

# Sviluppare software libero in un'impresa

Amadeu Albós Raya

PID\_00145047



Universitat Oberta  
de Catalunya

[www.uoc.edu](http://www.uoc.edu)



## Indice

<b>Introduzione.....</b>	<b>5</b>
<b>Obiettivi.....</b>	<b>6</b>
<b>1. La produzione di software libero.....</b>	<b>7</b>
1.1. La produzione di software libero .....	7
1.2. Il progetto di software libero .....	9
1.3. La gestione del progetto .....	11
<b>2. La comunità di utenti.....</b>	<b>15</b>
2.1. Gestione della comunità .....	15
2.2. Caratteristiche della comunità .....	18
2.3. Gestione della qualità .....	20
2.4. Legalità e contributi .....	23
<b>3. Caso di studio.....</b>	<b>25</b>
3.1. L'impresa .....	25
3.2. I prodotti .....	26
3.3. La comunità degli utenti .....	27
3.4. Posizionamento ed evoluzione .....	29
<b>Riepilogo.....</b>	<b>30</b>
<b>Bibliografia.....</b>	<b>31</b>



## Introduzione

Attraverso questo modulo ci addentriamo nel mondo della produzione di software libero e nelle sue peculiarità più rilevanti riguardo al prodotto, all'impresa ed alla comunità di utenti.

In un primo momento, analizzeremo lo sviluppo di software libero dal punto di vista del progetto, considerando gli aspetti principali legati alla popolazione ed alla gestione del progetto, così come la partecipazione della comunità di utenti in diversi ambiti.

Mediante il progetto di software libero si realizza la relazione tra impresa e comunità di utenti. Il bilanciamento dei tratti particolari di questa relazione è fondamentale per il conseguimento degli obiettivi del progetto.

Successivamente, descriveremo le caratteristiche che rendono particolare la comunità degli utenti del software libero e la loro gestione da parte dell'impresa. Questa gestione accompagna, essendone complemento, la metodologia di produzione, ed implementa la strategia relazionale già citata.

Alla fine, il modulo si concluderà con la presentazione del caso di studio di un'impresa reale che produce software libero.

Questo modulo si sviluppa come una guida attraverso testi esistenti, il cui obiettivo è di approfondire in dettaglio le peculiarità dei diversi aspetti che si presentano e che sono rilevanti per la produzione d'impresa del software libero.

## Obiettivi

Al termine di questo modulo, si dovrà essere in grado di:

1. avere familiarità con la metodologia usata nella produzione di software libero.
2. comprendere l'importanza che riveste la comunità degli utenti nello sviluppo di prodotti basati sul software libero.
3. identificare ed analizzare i fattori rilevanti che influiscono sul successo della produzione di software libero.
4. capire l'importanza che gioca la formulazione precisa di una metodologia che permetta di rendere complementari gli sforzi dell'impresa e della comunità degli utenti.
5. approfondire le implicazioni dirette ed indirette del portare a compimento un progetto di sviluppo basato sul software libero.

## 1. La produzione di software libero

In questa prima sezione ci concentreremo sulla produzione di software libero dal punto di vista del suo sviluppo e della sua elaborazione, indipendentemente da eventuali modelli di business che lo sfruttino a scopo di lucro.

In diverse materie del master, soprattutto in quelle dedicate alla produzione del software (apice), viene analizzato in dettaglio il processo tecnologico che caratterizza l'elaborazione di software libero.

Questo processo tecnologico è complementare alle metodologie che ci consentono di formulare un progetto di cooperazione sostenibile e duraturo nel tempo. In questo senso, la collaborazione della comunità degli utenti nel progetto di software libero è fondamentale per ottenere una massa critica di utenti che aiuti a realizzare la realizzabilità del progetto.

Di conseguenza, una buona parte di queste metodologie e delle azioni conseguenti sono dirette ad offrire supporto e garanzie alle relazioni tra il progetto e la comunità di utenti. Per renderci conto dell'importanza di questa relazione, è sufficiente dare un'occhiata alle risorse che i progetti di software libero più popolari offrono alle loro comunità di utenti.

### Progetti popolari

Ad esempio, OpenOffice.org (<http://contributing.openoffice.org/>) e Mozilla (<http://www.mozilla.org/contribute/>).

Per sviluppare questi concetti, nei prossimi paragrafi descriveremo tre punti di vista complementari. Inizieremo presentando alcune idee di base sulla produzione del software libero. Poi forniremo brevemente i principali dettagli sui passi da compiere per mettere in moto un progetto basato sul software libero. Infine, approfondiremo i principali aspetti che caratterizzano la gestione del progetto di software libero.

### 1.1. La produzione di software libero

La produzione di software libero, allo stesso modo che per qualsiasi altro software, risponde alla necessità di risolvere una problematica tecnologica concreta. concreta<sup>1</sup>.

<sup>(1)</sup>Ad esempio, aggiungere delle funzioni ad un'applicazione o risolvere errori di funzionamento.

Anche se il processo tecnologico di raffinare e apportare evoluzioni ad un'applicazione di software libero può presentare molte somiglianze rispetto ad un'applicazione basata sul software proprietario, la differenza che le dà

l'apertura del modello le conferisce un funzionamento particolare. Il carattere aperto e di cooperazione della sua produzione incide sulla struttura evolutiva quantitativa e qualitativa, di versione in versione.

Sono molti gli autori che hanno scritto testi sulle particolarità che implica produrre software libero. Dal momento che questo modulo non è destinato a descrivere in maniera esaustiva le suddette particolarità, già ampiamente trattate in altre materie, ci concentreremo solo su alcune, le più rilevanti per il nostro caso.

Per questo motivo, prenderemo spunto da alcuni dei concetti presenti nel saggio *La catedral y el bazar*, di Eric S. Raymond, nel quale viene fatta un'analisi delle particolarità del software libero, e viene trattato soprattutto il caso GNU/Linux.

- **L'origine della produzione**

A grandi linee, la produzione del software libero nasce a partire dalle necessità proprie dell'utente o dello sviluppatore nella sua attività quotidiana. Questo significa che la collaborazione nello sviluppo del software comincia tramite la ricerca di un problema la cui soluzione ci sembri interessante, rilevante o necessaria.

- **La comunità di utenti**

La comunità di utenti del software libero, che racchiude tanto utenti finali quanto sviluppatori e programmatori, è la base che dà un senso alla definizione di sviluppo di software libero.

Trattare gli utenti come dei collaboratori nel progetto di produzione è la maniera più facile per migliorare e perfezionare rapidamente il codice (sempre che la base di collaboratori sia sufficiente).

Per questo, i collaboratori rappresentano una delle risorse più preziose nello sviluppo dell'applicazione, e pertanto è prezioso anche riconoscere le buone idee e le soluzioni che propongono.

- **Le versioni dell'applicazione**

Una delle caratteristiche della produzione di software libero è la riutilizzazione e la riscrittura del codice originale, che porta ad ottenere un codice nuovo, senza errori, con nuove funzionalità o con un miglior rendimento (tra le altre cose).

Dall'altra parte, nei progetti di sviluppo di software libero si promuove la liberazione rapida e frequente del codice, così da rendere l'attività del progetto dinamica ed incessante.

- **Il coordinamento della produzione**

La persona (o le persone) che coordina il progetto deve saper gestire il potenziale globale della comunità di utenti, dirigendo l'evoluzione del progetto senza coercizioni, ma sfruttando i mezzi e le sinergie che offre una rete come Internet.

#### Web consigliato

E. Raymond (1997). *The cathedral and the bazaar* (<http://www.catb.org/~esr/writings/cathedral-bazaar/>).

#### Origini della produzione

Una buona parte dei fondamenti del software libero si basa sulla pubblicazione degli aggiustamenti o degli sviluppi concreti che realizzano dei lavoratori nell'esercizio quotidiano delle loro attività lavorative.



L'eredità del codice dell'applicazione e della sua gestione del coordinamento è importante per il futuro del progetto di sviluppo del software libero. La scelta del successore chiamato a controllare ed a gestire la produzione non deve essere lasciata al caso.

## 1.2. Il progetto di software libero

A complemento delle considerazioni tecnologiche e funzionali delle applicazioni basate sul software libero, uno degli obiettivi primordiali di ogni progetto di software libero è la diffusione dell'applicazione o l'ottenimento di una massa critica di utenti.

È poco utile per il futuro del progetto, quindi, che il codice generato, anche se può risolvere problemi o carenze concrete, non sia conosciuto ed utilizzato dagli utenti potenziali. Inoltre costituisce un obiettivo necessario per il suo mantenimento ulteriore e per la sua evoluzione nel tempo. Nel caso del software libero, il compimento di questo aspetto è fondamentale per la creazione di una comunità di utenti stabile e duratura.

Esistono diverse guide che, in maggior o minor misura, apportano i concetti necessari alla creazione ed alla gestione di progetti basati sul software libero. In questa parte svilupperemo la questione attraverso l'articolo Free Software Project Management HOW TO di Benjamin Mako, che passa in rassegna le principali peculiarità del progetto dal punto di vista pratico.

### Web consigliato

**B. Mako** (2001). *Free Software Project Management HOW TO* (<http://mako.cc/projects/howto>).

### Messa in moto

Prima di lanciare il progetto basato sul software libero, è molto importante disegnare una struttura solida che formi una base per poter poi sopportare il processo di sviluppo con garanzie sufficienti.

In generale, la struttura basica del progetto deve dare risposta ai seguenti aspetti:

- La necessità di creare un progetto nuovo, a causa di idee e obiettivi propri o dell'esistenza di progetti simili.
- La definizione delle principali caratteristiche dell'applicazione (funzionalità, licenza, numerazione, ecc.).
- L'infrastruttura di base a supporto della diffusione del nuovo progetto e della collaborazione nel suo processo di sviluppo (pagina web, posta, ecc.).

## Gli sviluppatori

Una volta messo in moto il progetto, il nostro secondo obiettivo sarà l'integrazione, il coinvolgimento e il consolidamento degli utenti e degli sviluppatori dell'applicazione. Quanto a questi ultimi, dovremo essere in grado di mettere in atto politiche e strategie che ci consentano di definire e di strutturare la loro collaborazione.

Le politiche di cooperazione devono dare soluzione principalmente a due obiettivi concreti:

- Il coordinamento della produzione interna e di quella esterna, che include la delega delle responsabilità e i protocolli di accettazione dei contributi.
- La gestione della produzione, come, ad esempio, la struttura dei rami dell'attività di sviluppo e dei magazzini associati.

## Gli utenti

Nei prodotti basati sul software libero, spesso gli utenti sono anche sviluppatori (e viceversa). Uno dei principali obiettivi che dobbiamo tenere in considerazione sono le prove o i test dell'applicazione, siano essi funzionali, operativi, di qualità, ecc.

## L'infrastruttura di supporto

L'attività quotidiana del progetto fondato sul software libero non si potrebbe realizzare senza un'infrastruttura di supporto adeguata agli obiettivi di cooperazione dello stesso.

Nella messa in moto del progetto si sono già compiute le prime azioni per realizzare questo sistema di infrastrutture, ma una volta che il sistema si attiva, è necessario adeguare, migliorare e rinsaldare le risorse esistenti d'accordo con l'evoluzione del progetto.

## L'applicazione

Senza dubbio, la componente più rilevante del progetto è quella da cui dipende il resto degli aspetti considerati è l'applicazione. Una delle caratteristiche più rilevanti che deve contenere l'applicazione è che l'utente abbia sufficienti garanzie sul funzionamento di ognuna della versioni che vengono lanciate sul mercato.

Il lancio delle versioni è un tema delicato su cui conviene soffermarsi. A grandi linee, terremo a mente questi aspetti:

### Risorse abituali

Alcune delle risorse abituali nei progetti di software libero sono: documentazione, mailing list, sistemi di controllo di errori e di versioni, forum, chat, wiki, ecc.

- Il controllo delle revisioni in termini di funzionalità e di correzione degli errori (versioni alfa, beta, distribuzione candidata, ecc.).
- Quando lanciare la versione completa, ovvero quando il codice sarà pronto per offrire le garanzie che ci aspettiamo e che si aspettano gli utenti.
- Come lanciare la versione (incartamento, codice fonte, formato binario, ecc.).

## Diffusione del progetto

Infine, e come avevamo già anticipato, diffondere l'esistenza del progetto è importante per il progetto stesso, ma questo compito deve essere riproposto di continuo, nel tempo, se si vuole che le fondamenta si consolidino.

Ad ogni passo avanti del progetto, dovremo pensare a far risaltare il progetto attraverso liste di posta relazionate con il software libero o attraverso Usenet, inserire il progetto in altri portali pubblici (come Freshmeat o Sourceforge), o anche ad annunciare le nuove versioni dell'applicazione nelle liste di posta del progetto stesso.

### 1.3. La gestione del progetto

In questa sezione approfondiremo gli aspetti della gestione del progetto che, in quanto promotori dello stesso, dovremo tenere in considerazione per dotarlo di garanzie di successo.

I concetti che presentiamo in questa sezione sono complementari a quelli esposti nelle sezioni precedenti, dato che rendono possibile operare in concreto e migliorare le diverse azioni che abbiamo descritto. Per questo è possibile che vi siano delle coincidenze dirette o indirette con questi temi.

Per fare una relazione degli aspetti di base della gestione del progetto basato sul software libero, terremo conto delle considerazioni espresse in *Producing Open Source Software* di Karl Fogel, e nello specifico del quinto capitolo, intitolato 'Money'.

## Finanziamento

Le caratteristiche peculiari dei progetti di software libero fanno sì che molti dei contributi siano sovvenzioni informali (ad esempio, quando il lavoratore di un'impresa pubblica gli aggiustamenti che ha apportato al codice nell'esercizio delle sue funzioni).

### Web consigliato

K. Fogel (2005). *Producing Open Source Software: How to Run a Successful Free Software Project* (capitolo 5 "Money"). (<http://producingoss.com/en/money.html>).

Nonostante questo, vengono fatte anche donazioni e sovvenzioni che consentono di ottenere entrate dirette per il funzionamento del progetto, ma bisogna tener conto della gestione di questi fondi, dato che una buona parte dell'appoggio che riceve un progetto di software libero si basa sulla credibilità dei suoi partecipanti.

### **Tipi di partecipazione**

Esistono molti tipi (e molte possibili combinazioni) di partecipazione in forma finanziaria ad un progetto di software libero. Inoltre, in questo modello di finanziamento influiscono anche aspetti che esulano dal progetto stesso, ma dipendono dal contorno e dal contesto della messa in opera.

Sommariamente, si può dire che la partecipazione in un progetto di software libero ha una relazione con la collaborazione dei suoi partecipanti, con il modello di business che utilizza l'impresa che lo promuove (nel caso che esista), con le azioni di marketing promosse, e con le licenze dei prodotti coinvolti e le donazioni effettuate.

### **Contratti indefiniti**

Il team di sviluppatori dell'applicazione è molto importante per lo sviluppo del progetto e per la sua evoluzione futura. La stabilità e la permanenza dei partecipanti nei loro ruoli di responsabilità consente di rinsaldare le basi e la credibilità del progetto di fronte alla comunità degli utenti.

### **Decentramento**

Una delle caratteristiche più rilevanti (e desiderabili) delle comunità di utenti di software libero è la distribuzione e quindi il decentramento delle decisioni che vengono prese all'interno del processo.

L'organizzazione del progetto, quindi, deve tener conto di questa struttura, per poterla sfruttare per stimolare la motivazione e rinforzare la comunità degli utenti dell'applicazione, di modo da far sì che il consenso sorga dalla stessa interazione tra i membri.

### **Trasparenza**

L'aspetto precedente, legato al decentramento, ci dà un'idea della trasparenza che deve ergersi sovrana nella relazione tra progetto e comunità.

#### **Un progetto stabile**

La credibilità è fondamentale per tutti gli attori direttamente o indirettamente legati al progetto, dato che non è possibile trasferirla agli eventuali sostituti, e che la sua perdita può compromettere il futuro dell'applicazione e del progetto stesso; pertanto, dobbiamo prendere le giuste iniziative per controllare e gestire attivamente il progetto.

Tanto gli obiettivi del progetto quanto le linee di evoluzione dell'applicazione devono essere chiari e ben noti a tutti coloro che sono coinvolti nei processi. Per questo, l'influenza del promotore sull'evoluzione futura deve manifestare i tratti di un comportamento onesto e trasparente, tali da garantire la credibilità del progetto *credibilidad del proyecto*<sup>2</sup>.

<sup>(2)</sup>Ad esempio, il manifesto di Openbravo (<http://www.openbravo.com/es/about-us/openbravo-manifesto/>).

## Credibilità

La credibilità del progetto (insieme con la credibilità dei suoi membri) è comparsa spesso in molti degli aspetti che abbiamo considerato finora. La sua rilevanza è molto legata alla comunità degli utenti del software libero e dà vita ad una condizione importante per il mantenimento nel tempo.

Il denaro o la posizione gerarchica non sono in grado di infondere la credibilità necessaria alle azioni di ogni membro in ogni momento. Pertanto la metodologia, i procedimenti o i protocolli stabiliti, o il funzionamento o la gestione operativa devono essere gli stessi per tutti senza eccezioni.

## Contratti

L'assunzione dei lavoratori è un aspetto che va curato nei minimi particolari nei progetti di software libero, a causa delle sue ripercussioni sulla struttura e sul funzionamento. Bisogna stare attenti a che tutti i dettagli ed i processi relazionati con le assunzioni restino aperti e trasparenti.

Per questo, è importante discutere questi cambiamenti con la collaborazione della comunità degli utenti, fino al punto che in alcuni casi può essere preferibile assumere direttamente sviluppatori della comunità con permessi di scrittura nel magazzino ufficiale (*committers*).

## Risorse

Il progetto di software libero non si basa solo sull'evoluzione e sul mantenimento del codice di un'applicazione basata sul software libero, ma deve comprendere anche altri aspetti di supporto complementari

### Risorse complementari

È questo il caso della gestione della qualità del codice prodotto, della protezione legale dei contributi, della documentazione e dell'utilità dell'applicazione e della consegna di risorse infrastrutturali per la comunità del software libero (pagine web, sistemi di controllo delle versioni, ecc.).

Queste risorse possono essere motivo di differenze significative nella diffusione e nell'acquisizione di popolarità tanto dell'applicazione quanto del progetto, nella comunità degli utenti del software libero.

## Marketing

Infine, anche se si tratta di un progetto basato sul software libero, devono essere applicate misure di marketing per la sua diffusione e l'acquisizione di popolarità tanto dell'applicazione quanto del progetto in generale.

Per questo è importante ricordare che l'intero funzionamento del progetto si trova esposto al pubblico, e che ognuna delle affermazioni che si fanno può essere facilmente dimostrata o smentita. Stabilire delle misure di controllo dell'immagine e del funzionamento del progetto permette di guadagnare in credibilità, trasparenza e verificabilità.

In mezzo a queste misure, va sottolineata l'importanza che assume il mantenere una politica aperta, onesta ed oggettiva rispetto ai progetti dei concorrenti. Da un lato perché riafferma un valore stabile per la comunità degli utenti, e dall'altro perché favorisce lo sviluppo di strategie di cooperazione tra progetti connessi.

## 2. La comunità di utenti

Come avevamo messo in rilievo fin dalla prima parte di questo modulo, il ruolo che gioca la comunità degli utenti del software libero è molto importante nel paradigma di sviluppo del software libero.

Tanto gli utenti come gli sviluppatori che sono parte della comunità collaborano al mantenimento, al supporto ed all'evoluzione dell'applicazione nel corso del tempo, assicurando così la coesione e la stabilità del progetto.

Di conseguenza, la loro partecipazione è fondamentale per dotare di garanzie valide gli obiettivi del progetto, e deve essere considerata come tale da qualunque organizzazione a scopo di lucro che desideri cogliere l'opportunità di un business basato sulla produzione del software libero.

Per questo motivo la relazione tra comunità ed impresa deve fondarsi sulla credibilità e sulla trasparenza di tutte le azioni e di tutte le decisioni che vengono prese, cosicché entrambe le parti possano trarre profitto da questa relazione. Non è invano che si richiede che il posizionamento dell'impresa quanto ai prodotti basati sul software libero sia ben definito e ben strutturato, ciò serve infatti a favorire la formazione di una rete di collaboratori intorno all'impresa.

Bisogna tener conto del fatto che la comunità di utenti è un'organizzazione dinamica e che evolve nel tempo, per cui sarà necessario stabilire delle metodologie di gestione per mantenere sempre la relazione ad un ottimo livello. Questa gestione comprende la fissazione di procedimenti per identificare lo stato attuale della comunità, valutare la qualità dei contributi al progetto da parte dei membri e definire gli aspetti legali legati a questi contributi.

Nei prossimi paragrafi prenderemo in esame ognuno di questi aspetti singolarmente.

### 2.1. Gestione della comunità

Per raggiungere gli obiettivi del progetto, l'impresa che intraprende un progetto di sviluppo di software libero deve strutturare minuziosamente la sua relazione con la comunità di utenti.

Nella prima parte di questo modulo abbiamo già menzionato i principali aspetti sui quali edificare un progetto di software libero. Nel caso in cui un'impresa faccia da promotrice del progetto, sarà necessario stabilire ed organizzare una strategia adeguata agli obiettivi dell'impresa, ma sempre tenendo in debita considerazione che deve offrire contropartite per la collaborazione che si aspetta di ottenere dalla comunità degli utenti.

In questo senso, e così come in qualsiasi altro progetto di software libero, aspetti come la credibilità o la trasparenza - tra gli altri ? giocheranno un ruolo molto importante nella creazione di una comunità di utenti intorno al progetto.

Ben Collins-Sussman e Brian Fitzpatrick hanno identificato e classificato le diverse *Open Source* che può adottare un'impresa basata sullo sviluppo di un software libero nella conferenza intitolata 'What's in it for me?' di OSCON 2007.

Questa classificazione tiene conto delle due componenti principali della relazione tra impresa e comunità:

- Da un lato, l'orientamento, la struttura ed il funzionamento generale del progetto, così come le responsabilità dell'impresa in esso.
- Dall'altro, i benefici e gli inconvenienti per l'impresa e per la comunità di utenti che scaturiscono dalla scelta di una strategia concreta per portare il progetto a termine.

In effetti, il lavoro di Collins-Sussman e di Fitzpatrick si avvicina molto ad essere un manuale di buone maniere atte ad instaurare una relazione sana e soddisfacente tra impresa e comunità di utenti.

Di seguito presenteremo brevemente i tratti principali di questa classificazione di strategie *Open Source*.

### ***Fake Open Source***

Questa strategia si fonda sulla concessione del codice fonte dell'applicazione con una licenza non approvata dall'OSI.

A dire il vero, non si tratta di una strategia *Open Source* vera e propria, dato che non solo ne vengono persi i benefici, ma alcuni membri della comunità potrebbero perfino arrivare a boicottare il progetto.

#### **Web consigliato**

**B. Collins-Sussman; B. Fitzpatrick** (2007). "What's in it for me?"

(<http://www.youtube.com/watch?v=ZtYJoatnHb8>).

#### **Web consigliato**

Open Source Initiative (<http://www.opensource.org/>).



Nonostante questo, però, il progetto può avere una copertura mediatica e così catturare l'attenzione con un costo o uno sforzo relativamente bassi.

### ***Throw code over the wall***

Questa strategia è simile alla precedente, ma questa volta l'impresa concede il codice con una licenza approvata dall'OSI, anche se continua a non preoccuparsi sul futuro del progetto.

Quanto sopra significa che se concede il codice e poi se ne dimentica, l'impresa dà di sé un'immagine di scarsa credibilità, dal momento che lancia un'applicazione senza che esista una comunità di utenti che permetta di mantenere il progetto attivo. Un possibile inconveniente è infatti che si creino comunità alternative che sviluppino il software indipendentemente dagli obiettivi dell'impresa.

### ***Develop internally, post externally***

Questa strategia si basa sullo sviluppo dell'applicazione all'interno dell'impresa, per poi pubblicarne gli aggiornamenti in un'area pubblica.

In questo modo l'impresa migliora tanto le relazioni pubbliche con la comunità degli utenti quanto la credibilità all'interno del mondo del software libero. Da parte sua, la comunità potrebbe collaborare attivamente al progetto. Nonostante questo, il fatto che lo sviluppo avvenga totalmente all'interno può dare impulso alla nascita di comunità parallele ed indipendenti dalle strategie dell'impresa (che genera un certo grado di sfiducia).

### ***Open monarchy***

Questa strategia si sviluppa attraverso la manifestazione al pubblico tanto delle discussioni quanto del contenuto delle applicazioni, anche se gli utenti che godono di diritti su di essa sono interni all'impresa.

Con questa strategia migliorano sensibilmente la credibilità e la trasparenza delle imprese e dei contributi della comunità (cosa che ha effetti positivi sul codice), anche se è l'impresa ad avere comunque l'ultima parola su tutte le decisioni che vengono prese. Quest'ultimo aspetto costituisce un rischio per la buona conservazione della comunità nel lungo periodo, ed esiste perfino il rischio di una separazione nel progetto, che potrebbe snodarsi su due distinti cammini (*fork*)

### *Consensus-based development*

Questa strategia si serve di praticamente tutte le possibili forme di relazione tra impresa e comunità, dato che praticamente tutto viene realizzato in forma pubblica.

Il progetto, in questo caso, si articola tanto attraverso la presa decentralizzata di decisioni, quanto attraverso sistemi di funzionamento di tipo meritocratico tra i collaboratori.

Queste caratteristiche si riversano a formare un modello sostenibile nel lungo periodo con volontari di alto profilo, dato che l'impresa guadagna in credibilità, trasparenza e fiducia di fronte alla comunità ed alle altre imprese basate sul software libero.

Di contro, nel breve periodo i benefici sono scarsi ed il volume di lavoro è significativo. Pertanto il ruolo dei leader del progetto è rilevante per il funzionamento strategico dell'intera organizzazione.

### **2.2. Caratteristiche della comunità**

La comunità di utenti del software libero è un'organizzazione dinamica ed in evoluzione, nel senso che esistono diversi fattori che influiscono sulla sua situazione e ne modellano in maggior o minor misura le tendenze.

Se si fa una riflessione su un progetto di software libero, si scopre che è preferibile creare quanto prima una solida comunità di utenti attorno all'applicazione, dato che una buona parte del suo successo e del conseguimento dei suoi obiettivi emergono proprio grazie alla comunità.

Una volta creatasi la comunità, è importante fare un piano di strategie che permettano non solo di mantenerla solida, ma anche di ampliarla e di farla evolvere per lo meno allo stesso ritmo del prodotto stesso. Uno step precedente a qualunque tipo di azione intesa in questo senso è di conoscere con precisione ed in ogni momento lo stato della comunità e la sua tendenza evolutiva più recente ed aggiornata, in relazione al progetto.

Cogliere appieno ed in ogni momento lo stato della comunità di utenti può essere relativamente complicato, sul piano pratico, soprattutto per le sue caratteristiche intrinseche di distribuzione e decentramento.

Nonostante ciò, si può comunque individuare una serie di indicatori che agevolino le decisioni, capaci come sono di fornire un quadro della situazione abbastanza realistico.

Nell'articolo di Crowston e Howison 'Assessing the Health of a FLOSS Community' viene proposta una guida semplice, ma efficace, per identificare e soppesare compiutamente lo stato di una comunità di utenti di software libero. Questa guida tiene in considerazione i principali indicatori cui prestare attenzione quando si voglia fare una valutazione dello stato di salute della comunità e, pertanto, del progetto basato sul software libero.

Nei prossimi paragrafi forniremo alcune delle conclusioni tratte nell'articolo.

### Ciclo di vita e motivazioni

Vari autori sono d'accordo nel considerare che i progetti si iniziano in seno a un ristretto gruppo di promotori, per poi trovare una loro struttura e svilupparsi in forma pubblica.

Una volta messo in moto il progetto, va iniziata una seconda fase relativa al perfezionamento progressivo del concetto di base. Questo implica la condivisione di idee, suggerimenti e conoscenze, la cui raccolta consente di migliorare il concetto originario. Questo processo non può essere realizzato senza la cooperazione della comunità del software libero.

D'altra parte, la motivazione che spinge i membri della comunità a partecipare attivamente al progetto si fonda principalmente sullo sviluppo intellettuale, sulla condivisione delle conoscenze, sull'interesse per l'applicazione, sulla filosofia che sta dietro al progetto o al software libero in generale e sulla reputazione pubblica, così come sugli obblighi interni alla comunità.

### Struttura e dimensione della comunità

La comunità degli utenti di un'applicazione basata sul software libero può trovare la sua struttura più appropriata attraverso diverse modalità, che si differenziano in base alle azioni ed alle decisioni dei promotori del progetto, così come in base alle caratteristiche dell'applicazione e della sua produzione.

In generale, si può dire che la comunità di utenti di un'applicazione è in salute se presenta una struttura gerarchica funzionale adeguata agli obiettivi, attorno ad un nucleo attivo di sviluppatori.

A grandi linee, possiamo classificare le seguenti tipologie di membri all'interno di un progetto:

#### Web consigliato

K. Crowston; J. Howison (2006). "Assessing the health of a FLOSS Community" ([http://floss.syr.edu/publications/Crowston2006/Assessing\\_the\\_health\\_of\\_open\\_source\\_communities.pdf](http://floss.syr.edu/publications/Crowston2006/Assessing_the_health_of_open_source_communities.pdf))

#### Struttura gerarchica

Questo concetto può essere paragonato alla struttura di una cipolla (*onion-shaped* in inglese): al centro si mettono i membri più attivi e collegati al progetto, e nella parte più esterna quelli che partecipano di meno.

- Sviluppatori del nucleo dell'applicazione, con diritti di scrittura sul contenitore e con una storia pregressa di contributivi significativi al progetto.
- Leader del progetto, che forniscano le motivazioni e portino il progetto e la sua comunità ad un livello di maturità e di stabilità.
- Sviluppatori in generale, che contribuiscono al codice ma non hanno diritti di scrittura sul contenitore. Sono spesso incaricati di compiti di revisione.
- Utenti attivi che provano l'applicazione, segnalano errori, forniscono una documentazione ed agiscono da tessuto connettivo tra il progetto e gli utenti passivi (oltre ad altri compiti).

**Nota**

Questa prima classificazione di tipologie non è una struttura chiusa, dato che ogni progetto la adatta alle sue particolari caratteristiche.

## Processi di sviluppo

Il processo di sviluppo del software libero risulta essere spesso poco strutturato all'interno dei progetti, e questo è dovuto basicamente all'assenza di un piano di orientamento, all'imprecisa assegnazione dei compiti specifici ed alla mancanza di un ordine di priorità nelle funzioni dell'applicazione.

L'organizzazione del progetto è una caratteristica rilevante per il funzionamento ed il coordinamento della produzione, anche se un certo grado di duplicazione degli sforzi può essere visto come un sintomo positivo circa la relazione ed il coinvolgimento della comunità nel progetto.

Alla stessa maniera, il ciclo di valutazione e di posteriore accettazione dei contributi al progetto da parte dei membri della comunità permette di disegnare un quadro preciso sulla salute del suo funzionamento. Ad esempio, il rifiuto di un contributo può essere rappresentativo di una visione coesa e qualitativa del progetto a lungo termine.

## 2.3. Gestione della qualità

In alcune occasioni, la qualità del software libero è stato oggetto di dibattito tra promotori e detrattori. Nel corso di esso viene posto l'accento su aspetti come l'apertura del modello di sviluppo o la formazione dei collaboratori che contribuiscono al progetto, ad esempio.

Come in qualsiasi progetto di software, la produzione di software libero deve porsi delle regole di controllo di qualità nel corso del suo ciclo di vita. La qualità, cioè, deve poter essere misurata e confrontata con i valori attesi in ogni tappa della produzione del software e del suo utilizzo, così come da ogni punto di osservazione (promotore, utente o comunità).

In questo senso, anche se l'apertura ed il decentramento del modello di sviluppo del software libero favoriscono i meccanismi di controllo e di gestione della qualità, non costituiscono altresì una soluzione di per se stesse, ed è importante sottolinearlo perché non si deve smettere di cercare nuove soluzioni di verifica della qualità.

Per sviluppare gli aspetti relativi alla qualità della produzione del software libero, sfrutteremo l'articolo 'Managing Quality in Open Source Software' di Dhruv Mohindra, che realizza uno studio attento sul controllo di qualità negli aspetti riconducibili al software libero. Nei prossimi paragrafi vedremo i principali concetti offerti dall'autore.

### La qualità nel software libero

In generale, la qualità di una soluzione di software può essere valutata tanto in base alla sua architettura o al suo disegno interno, quanto in base alle funzioni che fornisce all'utente.

Le caratteristiche di apertura e decentramento proprie del modello di sviluppo del software libero costituiscono una infrastruttura sulla quale si possono poggiare politiche di gestione della qualità basate sull'individuazione e sulla risoluzione di alcune problematiche (tra vari aspetti). Pur essendo vero questo, a volte la mancanza di chiarezza e/o di una corretta strutturazione dei processi di produzione può generare risultati che non sono quelli attesi.

### Misurazione della qualità

Esistono diversi modi per valutare la qualità funzionale di un'applicazione. La funzionalità di metodologie quantitative dipende in gran parte dalla tipologia stessa del software, e pertanto vanno scelte in funzione delle caratteristiche e degli obiettivi dell'applicazione.

Quanto al discorso relativo alla qualità 'non quantificabile', bisogna porre in risalto il ruolo che riveste la comunità del software libero. Da un lato i test che realizza il team sulla qualità, dall'altro l'attività propria degli utenti dell'applicazione, che segnalano l'evidenza di errori di funzionamento o di elementi migliorabili del prodotto.

#### Web consigliato

D. Mohindra (2008). "Managing Quality in Open Source Software"

([http://www1.webng.com/dhruv/material/managing\\_quality\\_in\\_oo.pdf](http://www1.webng.com/dhruv/material/managing_quality_in_oo.pdf)).

In quest'ottica, lo stesso funzionamento distribuito e decentralizzato della comunità di utenti è importante per dare più garanzie di qualità al processo di produzione.

### **Controllo e revisione**

Un fattore importante per la qualità del prodotto finale è il controllo, abbinato alla revisione, di tutto il processo di sviluppo. In generale, i progetti di produzione di software libero utilizzano sistemi di controllo delle versioni per supportare con efficacia ed efficienza l'evoluzione delle diverse componenti del progetto.

Ci sono più maniere di organizzare il controllo e la revisione dell'evoluzione del software, così come dei rami di sviluppo e del contenitore (tra gli altri). Ad ogni modo conviene adattare la metodologia della produzione ed i sistemi di controllo e revisione dell'evoluzione alle peculiarità del progetto e del prodotto che si sta realizzando.

### **Leggende sul software libero**

Nonostante la sua costante diffusione, il software libero ancora oggi porta con sé alcuni miti -tanto positivi quanto negativi- che possono influire sulla considerazione che gli viene data.

Questi miti non hanno nessuna base salda attraverso la quale si possa ideare ed implementare una gestione coerente e sostenibile della qualità, e pertanto si deve analizzare ognuno di essi singolarmente.

A continuazione elenchiamo alcune leggende abituali legate alla qualità del software libero.

- Il fatto che il codice fonte sia pubblico non è garanzia di sicurezza e qualità, dato che in ogni caso dipende dall'interesse e dalla revisione della comunità.
- Il congelamento delle funzioni non aumenta la stabilità dell'applicazione di per se stessa, dal momento che la cosa importante è che il codice sia scritto bene fin dall'inizio.
- Il miglior modo di comprendere un progetto non è quello di correggere i suoi eventuali errori, dato che la documentazione è di gran lunga migliore per questo obiettivo.
- In generale, gli utenti non dispongono dell'ultima versione del contenitore con le correzioni degli errori aggiornate ogni giorno.

A grandi linee, i processi di verifica e di revisione, così come le discussioni pubbliche e la cultura *hacker* proprie della comunità di utenti, devono essere accompagnati da una pianificazione e da una gestione attiva della qualità della produzione.

Questa gestione deve essere diretta a coprire eventuali lacune in uno o più aspetti del prodotto, ad esempio la pianificazione della produzione, lo sviluppo delle funzioni o la documentazione dell'applicazione.

### Considerazioni aggiuntive rispetto alla qualità

In generale, tanto la pubblicazione del codice fonte quanto l'inclusione di sistemi di gestione dell'errore, ed il fatto di ripartire le responsabilità relative al prodotto tra tutti coloro che vi sono implicati, sono aspetti chiave per la gestione della qualità.

In questo senso, è molto importante per la qualità generale del progetto considerare anche la trasparenza intrinseca in tutte le azioni, fidarsi del team di sviluppo, verificare e testare tutte le parti del codice fonte, e promuovere tanto la filosofia quanto l'importanza di farlo bene fin dall'inizio.

## 2.4. Legalità e contributi

In un progetto basato sul software libero con la partecipazione della comunità degli utenti, assume particolare rilevanza la gestione legale dei contributi di ogni membro coinvolto.

Questa gestione è importante per i promotori del progetto tanto quanto per i membri della comunità, dato che maneggia le caratteristiche di paternità e di titolarità dei diritti del codice. La sua importanza è data anche dalle implicazioni che può creare la combinazione di codici di differenti autori in uno stesso prodotto.

Per sviluppare questi concetti prenderemo spunto dalla lettura della sezione 2.4 'Autori e titolari del diritto', presa dai materiali didattici del corso *Aspetti legali e dello sfruttamento del software libero*.

### L'autore

L'autore di un'opera è la persona fisica o giuridica che realizza l'opera, e questo implica inevitabilmente che gli appartiene la paternità della sua creazione originale.

#### Web consigliato

M. Bain ed altri (2007). *Aspectos legales y de explotación del software libre*. Universitat Oberta de Catalunya (<http://ocw.uoc.edu/informatica-tecnologia-i-multimedia/aspectes-legals-i-dexplotacio/materials/>).

Le opere realizzate da più persone si collocano in diverse fattispecie:

- Un'opera in collaborazione è il risultato unitario di una composizione di diverse parti che possono essere utilizzate e sfruttate indipendentemente.
- Un'opera collettiva è l'unione di vari contributi che non si possono utilizzare e sfruttare in modo indipendente.
- In un'opera realizzata su commissione (o con contropartite economiche), la paternità ricade sulla persona fisica o giuridica che realizza il prodotto.

Nel mondo del software libero, la paternità dipende in larga misura dalle considerazioni di cui sopra, tenendo sempre a mente che a volte il trasferimento della paternità può risultare utile e pratico.

D'altro canto, le condizioni attraverso le quali si realizzano le opere derivate (preesistenza del contenuto) possono cambiare anche sostanzialmente a causa dell'autore o dell'opera stessa. In ogni caso, le licenze libere devono specificare le condizioni di derivazione e di redistribuzione delle opere.

### **Il titolare originario e il titolare derivato**

Il titolare originario dell'opera è sempre il suo autore. Nonostante questo, alcuni diritti sull'opera possono essere ceduti ad altre persone, fisiche o giuridiche.

In questo caso, la persona che riceve la cessione di parte dei diritti relativi ad un'opera diventa il titolare derivato della stessa. È necessario sottolineare che solo il titolare di un diritto concreto può concedere licenze sul suo diritto.

### **Identificazione del titolare**

Per poter esercitare i diritti di cui sopra deve essere possibile identificare l'autore di ogni opera. Questo può essere complicato nel mondo del software libero, dal momento che coloro che contribuiscono al progetto possono essere molteplici.

Per attenuare questo tipo di problemi, i progetti basati sul software libero conservano sempre le liste di coloro che vi hanno preso parte. A volte questi progetti richiedono la cessione dell'intero pacchetto, o solo di alcuni dei diritti, prima di accettarne il contributo.



### 3. Caso di studio

Nelle sezioni precedenti sono stati esaminati tanto il progetto di software libero quanto la gestione della comunità degli utenti. Entrambe le sezioni riportano i principali aspetti legati alla produzione del software libero dal punto di vista della gestione del progetto.

Per concludere questo modulo, dedicheremo questo ultimo capitolo a rendere più concreta una buona fetta delle idee e delle proposte presentate, attraverso lo studio di un caso concreto di un'impresa basata sul software libero.

Le prossime parti hanno l'obiettivo di fornire una guida per rendere più chiaro come un'impresa basata sulla produzione di software libero implementa la sua propria metodologia, crea e gestisce la relazione con la comunità degli utenti, e come affronta molte delle decisioni che deve prendere con il passare del tempo.

Ci concentreremo sul caso della Openbravo, S.L.

#### 3.1. L'impresa

Openbravo, S.L. è un'impresa che si dedica a sviluppare soluzioni professionali basate sul software libero per le imprese.

##### Modello di business

Il modello di business che utilizza l'impresa si basa sulla prestazione di servizi relativi ai prodotti che sviluppa. Come è già stato detto, la sua strategia di business è basata sull'associazione e sulla collaborazione tra imprese per sfruttare la stessa possibilità di business.

##### Strategia dell'impresa

Il modello di business viene portato avanti mediante dei *partner* che offrono servizi ai clienti finali (ad esempio la personalizzazione ed il supporto). In un certo qual modo, questo particolare tipo di gerarchia tra produttore, distributore (o *partner*) e cliente stabilisce un'atmosfera di cooperazione con obiettivi comuni.

#### Nota

Tutta l'informazione presentata in questa sezione è stata tratta dal suo sito web (<http://www.openbravo.com/>).

Per portare a termine la sua strategia, l'impresa pubblica un manifesto come fosse una dichiarazione di intenti, che include tanto aspetti legati al software libero (come la trasparenza, l'apertura e la collaborazione), quanto gli obblighi dell'impresa contratti rispetto a terzi (come l'accesso libero e la gestione dei contributi).

## Gestione e direzione

La gestione dell'impresa combina compiti interni ed esterni all'organizzazione, tanto nella squadra di direzione quanto nel Consiglio di Amministrazione, prodotto dell'investimento esterno che ha ricevuto l'impresa, così come della speciale metodologia basata sul software libero.

### 3.2. I prodotti

Openbravo realizza due soluzioni basate sul software libero, che possono funzionare l'una indipendentemente dall'altra o in combinazione. Entrambi i prodotti sono distribuiti con licenza libera e con download diretto da internet.

I prodotti che offre Openbravo sono i seguenti:

- **Openbravo ERP**

Openbravo ERP (*Enterprise Resource Planning*) è un sistema di gestione imprenditoriale in un ambiente web, che integra in una struttura organizzata a moduli diverse funzioni di gestione, quali gli approvvigionamenti, il magazzino, la produzione o la contabilità.

Il prodotto è dato con licenza MPL 1.1 e può funzionare in diversi ambienti ed in diversi sistemi di base dei dati, così come può integrarsi con Openbravo POS.

Tra le molte informazioni sul prodotto che vengono fornite spicca la mappa del piano di sviluppo del progetto.

- **Openbravo POS**

Openbravo POS (*Point Of Sales*) è un sistema di terminali di punti di vendita che può essere integrato con Openbravo ERP.

Il prodotto è dato con licenza GNU/GPL e può funzionare in differenti contesti e con diversi sistemi di base di dati. È realizzato in particolare per terminali tattili.

Tra le informazioni che vengono fornite, anche qui spicca la mappa del piano di sviluppo del progetto.

#### Web consigliato

Mozilla Public License 1.1  
(<http://www.mozilla.org/MPL/MPL-1.1.html>).

#### Caratteristiche principali di Openbravo ERP

<http://sourceforge.net/projects/openbravo/>.

#### Web consigliato

GNU General Public License  
(<http://www.gnu.org/licenses/gpl.html>).

#### Caratteristiche principali di Openbravo POS

<http://sourceforge.net/projects/openbravopos/>.

### 3.3. La comunità degli utenti

La comunità degli utenti del software libero riveste un ruolo rilevante nella strategia imprenditoriale di Openbravo. Qui sotto ne menzioneremo gli aspetti principali.

#### Strategia *Open Source*

Per identificare la strategia *Open Source* di Openbravo dobbiamo tenere a mente le peculiarità della metodologia di sviluppo dei prodotti e la struttura di business che utilizzano.

Il nucleo di entrambi i prodotti viene sviluppato per lo più internamente all'impresa, mantenendo però contenitori pubblici ed una comunità di utenti attiva al suo fianco. D'altra parte, nello sviluppo dei complementi, delle personalizzazioni e delle estensioni rispetto al prodotto originale entrano in gioco tanto la comunità degli utenti quanto i *partner*.

Questo ultimo caso va analizzato separatamente, dal momento che corrisponde all'utilizzo di un'opportunità da parte di un'organizzazione differente.

Detto questo, Openbravo realizza una strategia *Open Source* che combina diversi orientamenti:

- Per lo sviluppo e la revisione dei prodotti, la strategia si avvicina all'*Open Monarchy*, principalmente a causa dello sviluppo interno del nucleo dei prodotti, dei contenitori pubblici del codice fonte, dell'approvazione finale dei cambi relativi al nucleo a carico dell'impresa e della progettazione dello sviluppo dei progetti (ad esempio, le direzioni intraprese).
- Per lo sviluppo dei complementi (ad esempio, la documentazione), la strategia è simile al *Consensus-based development*, visto che lo sviluppo avviene all'interno di una comunità di utenti.
- Infine, la strategia per lo sviluppo delle estensioni e delle personalizzazioni dipende dallo sviluppatore che la pianifica. Se sono progetti realizzati in seno alla comunità (attraverso le risorse offerte da Openbravo), si avvicinano più propriamente al modello del *Consensus-based development*, mentre se sono sviluppate dai *partner* dipenderanno tanto dalla loro particolare strategia quanto dalle caratteristiche dello sviluppo.

#### Strategia del *partner*

Nel caso in cui il *partner* sviluppi delle estensioni del prodotto originale, la strategia *Open Source* dipenderà tanto dalla sua filosofia imprenditoriale quanto dalle caratteristiche del prodotto (ad esempio, la licenza MPL è più flessibile con i moduli proprietari rispetto alla GPL).

## Struttura della comunità

La comunità degli utenti di Openbravo ERP è definita e strutturata secondo un sistema meritocratico: esistono vari livelli di collaborazione ed ognuno di essi si definisce a partire dalle conoscenze necessarie per il livello, dalla quantità di contributi, dalle responsabilità e dai privilegi.

Nel caso di Openbravo ERP esistono tre diversi profili di collaborazione (sviluppatori, esperti funzionali e tester), mentre nel caso di Openbravo POS esiste solo il profilo dello sviluppatore. I membri della comunità di utenti si organizzano e si distribuiscono a seconda dei progetti attivi nella comunità.

## Risorse a disposizione della comunità

Openbravo dispone di diverse risorse (alcune della quali in più di una lingua) sia per la comunità sia per i *partner* e per gli utenti in generale, tra le quali troviamo:

- Web corporativa
- Area dei *partner*
- Wiki del progetto
- Portale della comunità degli utenti di Openbravo
- *Blog* dei lavoratori
- Modellazione dei prodotti (Openbravo ERP e Openbravo POS)
- *Bug tracker*
- Università
- Liste di distribuzione della posta
- Contenitore del codice di Openbravo
- Servizio di notizie di Openbravo

In generale, la comunità degli utenti ha a disposizione una guida specifica disponibile nel wiki che descrive come collaborare al progetto. Dispone anche di una lista esauriente dei canali di comunicazione ai quali può accedere. In più, le strategie di orientamento di ogni prodotto sono l'ultima componente presente nella guida alle risorse per la comunità degli utenti.

### Web consigliato

Dal portale web dell'impresa si può accedere a tutte le risorse menzionate (<http://www.openbravo.com/>).

### 3.4. Posizionamento ed evoluzione

L'impresa è nata nel 2001 con il nome Tecnicia. Nel 2006 ha ottenuto dei finanziamenti per più di sei milioni di dollari e ha cambiato nome, passando a Openbravo. Nello stesso anno ha liberato il codice fonte dei prodotti che sviluppa con licenze libere.

Nel maggio del 2008, i flussi di finanziamenti sono saliti a più di dodici milioni di dollari, e tra i suoi investitori appaiono Sodena, GIMV, Adara e Amadeus Capital Partners.

Negli anni, Openbravo ha ricevuto diversi premi legati al mondo dell'impresa e del software libero, e allo stesso modo ha ricevuto sovvenzioni nell'ambito del programma di impulso alla innovazione tecnologica (PROFIT) da parte del Ministero spagnolo per l'Industria, il Turismo ed il Commercio.

Sia l'impresa sia la comunità degli utenti hanno una tendenza evolutiva positiva, se pensiamo che il progetto si situa attualmente tra i venticinque più attivi di SourceForge, con più di un milione di download accumulati agli inizi del 2009.

#### Web consigliato

Sui progetti più attivi di SourceForge:

<http://sourceforge.net/top/mostactive.php?type=week>.

## Riepilogo

In questo modulo abbiamo esposto le principali caratteristiche legate alla creazione, alla gestione ed al mantenimento del progetto di sviluppo del software libero, facendo particolare attenzione alla partecipazione della comunità degli utenti.

In una certa maniera, i fondamenti della produzione di software libero non sono troppo diversi dalle metodologie di sviluppo dei software più tradizionali. Nonostante questo, le peculiarità dell'apertura del codice e dell'esistenza di una comunità di utenti ne modellano il funzionamento e lo rendono particolare sotto molti punti di vista.

Per quello che riguarda il progetto in sé, bisogna sottolineare l'importanza che rivestono l'identificare, definire e strutturare gli aspetti funzionali del progetto (ad esempio, l'infrastruttura, la gestione delle versioni o le misure di coordinamento) e quelli legati al software libero (ad esempio, la credibilità, la trasparenza o le tipologie di partecipazione).

A questi aspetti vanno sommati i fattori legati alla comunità del software libero, come la strategia di gestione della comunità da parte dell'impresa (strategia *Open Source*), la metodologia ed il ciclo di vita del prodotto, la gestione della qualità e gli aspetti legali relativi ai contributi apportati dagli utenti.

Da ultimo, abbiamo presentato un caso di studio rappresentativo di una buona parte degli aspetti esaminati nel corso del modulo.

## Bibliografia

**Bain, M. y otros** (2007). *Aspectes legals i d'explotació del programari lliure*. Universitat Oberta de Catalunya <[http://ocw.uoc.edu/informatica-tecnologia-i-multimedia/aspectes-legals-i-dexplotacio/Course\\_listing](http://ocw.uoc.edu/informatica-tecnologia-i-multimedia/aspectes-legals-i-dexplotacio/Course_listing)> [Consultazione: febbraio 2009].

**Collins-Sussman, B.; Fitzpatrick B.** (2007). *What's in it for me? How your company can benefit from open sourcing code*. OSCON: 27 luglio 2007 <<http://www.youtube.com/watch?v=ZtYJoatnHb8>> e diapositive <<http://www.red-bean.com/fitz/presentations/2007-07-27-OSCON-whats-in-it-for-me.pdf>> [Consultazione: febbraio 2009].

**Crowston, K.; Howison, J.** (mayo, 2006). *Assessing the Health of a FLOSS Community*. IT Systems perspectives (pág. 113-115). <[http://floss.syr.edu/publications/Crowston2006Assessing\\_the\\_health\\_of\\_open\\_source\\_communities.pdf](http://floss.syr.edu/publications/Crowston2006Assessing_the_health_of_open_source_communities.pdf)> [Consultazione: febbraio 2009].

**Fogel, K.** (2005). *Producing Open Source Software: How to Run a Successful Free Software Project*. <<http://producingoss.com>> [Consultazione: febbraio 2009].

**Mako, B.** (2001). *Free Software Project Management HOW TO*. <<http://mako.cc/projects/how-to>> [Consultazione: febbraio 2009].

**Mohindra, D.** (2008). *Managing Quality in Open Source Software*. <[http://www1.webng.com/dhruv/material/managing\\_quality\\_in\\_oo.pdf](http://www1.webng.com/dhruv/material/managing_quality_in_oo.pdf)> [Consultazione: febbraio 2009].

**Openbravo** <<http://www.openbravo.com/>> [Consultazione: febbraio 2009].

**Raymod, E.** (1997). *The cathedral and the bazaar* <<http://www.catb.org/~esr/writings/cathedral-bazaar/>> [Consultazione: febbraio 2009].

**Tawileh, A. ed altri** (agosto 2006). *Managing Quality in the Free and Open Source Software Community* (pags. 4-6). Proceedings of the Twelfth Americas Conference on Information Systems. México: Acapulco. <<http://www.tawileh.net/anas/files/downloads/papers/FOSS-QA.pdf?download>> [Consultazione: febbraio 2009].

