'ambito delle attività centrali di prevenzione degli incidenti informatici.

Le attività che sono state quindi messe in atto dal gruppo di lavoro progettuale sono state:

- il consolidamento e l'affinamento di quanto già realizzato dalla ULS Consip/MEF in ottica di un utilizzo da parte di tutte le amministrazioni aderenti ad SPC;
- la creazione di una serie di pagine web di presentazione, gestione e inquadramento dell'iniziativa;
- l'individuazione di un applicativo Open Source per una gestione semplificata dell'ambiente applicativo - Webmin con licenza di tipo BSD (Berkeley Software Distribution);
- lo sviluppo di specifiche funzionalità di backup e restore delle applicazioni e delle informazioni per garantire una corretta ripartenza a fronte di eventuali fault di sistema.

Inoltre al fine di consentire una totale portabilità delle soluzioni individuate, a prescindere dal hardware e dai sistemi operativi, è stato stabilito di realizzare il tutto con la tecnica del "virtual appliance" ossia della virtualizzazione di un ambiente, finalizzato ad uno specifico utilizzo, costituito dal sistema operativo e dalle relative applicazioni, preconfigurato e pronto per essere utilizzato in ambiente di esercizio.

In questo modo è stato possibile realizzare un DVD, che contiene tutto l'ambiente virtuale realizzato, e che può essere semplicemente utilizzato su un qualunque server che abbia:

- le caratteristiche hardware minime individuate;
- almeno una versione di VMware PLAYER funzionante.

Questo DVD sarà messo a disposizione di tutte le amministrazioni aderenti al Sistema Pubblico di Connettività.

Content Management System

Tra i progetti in fase di avvio si segnala quello relativo al Content Management System (CMS), software che consente la gestione strutturata dei contenuti dei siti internet, garantendo anche uniformità di navigazione e di layout grafico. L'esigenza immediata è stata di adottarne uno per l'intera Server Farm Internet gestita da Consip che annovera, tra gli altri, i siti internet del Dipartimento del Tesoro, della Ragioneria Generale dello Stato, del Dipartimento degli Affari Generali, del sito del Service Personale Tesoro e della Corte dei Conti.

È stata condotta, al riguardo, un'analisi su prodotti di mercato "licenziati" e prodotti Open Source. Da tali analisi non è emerso un prodotto che "chiavi in mano" soddisfi tutti i requisiti necessari per il CMS Consip/MEF. Per tale motivo oltre all'acquisizione dell'eventuale licenza, è necessaria una forte attività di personalizzazione. In questo senso si è valutata la possibilità di ricorrere a prodotti a codice aperto. La scelta di utilizzare un CMS Open Source mostra notevoli vantaggi, soprattutto, nel caso di riuso del software, dando impulso, indirettamente, anche all'interesse nazionale. In questo contesto, Consip ha scelto, lavorando attivamente con Cnipa, uno sviluppo del CMS di tipo "collaborativo", cooperando con altre Amministrazioni (es. Ministero dei Beni Culturali,

Istat ecc) e Imprese e creando, di fatto, un organismo assimilabile ad una comunità. Si è scelto una modalità operativa di tipo cooperativo, con la partecipazione anche di imprese, per consentire una gestione, nel tempo, del software con spese minimali e permettere di valutare collegialmente la promozione e la realizzazione di nuove versioni del prodotto, tenendo conto delle esigenze distintive di ogni partecipante. La presenza di più imprese, anche medio-piccole, interessate a creare e conservare le conoscenze tecniche sul prodotto mantenuto, evita la posizione dominante del fornitore, spesso contraddistinto da grandi aziende o multinazionali, che aveva sviluppato il software originario, abitualmente il solo a poter dare servizi di personalizzazione nel breve-medio periodo.

In considerazione della peculiarità del prodotto Open Source è stata richiesta al fornitore che svilupperà il CMS, l'applicazione di una metodologia di progetto strutturata in modo da salvaguardare e garantire nel tempo il mantenimento della versione base del prodotto. Secondo tale visione, il fornitore, in prima battuta, si dovrà impegnare:

- a sottoporre la propria candidatura a far parte della comunità di sviluppo del prodotto;
- a proporre alla comunità ufficiale tutte le personalizzazioni richieste da Consip/Amministrazione e che possono essere valutate di interesse generale da parte della community;
- ad attivarsi in prima persona nello sviluppo delle personalizzazioni richieste da Consip/Amministrazione indipendentemente dai tempi tecnici previsti dal board della community per la valutazione delle segnalazioni sottomesse;
- ad adeguare, in funzione dei feedback ottenuti dalla community, eventuali soluzioni già messe in atto per le esigenze di Consip/Amministrazione in modo da adeguarle e renderle rispondenti alle indicazioni ottenute;
- a garantire il completo allineamento della versione personalizzata per Consip con le evoluzioni della versione base.

Anche per la scelta del prodotto Open Source è stato indispensabile applicare una specifica metodica per garantire una selezione accurata del software. È stato importante, infatti, valutare:

- la maturità del prodotto;
- l'organizzazione della comunità degli sviluppatori;

- la penetrazione sul mercato;
- la disponibilità di documentazione;
- le funzionalità presenti;
- la scalabilità;
- la flessibilità;
- la riusabilità;
- la sicurezza;
- la disponibilità di sorgenti;
- l'indipendenza da tecnologie proprietarie;
- il livello di supporto/manutenzione;
- la possibilità di contribuire sia in termini di sviluppo che in termini di redazione di documentazione o traduzione in altre lingue.

Uno degli aspetti importanti di una comunità Open Source è quello di favorire il riuso delle componenti/funzionalità realizzate. Una volta sviluppate le funzioni aggiuntive del CMS l'intenzione è di renderle utilizzabili a tutte quelle Amministrazione che vorranno avvalersene. Ampliare la base di utenza è, infatti, fondamentale perché tanto è più ampia l'utenza tanto più elevata è la possibilità che i bug vengano scoperti e corretti rapidamente.

Collaborare in un contesto di questo tipo significa avere la capacità e la volontà di far evolvere il prodotto stesso attraverso l'introduzione di idee innovative. La comunità può essere vista come motore di innovazione che poi si trasmette anche alle aziende che fanno parte della comunità sia attraverso il prodotto che attraverso le persone che, con la loro conoscenza, hanno contribuito alla crescita tecnologica.

Mettere questo prodotto in condivisione con tutte le PA favorirà sicuramente non solo il riuso ma spingerà verso l'abitudine a valutare la qualità delle componenti aggiuntive sviluppate per Consip/Amministrazione perché dovranno essere riusate in un contesto diverso da quello per il quale è stato concepito. Se si pensa che le PA offrono tramite i propri siti web tipicamente tutte il medesimo prodotto ovvero servizi al cittadino e documentazione istituzionale, è molto probabile che a tendere, il risultato finale potrebbe essere un prodotto "su misura" per la Pubblica Amministrazione italiana. Per ottenere tali risultati tramite personalizzazioni con software proprietario il lavoro sarebbe sicuramente molto più lungo e articolato e con un costo complessivo superiore.
Altro elemento da non trascurare è il fattore mercato. Sul territorio italiano sono presenti ormai da anni piccole e medie aziende di servizi che da tempo hanno sviluppato competenze significative in ambito Open Source e nel caso particolare sui CMS. Queste aziende sono in grado di garantire presenza, flessibilità e competenza, cose che spesso le grandi aziende o le multinazionali hanno difficoltà ad assicurare.

8. Prospettive future

Il software Open Source si sta diffondendo significativamente e rapidamente nei diversi settori d'industria ed in particolare in quello della Pubblica Amministrazione, attraverso l'implementazione di soluzioni informatiche standard di facile riutilizzo e come valida alternativa nelle attività di sostituzione di piattaforme critiche, quali ad esempio Customer Relationship Management (CRM) e Content Management System (CMS).

Tale evoluzione, unita al crescente interesse nel modello di sviluppo in community, sta determinando un cambiamento nei modelli di business dei fornitori di software proprietario e negli accordi di partnership con le terze parti: sta cambiando, anche se lentamente, la modalità di approccio alle tematiche architetturali ed organizzative da parte della PA, che costituisce, attraverso la propria domanda, un importante elemento di legittimazione.

Nelle decisioni di acquisto, le organizzazioni iniziano a considerare l'Open Source come una alternativa al software proprietario. Non c'è più esitazione, come in passato, nel valutare il software Open Source a fianco di soluzioni proprietarie.

Comunque, i Chief Information Officier (CIO)[61], pur considerando la possibilità di inserire il software aperto nel possibile corredo delle suite software enterprise, evidenziano ancora la presenza di significative barriere nell'adozione del software aperto quali:

- la carenza di adeguato supporto commerciale;
- la necessità di una migliore interoperabilità e integrazione del software.

Quello che emerge è in generale la richiesta nei confronti del software Open Source di:

- indirizzare e risolvere specifiche necessità di business;
- incontrare i requisiti tecnici dell'organizzazione quali ad es. affidabilità e coerenza architetturale;

[61] *Valutazioni espresse nell'Executive Summary Report nell'ambito dell'Open Source ThinkTank 2007*

- essere competitivo in termini di TCO;
- fornire diverse opportunità di scelta nei servizi di supporto (di tipo commerciale);
- essere una soluzione a lungo termine (avere uno o più venditori e community solide).

Gartner[62] ha espresso fiducia nel fenomeno Open Source e ne disegna un futuro di forte sviluppo e consolidamento. La valutazione interessa nello specifico sia i prodotti Open Source, in continua evoluzione, sia le prospettive commerciali che questi avranno, sia l'impatto non più trascurabile che si imporrà a livello di quelle aziende che producono soluzioni tecnologiche o che le adottano o le sviluppano in proprio.
L'Open Source arriverà inevitabilmente all'interno di tutti i sistemi informativi; è impensabile evitare o rinviare scelte su questa tematica. Esistono ovviamente prodotti maturi in vera competizione con il software proprietario mentre altri ancora non lo sono; per l'Open Source è applicabile, ancor più dei software proprietari, una sorta di prospettiva darwiniana: possono sopravvivere solo le porzioni di codice, le applicazioni e le soluzioni più innovative e più solide.

La maturità dei prodotti software Open Source dipende sia dal livello dello stack software in cui si pongono e sia da quanto sono diffusi. Quindi prodotti con un potenziale di ampia diffusione, quali sistemi operativi, database, suite CMS, sono supportati da molteplici fornitori e sono diffusi in modo cross in diversi settori d'industria.
Le applicazioni inerenti specifici domini hanno un minore potenziale di mercato e quindi rivelano una maggiore lentezza del processo di maturità; talune applicazioni non diverranno mai mature a meno di un'integrazione all'interno di framework di prodotti commerciali.
Parte del software commerciale oggi è composta da software libero e si prevede che entro il 2011 almeno l'80% del software commerciale sarà costituito in quantità rilevante da software Open Source; questo trend di crescita, di convergenza e sovrapposizione appare inarrestabile; già oggi nelle applicazioni aziendali mission critical, la percentuale di software proprietario, quella di software sviluppato internamente e quella di Open Source si equivalgono[63].

[62] *Open Source Summit 19-21 settembre 2007 Las Vegas, N.*

[63] *Fonte e-Week: Open Parenthesis, Mark Driver, Gartner Open Source Summit 2007*

Le soluzioni Open Source tendono a generare una protezione sull'investimento da parte dei clienti, laddove le soluzioni proprietarie si fondano sulla protezione della proprietà intellettuale di fatto impedendo una reale "vendor indipendance" e imponendo situazioni di evidente lock-in.

Anche nel Settore Pubblico, i Governi ed altre organizzazioni IT pubbliche stanno considerando un approccio realistico all'uso del software aperto, principalmente guidato dalla necessità di ridurre i costi, imporre degli standard aperti, mantenere un certo grado di indipendenza dai fornitori e sviluppare nuove filiere produttive nazionali.

Sulla base di un survey condotto da Gartner nel 2006 [64] su un campione di quasi 100 organizzazioni pubbliche nel mondo, emerge che:

- il software Open Source è considerato una alternativa consolidata per i livelli più bassi dello stack software; la sua diffusione è notevolmente presente a livello di sistema operativo, middleware, database, application server;
- è in crescita l'adozione di software aperto di tipo applicativo; in particolare, le aree che presentano una maggiore crescita riguardano le applicazioni enterprise, tra cui i Content Management System, il CRM e la Business Intelligence;
- in termini di portfolio software, si conferma il trend che vede una crescita della partecipazione del settore pubblico nelle source-community. Si tratta di community con focus specifici che operano in modo analogo a quelle Open Source, in cui le organizzazioni esterne impiegano risorse umane e finanziarie nello sviluppo di software che viene rilasciato sotto licenza OSI;
- il driver più rilevante verso l'adozione di Open Source è rappresentato dalla conformità del software verso standard e processi di sviluppo aperti, dalla flessibilità e indipendenza dai fornitori;
- agilità, velocità di sviluppo, facilità dei processi di acquisto e approccio prototipale sono percepiti come i più significativi problemi risolti dall'Open Source, attribuendo nelle motivazioni poste all'adozione di software aperto maggiore peso alla velocità rispetto al costo;

[64] *Gartner : Public-Sector Open Source Survey, Worldwide – Aprile 2007*

- l' Open Source non è in diretta competizione con il software proprietario, ma è visto come una opportunità per veicolare la sostituzione di applicazioni "custom" sviluppate interamente su tecnologie proprietarie. Questo condurrà ad una crescita rilevante delle community in domini applicativi specifici;
- nelle grandi organizzazioni è carente il coordinamento centralizzato dei diversi progetti; sono ancora immaturi i processi di governance e non c'è ancora chiarezza sulle policy Open Source da adottare.

Allegati

Statistiche di diffusione

Figura 1. Adozione del software Open Source nel mondo

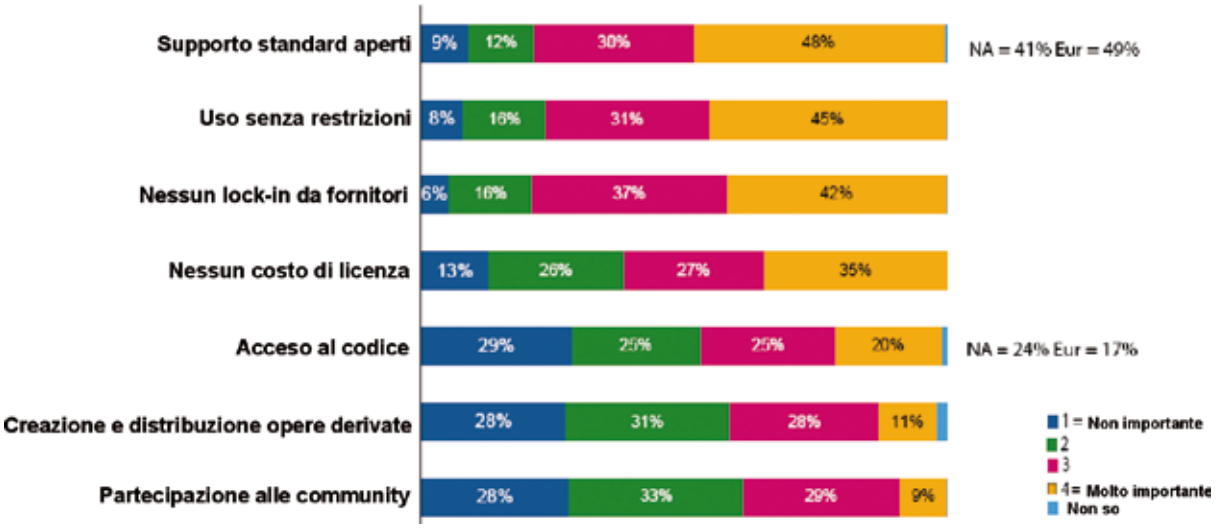
Con il colore rosso si indicano quei Paesi cha hanno assunto un ruolo istituzionale più incisivo nella diffusione del software aperto, a seguire i colori arancio, rosa e grigio. La distribuzione di punti sta ad indicare la granularità dei livelli di politica (locale, regionale, statale) di diffusione del software che implica la mancanza di una politica nazionale[65].

[65] *M. Berra: software Open Source e politiche dei governi- Paal 2007*

Contesto nord-americano ed europeo

Si riportano i risultati di un survey condotto nel 2007 da Forrester Consulting e sponsorizzato da Unisys che ha coinvolto i CIO di oltre 450 organizzazioni[66].

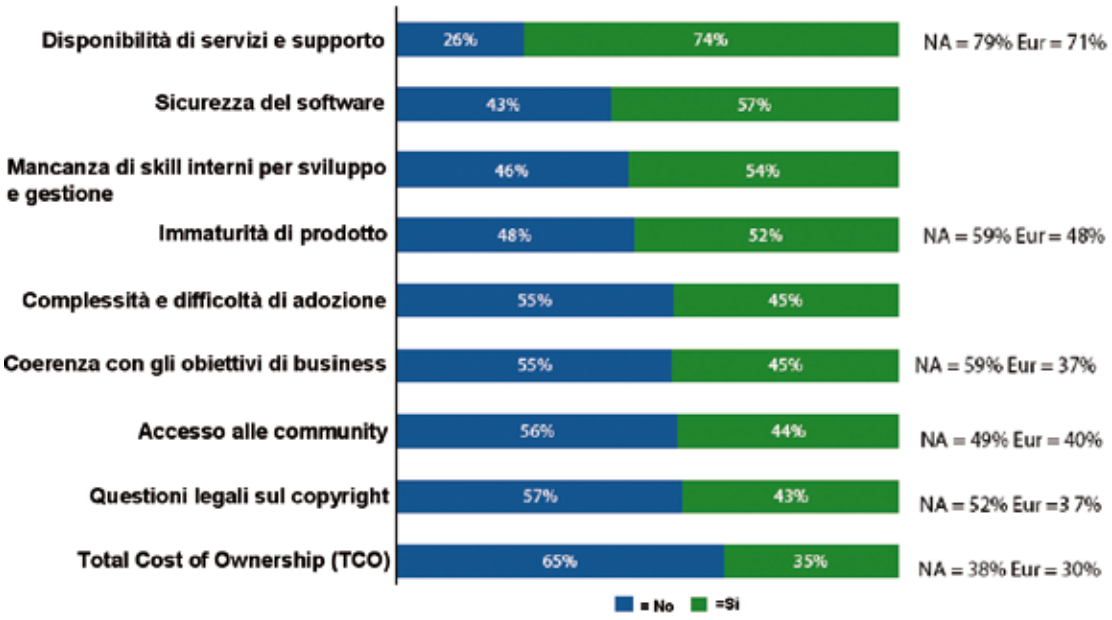
Figura 2. Attributi del software Open Source



Relativamente alle motivazioni connesse all'Open Source, si rileva come il supporto di standard aperti, l'utilizzo del codice senza vincoli e l'eliminazione dei lock-in dai vendor siano in ordine di importanza le tre principali motivazioni addotte all'adozione dl software aperto. Aspetti considerati di minore importanza sono la partecipazione diretta alle community e l'accesso al codice: le cause possono essere ricercate nella mancanza di propri skill tecnici e perché sono aspetti non direttamente correlati al business.

[66] *Forrester Consulting: "Open Source Software Expanding Role in the Enterprise" Marzo 2007*

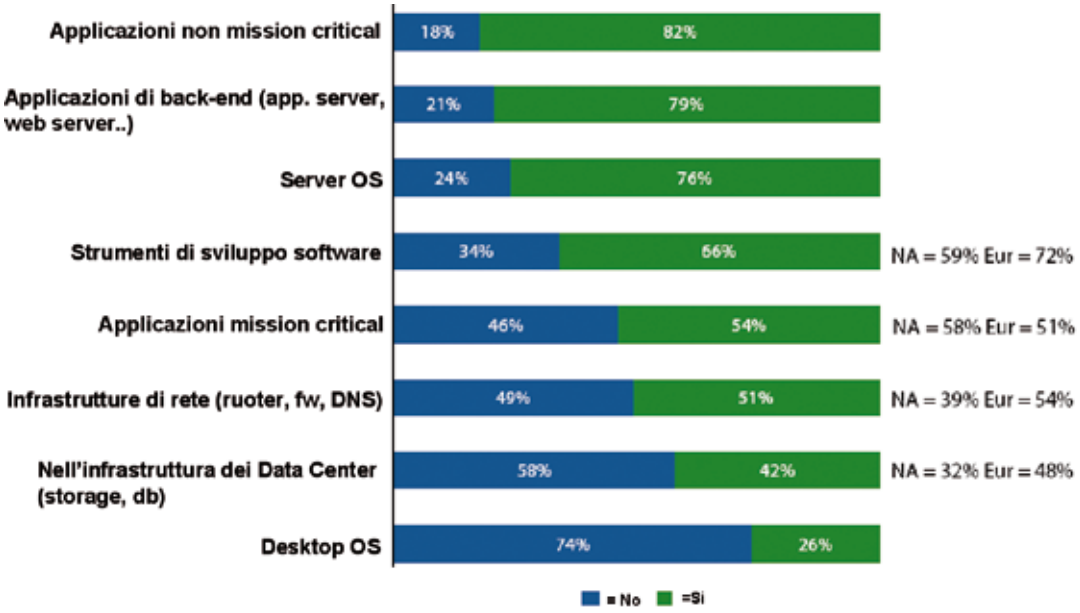
Figura 3. Punti di attenzione del software Open Source



La più importante preoccupazione che emerge riguarda la disponibilità di servizi e di supporto sul software Open Source utilizzato. Per il software più maturo (Jboss, Linux, Apache...) il supporto è da subito disponibile da molteplici fornitori commerciali, per molti altri prodotti non c'è chiarezza se e da chi viene fornito supporto.

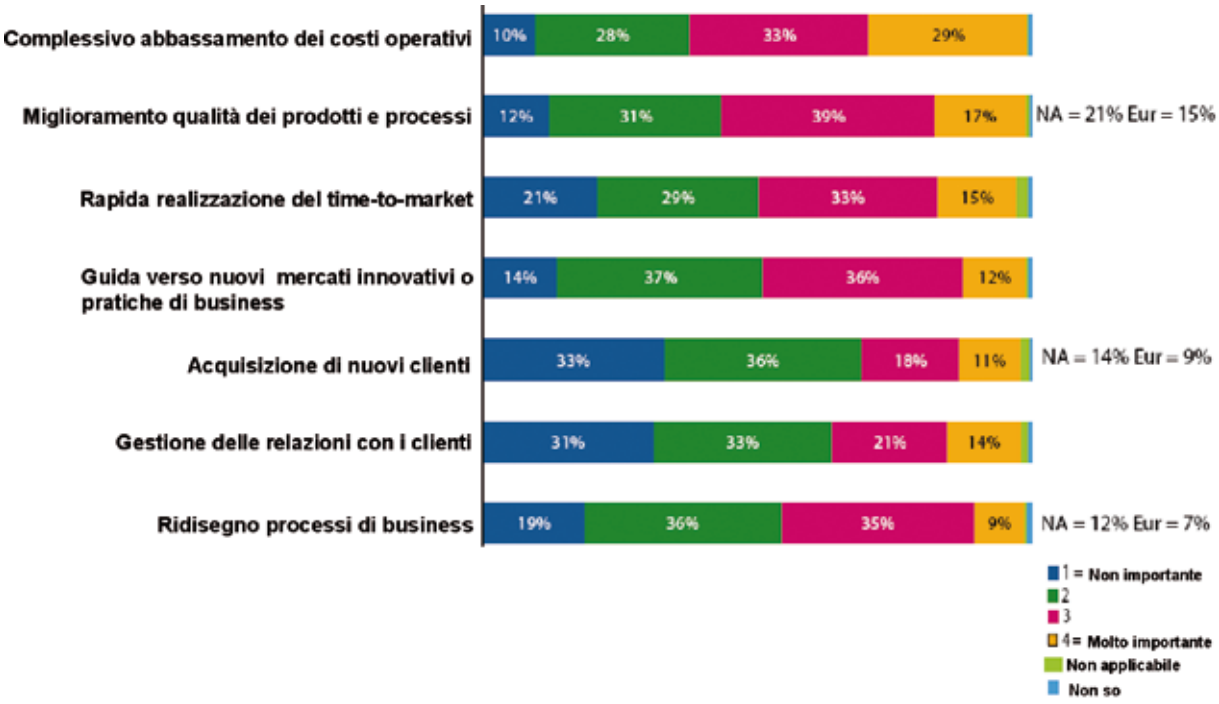
Altro aspetto critico riguarda la sicurezza del software Open Source. Tale evidenza deriva dal fatto che il codice sorgente è disponibile a chiunque, anche a coloro che hanno intenzioni malevoli; di converso c'è chi ritiene che tale caratteristica consenta la continua ispezione del codice identificandone le potenziali vulnerabilità.

Figura 4. Ambito di utilizzo applicazioni Open Source



Molte delle organizzazioni hanno indicato che non sono ancora pronte ad utilizzare software Open Source come specifiche applicazioni mission critical. Il trend che comunque emerge è che gradualmente il software Open Source si sta indirizzando verso le applicazioni di business. Si nota infatti che l'Open Source è comunque adottato nell'infrastruttura applicativa delle organizzazioni; tale indicatore è importante in quanto indica che l'Open Source diverrà sempre più parte integrante del futuro corredo software delle organizzazioni. Il 79% utilizza il software Open Source come Web e Application Server e Database. Per quanto riguarda l'adozione dell'Open Source nell'area Desktop, non c'è ancora una diffusione significativa. Il 26% delle organizzazioni utilizza software Open Source su desktop, mentre il 76% lo utilizza come sistema operativo su piattaforma server.

Figura 5. Benefici sul business



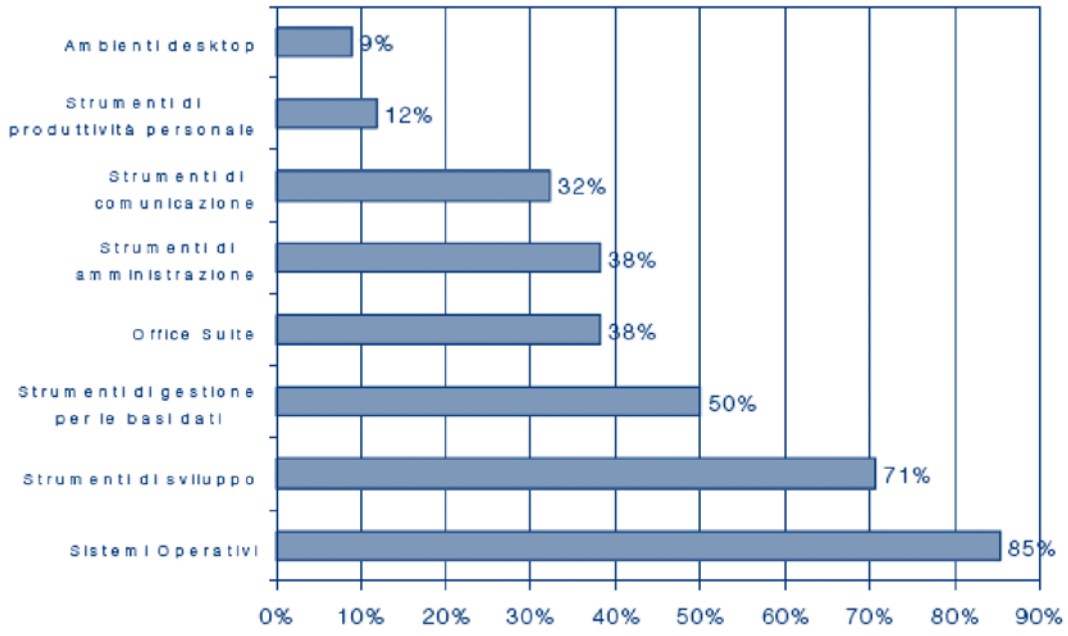
Anche se non necessariamente l'Open Source, da solo, rappresenti un modo sempre efficace per dare valore aggiunto al business rispetto al software proprietario, emerge che il software aperto assume un ruolo chiave nell'aiutare le organizzazione a ridurre i costi operativi, che è un beneficio che incide indirettamente sul business. Inoltre, emerge che l'Open Source è importante per il miglioramento della qualità dei prodotti e processi di una compagnia.;questo aspetto è coerente con le caratteristiche proprie dell'Open Source di alta qualità e flessibilità del software che possono consentire quindi cambiamenti e modifiche veloci e con minore effort.

Molti invece non considerano l'Open Source come leva per acquisire e mantenere nuovi clienti o per gestire le relazioni. La motivazione è che l'Open Source è tecnologia mentre l'impatto sul cliente si ha per come viene utilizzata la tecnologia e non per il suo valore intrinseco.

Contesto nazionale

L'utilizzo del software Open Source nella settore della Pubblica Amministrazione in Italia si limita essenzialmente alla parte infrastrutturale. Tale ritardo è dovuto, indipendentemente dalla qualità del prodotto, principalmente alla mancanza di una forte sponsorizzazione. Infatti la conoscenza (a volte addirittura la sua esistenza) è dipendente dalla capacità di aggiornamento del singolo. Una analisi di quanto sia diffusa l'adozione del software Open Source nell'ambito della Pubblica Amministrazione deriva da ad esempio dal Rapporto annuale 2006 del CNIPA sullo stato di informatizzazione della P.A..

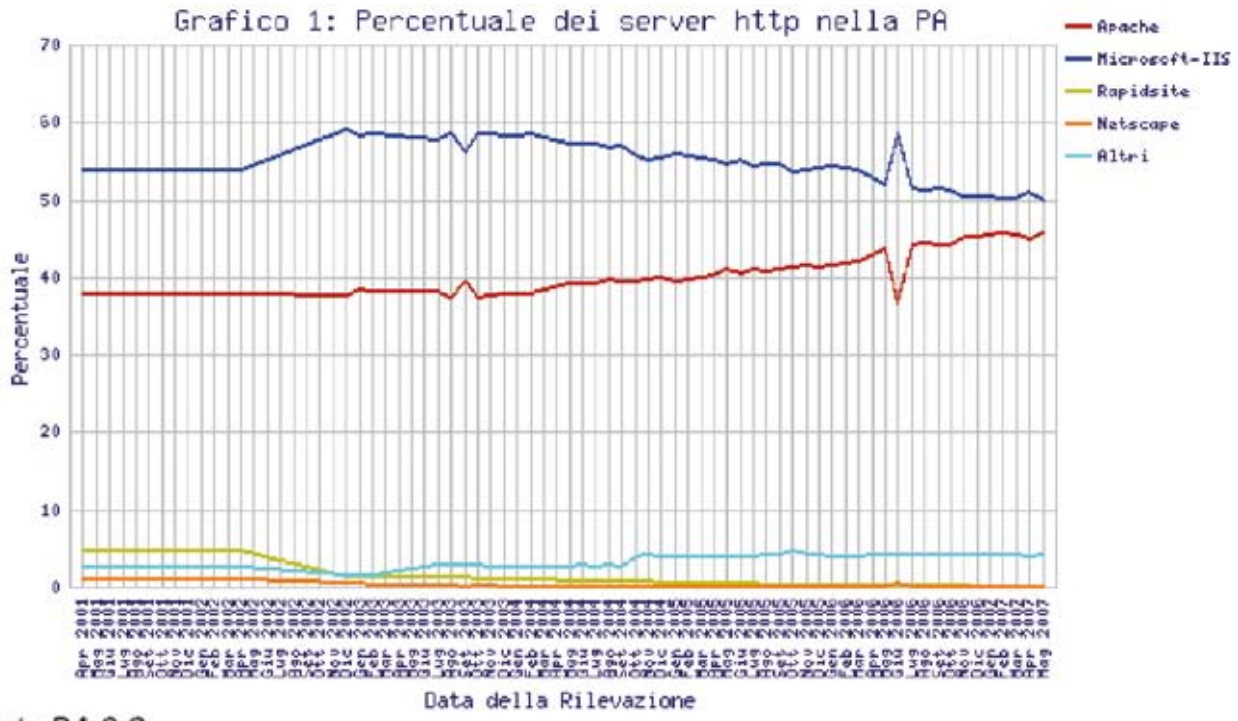
Figura 6. Ambito di utilizzo del software Open Source



Fonte: CNIPA - Rapporto annuale 2006 sullo stato di informatizzazione della P.A.

Il 72% delle amministrazioni utilizzano software Open Source; il principale utilizzo è sul lato back-office. Questo dimostra che il software aperto viene riconosciuto affidabile per il livello server e middleware. In sintonia con il precedente contesto, è di fatto scarsa la diffusione sul lato front office ovvero sul Desktop.

Figura 7. Diffusione degli Application Server nella PA



fonte P.A.O.S.

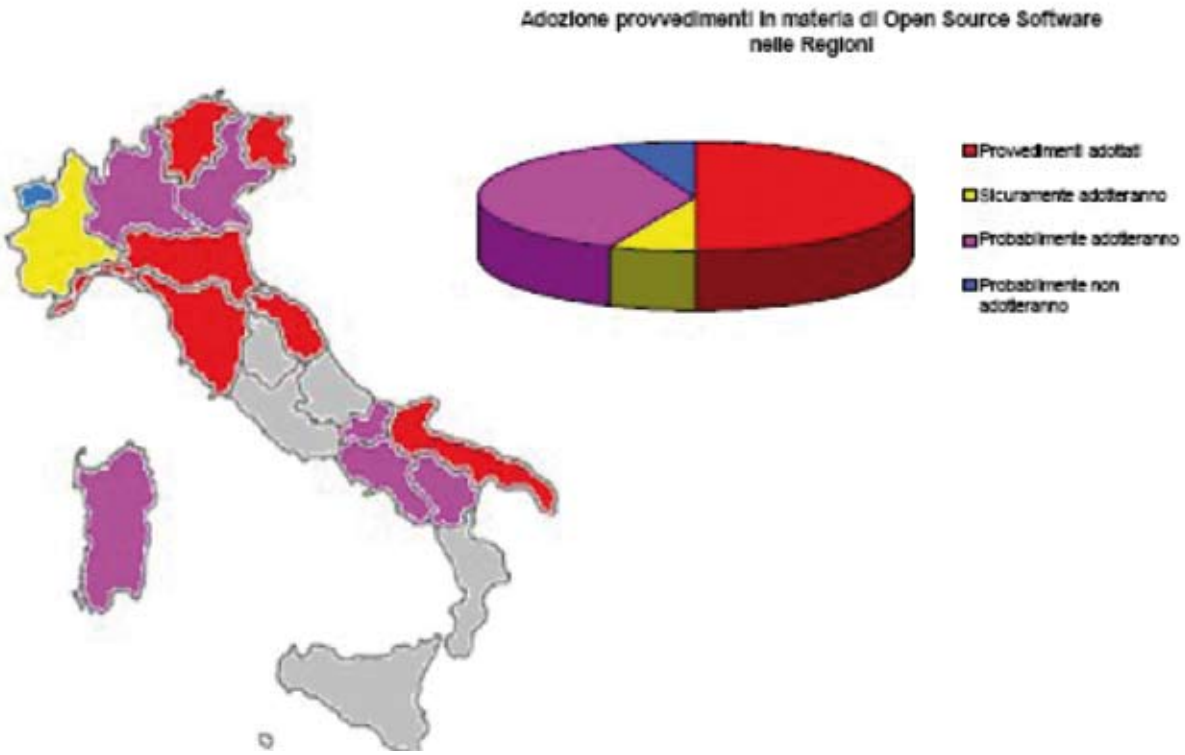
Più del 50% delle regioni e delle province e 40% dei Comuni fanno ricorso a software Open Source come sistemi operativi e application server.

Figura 8. Vantaggi nell'utilizzo del software Open Source

Fonte: CNIPA - Rapporto annuale 2006 sullo stato di informatizzazione della P.A.

Il 56% delle amministrazioni adduce quale vantaggio riscontrato nell'adozione di software aperto quello economico, dovuto all'iniziale abbattimento dei costi di licenza.

Figura 9. Diffusione della legislazione sul software Open Source nelle P.A.L.



Fonte: PAAL 2007

Nella figura si riporta la distribuzione dei provvedimenti di legge adottati nelle regioni; si assiste ad un inversione di marcia per quanto concerne l'adozione di Politiche (legge regionale o altro atto d'indirizzo) in favore del software a codice sorgente aperto[67].

[67] *F. Marzano: Le leggi regionali sul software libero e confronto – Paal 2007*

Bibliografia

S. Aliprandi

2003 Open Source e opere non software

2003 Open Source definition

M. Berra

2007 Software Open Source e politiche dei governi

M. Bertani

2005 Open Source

2005 Software Open Source e diritto d'autore

L. Bertini

2007 Appalti pubblici e innovazione: il procurement come strumento di supporto allo sviluppo della "Società della Conoscenza", Quaderni Consip-MEF

Beyon-Davies, P. Carne, C. Mackay, H & D. Tudhope

1999 Rapid Application development : an empirical review

D. Brink, L. Roos, J. Van Belle, J. Weller

2007 A model for the successful migration to Desktop OS

M. Castells

2001 The Internet galaxy: Reflections on the Internet, Business and Society

CNIPA

2007 Lo stato dell'informatizzazione delle PAC

S. Comino, F. Manenti, M.L. Parisi (Università degli Studi di Padova)

2007 From planning to mature:on the determinents of Open Source take off

S. Comino, F. Manenti
2005 Government policies supporting OS software for the mass market

D. Dalmas
2005 La diffusione del software libero

J. Dedrick & K. Kraemer
2001 China IT Report

P. Dohnam
2004 Ten rules for evaluating Open Source software

F. Dujinhouwer & C. Widdows
2003 Capgemini Open Source maturity model

I. Floyd, M. Jones, M. Twidale
2007 Patchwork Prototyping with Open Source Software

I. Floyd, M. Cameron Jones, D. Rathi, M. Twidale
2007 Web Mash-up and patchwork prototyping: user driven technological innovation with web 2.0 and Open Source software

S. Forge
2005 Towards an EU policy for OS software

Forrester Consulting
2007 OS Softwares expanding role in the enterprise

F. Galgano e P. Sammarco
2006 I nuovi contratti dell'informatica

J. Gardiner, P. Healey, K. Johnston, A. Prestedge
2003 The state of OS software in South Africa

Gartner Group
2007 Open Source Software Grows in the Public Sector
2006 Public Sector OS Survey, worldwide: getting there
2007 Long term trends that will radically alter licensing in the software market

E. Giannantonio
2001 Manuale di diritto dell'informatica

GITOC (Government Information Technology Officers Council)
2003 Using OS in the South Africa Government

P. Iyengar
2005 State of the information and communication technology industry in India.

Kirk ST.Amant & Brian Still
2007 Handbook of Research on Open Source Software

F. Marzano
2007 Le leggi regionali sul software libero a confronto

G. Mazzoni
2001 Open Source, GPL solida in Italia

A. Monti
2005 Software e Open Source

NetProject
2003 IDA OS software migration guidelines

Olliance Group, DLA Piper
2007 Open Source Think Tank: the future of commercial Open Source

V. Paruzzi
2006 Produrre sapere in rete in modo cooperativo

B. Perens
1999 The Open Source definition

E. Raymond
1997 The Cathedral and the Bazaar
2005 Information Industry Economic Report

G. Sanseverino
2007 Le licenze free e Open Source

P. Schmitz
2001 Use of OS in Europe, an IDA study

G. Sicchiero
2004 Linee di differenza tra contratti open e proprietari

R. M. Stallman
2002 Free Software Definition
2002 The GNU project
2002 Why software should be free
2005 Free software as a social movement

A. Todon
2004 Il software libero Open Source. Una dimensione sociale

Bibliografia Quaderni Consip

I Quaderni Consip sono disponibili sul sito web:
The Consip's Working Papers are available on the web site:

http://www.consip.it/on-line/Home/Pressroom/QuaderniConsip.html

XI/2007 – Antonio Ballarin: "La conoscenza e la sua gestione. Motivazioni economiche, modelli organizzativi e sistemi automatici"

X/2007 – G. L. Albano, G. Spagnolo and M. Zanza: "Regulating joint bidding in public procurement"

IX/2007 – Federico Dini and Giancarlo Spagnolo: "Buying Reputation on eBay"

VIII/2007 – Francesco Felici: "The role of core inflation in monetary deliberations: an application to the Norwegian economy"

VII/2007 – Luana Marchetti e Luca Nicoletti: "La gestione delle Identità Digitali - L'esperienza del MEF"

VI/2007 – Leonardo Bertini e Anna Vidoni: "Il Mercato Elettronico della Pubblica Amministrazione - MEPA Scenario, funzionalità e linee di tendenza"

V/2007 – Patrizia Cannuli, Mauro Galinelli, Serena Telara, Vito Saianella: "Il Data Warehousee il Portale di Business Intelligence del Programma di Razionalizzazione degli Acquisti della P.A."

IV/2007 – Gian Luigi Albano, Laura Carpineti, Federico Dini, Luigi Giamboni, Federico Russo e Giancarlo Spagnolo: "Riflessioni sull'impatto economico degli istituti innovativi del codice dei contratti pubblici relativi a lavori, servizi e forniture"

III/2007 – Gino Carucci, Roberto Marino, Cristina Corradi: "Diffusione dei processi ITIL e degli strumenti informatici per la gestione delle attività dei CED Applicazione delle best practices ITIL nel sistema gestionale di "La Rustica""

II/2007 – Michele Canalini: "Il Protocollo Informatico, Gestione Documentale e Workflow management. L'esperienza SIGED nel contesto MEF-DAG."

I/2007 - Federico Dini, Nicola Dimitri, Riccardo Pacini, Tommaso Valletti: "Formule di Aggiudicazione nelle Gare per gli Acquisti Pubblici"

Design:
CReA Catizone Randi e Associati Srl
www.crea-design.it

Finito di stampare
nel mese di gennaio 2008
presso