Open source, software libero e altre libertà

Carlo Piana

Indice

Introduzione

a Paola e Livia, le mie involontarie cavie intellettuali

Libri sull'open source, software libero, libertà e vari dintorni non mancano, in italiano e in inglese (e probabilmente in cento altre lingue a me più o meno sconosciute!). Simone Aliprandi, col quale ho condiviso alcuni sforzi letterari e divulgativi (con meno successo del summenzionato) dal quale ho saccheggiato parte dei contenuti (soprattutto nel capitolo sulle Creative Commons), si è speso in lungo e in largo. Altri contributi sono reperibili singolarmente e collettivamente; ad alcuni ho persino dato una mano. Giovanni Ziccardi, Andrea Rossetti, Giovanni Battista Gallus, Francesco Paolo Micozzi, Marco Ciurcina, Marco Ricolfi, Vincenzo Zeno Zeno-Zencovich Luigi Carlo Ubertazzi, Marco Bertani, Flavia Marzano e molti altri (che mi perdoneranno per non averli menzionati tutti) hanno già più che adeguatamente battuto il terreno in Italia. Nel mondo molti altri, che sarebbe troppo lungo qui ricordare, hanno fatto ampia divulgazione, cito solo alcuni coi quali ho la fortuna di interagire di frequente, quali Eben Moglen (**Il** maestro!), Larry Rosen; Heather Meeker, Pamela Jones con Groklaw, Mark Webbink, e l'immanente Richard Stallman (RMS) hanno più che ampiamente asfaltato un'autostrada su cui modestamente e lentamente mi sono incamminato. E sarebbe impossibile discutere di libertà di software e contenuti senza il contributo essenziale di Lawrence Lessig.

Non è un'area del diritto scarsamente presidiata.

Un libro introduttivo sui concetti di apertura delle licenze per arrivare all'apertura di contenuti – e di qualsiasi cosa sia ristretta da un diritto di esclusiva su beni immateriali –, scritto in modo semplice, immediato e divulgativo, ma sufficientemente rigoroso, sistematico e coerente, in Italiano sull'openness nel diritto, però, come avrei voluto leggere io vent'anni or sono, non c'è.

Quando la redazione di Tech Economy mi ha chiesto di scrivere in materia di software libero sulla rivista online, mi è parso evidente che si poteva colmare la lacuna in maniera efficiente. Ho iniziato pertanto a scrivere alcuni pezzi avendo in mente di raccoglierli definitivamente, con pochi riadattamenti, in un libercolo in cui anche chi è particolarmente digiuno dei concetti di base possa trovare un concreto avviamento alla materia. Ho testato dunque il capitolo introduttivo

(che è stato anche la prima puntata della mia serie di articoli[1] Tech Economy) con la mia famiglia, e apparentemente dopo molti anni hanno iniziato a capire ciò di cui parlavo.

Se anche solo due persone, a me particolarmente care, hanno avuto modo di capirci qualcosa, be', allora sono soddisfatto.

In questa opera non c'è tutto. Non ho parlato di *open hardware*, per esempio, né di *open science*, argomenti che mi affascinano e che in certo modo ho anche affrontato. Ma ho cercato di trovare i campi di analisi più rilevanti per una (parolona!) "**teoria generale**" del diritto dell'openness. Benché in maniera – lo ripeto – divulgativa e non certamente con un'analisi approfondita (rimando ai numerosi contributi nei vari volumi dell'International Free and Open Source Software Law Review[2] per chi volesse approfondire in maniera più articolata) Questo volevo realizzare, dare alcuni fondamenti teorici comuni e spunti di ragionamento a chi volesse raccogliere il testimone e dare un quadro più completo e definitivo.

Per il momento, sono soddisfatto del risultato raggiunto. Alle stampe!

[1] http://www.techeconomy.it/author/carlo-piana/
[2] http://ifosslr.org "IFOSSLR"

Capitolo 1

Brevi cenni sull'universo (aperto)

Un detto piuttosto famoso recita più o meno "chi ha come unico strumento un martello, tende a vedere qualsiasi problema come un chiodo". Sembrerebbe questa la ragione per cui un giurista tenda a vedere qualsiasi tema come un tema giuridico. Mi dichiaro allora colpevole del reato ascritto: vedo l'"open" dal punto di vista giuridico. In questo capitolo discuterò dell'*openness*, apertura, in tecnologia, in senso generale, per fornire una piccola teoria generale che possa funzionare come guida intellettuale per tutti i capitoli successivi. Mi occuperò in seguito di tutte le sfaccettature dell'open, dall'*open source / software libero* all'*open content*, dagli *open data* agli *open standard*, dall'*open hardware* all'*open whatever*. Questo capitolo è dunque da leggere obbligatoriamente per capire – quale *fil rouge* – tutto il resto della discussione.

Ma **cos'è "open"**?

Un altro detto famoso recita "non so cos'è la pornografia, ma la riconosco quando la vedo". Molti pensano di sapere cosa sia l'open-qualcosa soltanto guardandolo. Ma, pur essendoci qualche affinità di concetti tra la pornografia e i *diversi* tipi di apertura di cui ci occupiamo, le cose non funzionano così. Per riconoscere cosa è "open" occorre saper riconoscere cosa è "chiuso" e dunque sapere in che modo qualcosa di immateriale possa essere chiuso. Guarda caso, e qui torniamo al punto di partenza, la gran parte dei modi in cui ciò di cui ci occupiamo viene "chiuso" è una norma giuridica, si chiami esso "copyright", "brevetto", "diritto *sui generis*", "segreto". Pertanto, stabilire se quel qualcosa sia aperto, significa stabilire se ricada, o non ricada, in – o sia sufficientemente libero da – vincoli giuridici che ne rendono difficoltoso o impossibile l'uso, la replicazione, la modifica, la diffusione secondo i dettami dell'*openness*.

Ecco dimostrato: è una questione giuridica.

1.1 *Ecce homo restrictivus*!

Diciamo la verità, l'uomo tende a rendersi comoda la vita. Non che questa sia una tendenza irragionevole o nefasta, intendiamoci. L'innovazione tecnologica e il progresso nascono da questa spinta. A volte però l'uomo tende a usare qualche espediente per ritrovarsi in una **situazione protetta**. Fino a non molto tempo fa, diciamo fino al nascere della borghesia e alla rivoluzione industriale, l'ambito della protezione promanava da chi deteneva il potere, fosse esso politico o religioso. Alcuni ambiti della vita e dell'iniziativa economica erano strettamente riservati al titolare del potere, il quale aveva pertanto la facoltà di concedere graziosamente (= per grazia) diritti e guarentigie. Risale a questo periodo il concetto di "patente", come "permesso", da cui deriverà il termine "*patent*", "brevetto". Inclusa la nefanda "patente da corsa", che non abilitava a guidare veicoli sportivi in circuiti motoristici, ma a depredare altre navi in nome del Re.

Nelle scienze liberali e nella tecnica, invece, vigeva, da un punto di vista dell'iniziativa economica, una larga libertà. Potere temporale e potere religioso si affiancavano – è vero – nel controllare la libera espressione del pensiero, e anche nella scienza, il professare certe teorie non portava alcunché di buono. Tuttavia, nei rapporti tra "pari" non vi era alcun limite nell'utilizzo di beni intellettuali altrui. Anzi, **il concetto di** "**bene intellettuale**" non esisteva, non esisteva dunque neanche il concetto di "bene intellettuale altrui". Addirittura, nel sistema delle botteghe, neppure il diritto morale di essere riconosciuto autore esisteva, le opere erano naturalmente collettive. Se poi uno voleva dipingere un San Sebastiano o una Deposizione dalla Croce, lo faceva prendendo a modello, molto spesso sostanzialmente copiando, pezzi di lavoro di qualcun altro. Se uno voleva creare un tessuto identico a quello visto in un viaggio nelle Fiandre, lo poteva fare, se ne era capace. Il rinascimento nacque così, senza **copyright**, senza brevetti, in parte senza marchi, senza design, senza diritti ***sui generis***, senza modelli di utilità, eccetera eccetera.

È solo con la l'invenzione e la diffusione della stampa a caratteri mobili che si affronta per la prima volta il problema derivante dal fatto che un rilevante investimento come quello effettuato per l'edizione di un libro, possa essere appropriato da qualcuno che si limiti a copiarlo, con poca spesa. Con la riduzione dei costi di stampa, si rende per la prima volta palese il valore "intellettuale" di creazione dell'opera rispetto al costo materiale di produrre il singolo esemplare. Prima, hai voglia copiare Leonardo o Raffaello dipingendo come Leonardo e Raffaello! Ecco che nasce il "*copyright*", con lo *Statute of Anne* (1709-10), che mira a proteggere il diritto **dello stampatore** sulla possibilità di trarre copie dei libri (ecco perché copy-right).

Ma per le altre opere dell'intelletto, come ci si faceva a proteggere? Se si inventava il modo di rendere la terracotta più leggera, più bella, più resistente, più bianca, come fare a non consentire ad altri di produrla "rubando" l'idea? Semplice: si teneva la cosa **segreta** (è avvenuto con la ceramica, con la seta, ad esempio). Ma non per ogni cosa tenere il segreto era possibile, alcune invenzioni

erano auto-evidenti a partire dal prodotto in sé. Se l'efficienza di una macchina a vapore veniva aumentata facendo passare nella caldaia tubi dell'aria calda, era sufficiente comprare una locomotiva, smontarla e vedere come funzionava (oggi si chiamerebbe "*reverse engineering*"). Inoltre, tenere segreta una tecnologia non consente ad altri di poterla migliorare: da un punto di vista dell'economia generale affidarsi al segreto non è efficiente per l'innovazione tecnologica. Ecco che intorno alla seconda metà del sedicesimo secolo inizia a fare la sua apparizione in maniera significativa una diversa protezione, quella delle invenzioni industriali, con i **brevetti**. Un brevetto consente una privativa sullo sfruttamento di una determinata invenzione, a patto di averla completamente descritta, e solo per un limitato periodo di **tempo**, dopo di che entra nel "**pubblico dominio**". La teoria classica prevede che ciò crei un incentivo a intensificare la ricerca e lo sviluppo della tecnica, incentivo costituito dalla promessa di ottenere un limitato monopolio, rendendo un bene scarso ciò che è naturalmente un bene illimitatamente disponibile, ovvero un'idea resa pubblica. È infatti un monopolio la privativa che queste leggi creano, dando a un soggetto il diritto arbitrario di decidere come sfruttare (con alcuni limiti) le proprie creazioni. Diritti che possono essere trasferiti a titolo originario o derivato (licenze).

E ciò sembrò talmente una cosa buona, che il sistema è pervenuto a noi sostanzialmente invariato sino ad oggi. A nessuno pare vero di poter creare, con un colpo di penna, valori economici dal nulla: i "beni intellettuali". Da allora ci si è mossi solo nella direzione di allargare i termini di tutela, introducendo nuovi diritti e nuove privative, allungando i termini di protezione – anche al di là di ciò che sarebbe naturalmente un "incentivo" –, rendendo in alcuni casi automatico ottenere e mantenere il diritto (come nel caso del copyright), **limitando** sempre di più lo spazio di ciò che è **libero** e non soggetto ad altrui diritti, fino alla situazione odierna in cui in pratica vi sono ambiti dove si è sostanzialmente privi di un pubblico dominio significativo (come nella letteratura, nel cinema, nella tecnologia dell'informazione).

1.2 I benefici dei *commons* e la tragedia degli *anticommons*

Il pensiero comune si è nel frattempo conformato allo stato dell'arte, tanto da far ritenere naturale pensare ai "beni intellettuali" come una proprietà, allo stesso modo di una sedia o di un terreno. Da qui il concetto di "proprietà intellettuale", che ha dato origine a teorie giusnaturaliste circa la stessa, quasi come se essa sia un diritto universale dell'umanità, e non una creazione del diritto che non è esistita se non nell'ultima parte della storia umana.

Segue l'**iperfetazione** attuale **di protezioni** e la moria degli usi liberi a cui assistiamo oggi. Non sazi, inventiamo protezioni ogni giorno. Sembra che se un

ambito non sia oggetto di almeno tre o quattro protezioni giuridiche contemporaneamente, sia un reietto della società.

Ma ogni situazione di protezione deve trovare un limite, o si scade nell'arbitrio, nella sopraffazione. Siamo al punto di non poter oggi girare un film senza dover chiedere almeno un centinaio di permessi, per inquadrare un ponte, un edificio, un poster, un prodotto industriale. Di non poter creare un prodotto di un qualsiasi tipo senza dover prendere in licenza centinaia, migliaia, decine di migliaia di brevetti, diventati in molti campi un freno anziché un incentivo all'innovazione.

La situazione è quella riassunta molto bene da Lawrence Lessig col termine "tragedia degli *anticommons*", ribaltando un'espressione resa popolare dal titolo di un lavoro di Garret Hardin "La tragedia dei *commons*" in cui descriveva come il libero (meglio: sregolato) utilizzo di risorse comuni porta alla distruzione delle risorse e all'impoverimento degli utilizzatori. Ma Hardin descriveva i beni fisici, non i beni intellettuali, dove, anzi, la creazione di un "common" consente la moltiplicazione del valore di quel bene.

L'openness, nell'accezione principale, è appunto questo: far rientrare quanto più possibile in un terreno di libero accesso, di libero utilizzo, di libera modifica, di libera redistribuzione, ciò che sarebbe invece ristretto e proprietario. Creare, in ultima analisi, dei *commons*.

1.3 Gli strumenti dei *commons* sono strumenti legali

Il principale strumento con cui si creano i *commons* sono di tipo legale. Paradossalmente, è diventato oggi difficile, in certi casi addirittura macchinoso, far sì che un bene intellettuale sia liberamente utilizzabile. In certi ambiti e in molte legislazioni ciò è addirittura in teoria impossibile, in altri casi è estremamente difficile e in molti di più richiede uno sforzo positivo notevole.

Regna sovrana in tale ambito la licenza, che è lo strumento giuridico con il quale chi è titolare di diritti di privativa ne concede ad altri il permesso (*licet* = è consentito) di farne ciò che altrimenti sarebbe riservato. Il primo, e in molti casi l'unico, modo per giudicare se qualcosa sia "open" è infatti verificare se chi ne ha i diritti abbia apposto una licenza che consenta agli altri di usare liberamente l'oggetto del rilascio. È una licenza particolare. È gratuita, mentre "tradizionalmente" la licenza è a pagamento. È pubblica, mentre tradizionalmente è individuale. È estensibile da chiunque la riceva dal primo percettore verso chiunque altro, mentre tradizionalmente è intrasmissibile o è trasmissibile solo a patto che chi la cede se ne spogli.

Semplice vero?

Per nulla!

Il concetto di "open" va valutato sotto molti profili, e non sempre gli stessi principi valgono per tutti gli ambiti e per tutti i diritti di privativa. Se un bene immateriale è protetto dal copyright, è sufficiente ottenere il permesso da chi ne detiene i diritti per essere (ragionevolmente) sicuri di essere in un terreno libero. Se è in un ambito protetto da brevetti, la cosa si complica e si può solo escludere che un limitato o definito numero di brevetti possa essere utilizzato contro l'apertura di quel bene, ma non che non esista un brevetto che tale apertura possa prevenire. È come avere cento lucchetti su una porta: toglierne 99 non è sufficiente a entrare.

Altri diritti di privativa, invece, sono del tutto irrilevanti rispetto alla creazione di un common intellettuale. È il caso dei **marchi**, che ben difficilmente possono essere d'intralcio ai liberi utilizzi di un bene intellettuale (non che non vi sia un uso abominevole dei marchi, ma solitamente ogni abuso è diretto verso altri marchi o segni distintivi).

1.4 Il *copyleft*

Inoltre, le licenze sono strumenti legali che concedono diritti, ma lo fanno in molti casi con alcune condizioni. Condizioni diverse possono essere incompatibili tra loro, creando così una frammentazione dei commons, rendendoli reciprocamente non mescolabili tra loro, o mescolabili in una sola direzione. È questo il caso del cosiddetto "***copyleft***", concetto misconosciuto ai più e che è di fondamentale importanza per chi si occupi di openness. Il *copyleft*, in breve, consiste in una serie di condizioni poste dal titolare dei diritti come contraltare della licenza, per fare in modo che ciò che è stato rilasciato come libero rimanga (*be left*) libero. "*Left*" (inteso come "sinistra") è anche il contrario di "*right*" (inteso come "destra"), e dunque "*copyleft*" è anche uno sberleffo a "*copyright*", pur non essendone affatto il contrario (un non-copyright, come molti pensano), anzi, essendo l'uso del *copyright* per mantenere aperto ciò che in assenza potrebbe venire chiuso, *proprietarizzato*. *Copyleft* che ha senso – a mio modesto parere è una necessità – in ambiti che vedono la necessità di preservare "ereditariamente" le condizioni di libertà e apertura in tutte le opere che sfruttano il lavoro precedente (è il caso ovviamente del software, in larga misura dei contenuti creativi), mentre non ha affatto senso, anzi, potrebbe essere deleterio, in ambiti che non vedono il copyright, e dunque il concetto di "opera derivata" come elemento principale di protezione, ed è ad esempio il caso dell'open hardware, degli open standard, degli open data.

1.5 Prime conclusioni

Brevi cenni, avevo detto, e brevi sono stati, rispetto a quanto ci sarebbe da dire in tema. Vedremo nei prossimi capitoli come il concetto di openness si declina e quali sono le implicazioni. Per il momento, *repetita iuvant*, non è possibile parlare di openness senza considerare che si tratta di vincere vincoli giuridici tramite strumenti giuridici. Anche se, ovviamente, per produrre *commons* immateriali occorre ben più di una licenza!

Capitolo 2

Una breve storia personale del software libero

Abbiamo visto cosa significa "aperto", in generale e le dinamiche che prevengono l'apertura. Abbiamo anche visto le dinamiche che rimuovono le chiusure e fanno di ciò che sarebbe "chiuso" qualcosa di "aperto".

Ora è tempo di applicare le conoscenze appena conseguite. Iniziamo dal campo che ha dato per primo e con maggiore dovizia spunti e strumenti all'apertura dei prodotti intellettuali: il **software**. E cominciamo da come **io** sono giunto a conoscere il software libero. Genesi, capitolo primo.

2.1 In principio era l'ignoranza

Ricordo la prima volta che ne sentii parlare. In un trafiletto su qualche ormai dimenticata rivista si parlava di alcuni tizi, prevalentemente in ambito universitario (MIT) i quali sostenevano, anatema! che il software dovesse essere libero. E menzionava il nome di questa accozzaglia di matti: la Free Software Foundation.

Per me, giovane praticante avvocato cresciuto in uno studio che faceva della tutela del software uno dei fiori all'occhiello, era cosa inaudita, che ovviamente non avrebbe mai avuto nessun seguito, se non tra questi che sicuramente erano hippy dalla dubbia igiene.

Avevo, in poche parole, avuto un contatto non ravvicinato con il software libero, e se a voi piace, "software open source". Siccome la poca conoscenza è ben più pericolosa di una completa ignoranza, avevo tratto conclusioni che più sbagliate non potevano essere su quale potesse essere il futuro di quell'abominio. Come oggi, a più di vent'anni di distanza, il fatto che ne scriva su queste pagine mi porta a considerare.

2.2 E l'ignoranza si fece approssimata conoscenza: il misterioso "codice sorgente"

Il primo contatto vero con il software libero fu una distribuzione marchiata Red Hat di GNU/Linux, doveva essere il 1996 o giù di lì.

Flash forward: anno 2000, il Millennium Bug, sul quale il giovane aveva costruito un poco di notorietà, si era dimostrato una bolla di sapone. Delle migliaia di cause che si prospettavano a seguito del malfunzionamento del software con il passaggio al nuovo millennio, neppure una. Nel frattempo però avevo avuto modo di valutare come il fatto di non avere a disposizione il codice sorgente avrebbe potuto trasformare un evento tutto sommato banale – il fatto di aver utilizzato per definire l'anno con due cifre anziché con quattro – in una catastrofe profetizzata, tanto da apparire in numerosi film (nessuno memorabile, per la verità).

Non avere il codice sorgente fa sì che, anche se tu hai pagato profumatamente lo sviluppo del software, il software non sia veramente "tuo". Puoi utilizzarlo così com'è, ma non puoi modificarlo. Per coloro che non sono addentro alla programmazione, il software, ancora adesso, è sviluppato in questo modo: un programmatore usa un linguaggio di programmazione, che grosso modo sembra inglese. Righe di codice, a-capo, commenti, parentesi di ogni tipo. Ma chi sa programmare, sa anche leggere e mettere le mani su quella roba lì. Che si chiama – appunto – "codice sorgente".

2.3 Tre protezioni su un unico oggetto

Il codice sorgente va molto bene per gli umani. Sembra un testo. Sembra tanto un testo, che quando ci si chiese quale protezione il software dovesse avere (per evitarne l'appropriazione da parte di chi non aveva contribuito allo sviluppo) venne naturale pensare al copyright come se fosse un'opera letteraria. E copyright fu. Prima protezione.

Ma il codice sorgente non va bene affatto per le macchine. I computer vogliono le istruzioni in maniera più compatta, serve un "traduttore" che si occupi di interpretare il codice sorgente, controlli che tutto quello che serve per dare le istruzioni ci sia, traduce il tutto in un linguaggio direttamente eseguibile sul computer, metta assieme tutti i pezzi. Questo procedimento si chiama "compilazione" e il risultato è chiamato "codice macchina" o "codice eseguibile" o "codice oggetto". Risalire al codice sorgente dal codice oggetto è estremamente più difficile. In un certo senso, il codice oggetto è segreto.

E segreto fu. Il segreto viene tutelato da apposite norme che impediscono di cercare di risalire al codice sorgente. Dunque, questa è una seconda protezione sul software, in aggiunta al copyright.

Intanto, una particolare setta di avvocati chiamati "*IP Lawyers*", i quali passano il giorno e la notte a inventarsi nuove protezioni, si dissero "ma perché se inventiamo un modo nuovo di far qualcosa con una macchina, possiamo ottenere un brevetto, mentre se inventiamo un modo nuovo di fare qualcosa con un computer, no? Otteniamo un brevetto". Se lo dissero tanto spesso e con tanta convinzione, e lo dissero con altrettanta convinzione ai giudici, che un giorno un giudice diede loro ragione. E brevetto fu. Ecco la terza protezione che insiste sul software.

Sui brevetti potremmo fare un lungo discorso, ma ci porterebbe troppo lontano, basti dire che i brevetti sul software, per quanto di dubbia legalità, soprattutto in Europa, esistono e vengono attivamente tutelati in giudizio.

2.4 Le licenze

Nella distribuzione del software, una licenza non è necessaria. Se vengo in possesso legittimamente di una copia del software, ho alcuni diritti e doveri che discendono direttamente dalla legge. Di principio non posso effettuare copie o prestare il software senza permesso, non posso decompilare il software per accertarne il codice sorgente, non posso utilizzarlo se confligge con un brevetto relativamente al quale non sussiste una valida licenza o non sia posseduto dal produttore.

Una licenza è necessaria se il titolare dei diritti di sfruttamento esclusivo vuole restringere ciò che l'utente può fare rispetto ai divieti e ai permessi che provengono direttamente dalla legge. Ciò è possibile, a condizione di non violare le norme imperative. Una licenza è necessaria anche per espandere questi diritti, e concedere usi e azioni che non sarebbero possibili in base direttamente alla legge, incluso concedere il codice come software libero.

Le licenze di software libero non sono un fenomeno nuovo. In effetti, alcune tipologie di licenze, specie quelle di derivazione universitaria (MIT, BSD) risalgono a un periodo in cui non esisteva un'espressa tutela del software. In ambito accademico tali licenze concedono un uso piuttosto "liberale"del software. In pratica è possibile qualsiasi utilizzo del software, inclusa la copia e la modifica, a due condizioni fondamentali: far sapere chi è l'autore del software (perché l'uso del software da parte di altri è fattore di prestigio per l'autore, e di acquisizione di "credito" universitario) e di escludere ogni forma di responsabilità. Si trattava di software libero o open source *ante litteram*.

2.5 La seconda fase: il copyleft

Richard M. Stallman, il fondatore del nuovo movimento di liberazione del software, si rese conto che in ambito "commerciale" non valeva la regola del tutti

condividono tutto, salvo il dovere di attribuzione, cui era abituato in ambito universitario. Galeotta fu una stampante di rete, una fantastica stampante laser dipartimentale, che aveva però l'abitudine di incepparsi. Siccome la utilizzavano in tanti, per produrre stampe di tante pagine, a ogni inceppamento i lavori restavano in coda fino a che qualcuno non li andava a ritirare e si accorgeva dell'inghippo. A quel punto i lavori a seguire venivano ritardati anche di molto. Per ovviare a ciò Stallman aveva modificato il software in modo che, invece di un semplicemente far accendere un una lucetta sulla stampante, venisse inviato un messaggio in rete, così il proprietario del lavoro inceppato – informato – poteva alzarsi e risolvere subito l'inceppatura.

Quando il dipartimento di Stallman ricevette una nuova stampante, Stallman cercò il modo di effettuare l stessa modifica, ma non trovò – con sua sorpresa – il codice sorgente. Pensando a un errore, contattò il produttore, ma gli venne spiegato che quella era la politica, il software era loro e non rilasciavano i sorgenti. Niente sorgenti, niente modifiche. Stallman non la prese bene, e trovò una soluzione piuttosto radical al problema:fondò il progetto GNU per realizzare da zero un intero sistema operativo (un UNIX) interamente fatto di software libero, ovvero che rispettava le quattro libertà che individuò nel seguente tetralogo:

- Libertà di eseguire il programma come si desidera, per qualsiasi scopo **(libertà 0)**.
- Libertà di studiare come funziona il programma e di modificarlo in modo da adattarlo alle proprie necessità **(libertà 1)**. L'accesso al codice sorgente ne è un prerequisito.
- Libertà di ridistribuire copie in modo da aiutare il prossimo **(libertà 2)**.
- Libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti da voi apportati (e le vostre versioni modificate in genere), in modo tale che tutta la comunità ne tragga beneficio **(libertà 3)**. L'accesso al codice sorgente ne è un prerequisito.

Di UNIX "liberi" già ce n'erano stati. Anzi, UNIX era nato libero. Poi diversi operatori avevano preso quel codice e, grazie al fatto che era licenziato con condizioni permissive, ne avevano creato diverse versioni proprietarie, ognuna incompatibile con le altre. Stallman non voleva che la propria creatura subisse la stessa sorte. E la soluzione fu il copyelft.

Copyleft, ne abbiamo parlato nel capitolo scorso, è un insieme di condizioni che tendono a forzare il fatto che una volta che un'opera è stata licenziata sotto una particolare licenza, tutte successive opere derivate conservino la stessa licenza. Non è però solo una questione di licenza. Occorre anche effettivamente avere la possibilità materiale di modificare il software. Averne semplicemente il diritto non è sufficiente. Ecco perché nel software è particolarmente importante avere accesso al codice sorgente. Ecco perché una delle condizioni più importanti per il copyleft nel software è quella di mettere a disposizione la versione completa del codice sorgente corrispondente alla versione modificata, così che anche lo sviluppatore iniziale possa giovarsi delle modifiche.

Nel software, per questa ragione, “copyleft” viene sovente tradotto con “accesso al codice sorgente”. Questa identificazione della parte con il tutto è alla base anche dell’espressione di “open source”(“sorgente aperto”) rispetto al più corretto “software libero”, in cui “copyleft” è un attributo di alcune licenze (ma ovviamente non di tutte). Non tutto il software libero, anche quello nato in tempi più recenti, infatti, pretende l’accesso al codice sorgente come condizione di licenza. Nondimeno molto spesso anche software non soggetto a condizioni di copyleft vede, di fatto, un accesso anche integrale al codice sorgente, ma senza che vi sia una garanzia che ciò avvenga, come appunto per gli UNIX nati in ambito universitario, e come più tardi accadrà con il sistema operativo dei Macintosh di Apple, MacOSX.

Per il proprio sistema operativo Stallman non vuole affidarsi alla buona volontà di chi lo modificherà, per ricevere gli sviluppi che altri faranno, come fecero i suoi predecessori. Vuole che ciò sia una precisa condizione, ovvero usa per primo il “copyleft”, e inventa la licenza GNU General Public License. Che avrà un successo enorme, tanto da essere applicata a più della metà dei progetti di software libero, e sarà nota sotto il nome più colloquiale di “GPL”.

GNU diventerà un sistema operativo di estremo successo quando al kernel originale (la parte che nel sistema operativo si occupa delle funzioni più di base e dell’interazione tra il software e l’hardware) che Stallman aveva sviluppato (Hurd) verrà preferito quello sviluppato da un giovane (allora) studente universitario finlandese, Linus Torvalds, che lo chiamerà “Linux”. Il sistema GNU + Linux verrà poi conosciuto più generalmente con il nome del solo kernel, ovvero “Linux”. Il successo di questo sistema è tanto enorme quanto in parte misconosciuto. GNU/Linux sarà il sistema operativo che fornirà la spina dorsale di Internet come la conosciamo, su di esso è fondato quasi tutto quello che sostiene Internet come infrastruttura (server, router, apparati di ogni genere) e come servizi: tutti i social network e quasi tutti i servizi in cloud sono basati sotto qualche forma di GNU/Linux, e Linux in combinazione con una variante di Java chiamata Dalvik sarà meglio nota come “Android”.

E’ stata una rivoluzione nata da una combinazione tra un ricercatore, una stampante e una licenza, che ha cambiato l’informatica per sempre. E lo ha fatto non solo con Linux, ma con una serie ormai quasi infinita che ne ha raccolto il testimone in quasi tutti i campi. Nel prossimo capitolo esamineremo il mondo delle licenze di software libero e le loro varie tipologie.

Capitolo 3

Le licenze di software libero (open source)

A volte alla domanda "sotto quale licenza è questo programma?" si sente rispondere "come, che licenza: open source". Il che è come rispondere "nome maschile" alla domanda "come ti chiami?". Come direbbero i francesi "*vive la petite différence*!"

Una licenza è – al fondo delle cose – un testo legale. È anche una dichiarazione programmatica, un manifesto, ma siccome il software libero è un fenomeno giuridico, l'aspetto contrattuale o para-contrattuale è ciò che caratterizza, da molti punti di vista, la licenza. Ovvero, risponde alla domanda "che cosa ci posso fare con il software che sto analizzando?".

3.1 Software libero, software non libero; software open source, software non open source.

Torniamo alla prima risposta. "È software open source".

Uso il titolo di una presentazione che diedi qualche anno fa a Firenze, parafrasando un più noto *pamphlet*[1] di Paolo Rossi (il comico), "Si fa presto a dire open source". Per dire che una cosa è open source, occorrerebbe prima sapere cosa distingue l'open source dal non open source, e il software libero dal non software libero. Cosa distingue il software libero dal software open source, l'abbiamo già detto nelle scorse puntate, è l'approccio filosofico: in termini giuridici, nulla. Le due nomenclature sono operativamente intercambiabili, per cui non

[1] Paolo Rossi, *Si fa presto a dire "pirla"*, Dalai Editore, Milano, 1997. Su Amazon: https://www.amazon.it/Si-fa-presto-dire-pirla/dp/8885988350

ci accapigliamo, ognuno usi quella che preferisce, io preferisco dire “software libero”.

Però i due mondi fanno effettivamente riferimento a due differenti **definizioni**. La definizione di software libero è semplice: è software libero il software la cui licenza soddisfa tutte e quattro le libertà del software, di cui abbiamo già detto nel precedente capitolo, quando ci siamo occupati della storia del software libero. Se ne manca anche solo una, non è software libero.

3.2 La Open Source Definition

La definizione di cosa sia software open source è un po’ più complessa. Il termine stesso open source nasce successivamente al software libero. Abbiamo già detto che il primo globale e cosciente sforzo di creare software libero come tale appartiene a Stallman e alla Free Software Foundation, e consisteva nel sistema operativo GNU. Con l’avvento di Linux, GNU divenne un sistema operativo pienamente funzionante. Una delle prime e più “pure” distribuzioni di GNU/Linux fu Debian.

Cos’è una distribuzione? Una distribuzione è un assemblaggio di software proveniente da fonti diverse, da parte di un operatore che sceglie cosa inserire e come configurare tale software (normalmente, una versione di Linux, una certa quantità di software del progetto GNU, un sistema di gestione dei pacchetti per l’aggiornamento e l’installazione di componenti aggiuntive, applicazioni e così via), compila il software a partire dalle sorgenti e lo distribuisce come un prodotto a se stante. Di qui il termine “distribuzione”. Tutto è GNU/Linux, ogni distribuzione è diversa dalle altre, ma anche simile alle altre.

Debian, appunto, è una distribuzione, che si caratterizza per due aspetti: un proprio sistema di gestione dei pacchetti (tutti con il suffisso .deb) e un relativo programma di gestione (apt), e una scelta piuttosto radicale su quali condizioni debbano rispettare le componenti per essere incluse. Si potrebbe dire “devono essere software libero”, ma il progetto Debian preferì articolare meglio tale requisito e “spacchettare” le libertà in caratteristiche più puntuali, che inserì nelle “Debian Free Software Guidelines”.

In uno sforzo di “vendere” meglio il concetto di software libero, e anche per risolvere una ambiguità del termine “software libero” (che in inglese si dice “Free Software”, dove “free” può anche dire “gratis”) alcuni attivisti e programmatori spinsero per usare un termine d’uso comune che non soffrisse dell’ambiguità “Free Software” = “software gratuito”, ovvero “open source” (“sorgente aperto”). E venne a tal fine fondata la “Open Source Initiative” (OSI) che si diede il compito di “certificare” cosa fosse open source e cosa no. Come riferimento normativo vennero usate le Debian Free Software Guidelines, alle quali vennero rimossi i riferimenti specifici alla distribuzione Debian, che divennero la Open Source Definition.

3.3 Proliferazione

OSI ha nel frattempo approvato un centinaio di licenze diverse. Che non esauriscono affatto l'orizzonte delle licenze possibili. È un numero enorme, ed averne così tante è un problema. Se avessimo poche licenze, con clausole in gran parte identiche e qualche clausola difforme, costruire una interpretazione solida e affidabile, che ricostruisca con una certa qual certezza le conseguenze giuridiche di ciascuna licenza e le condizioni alle quali tali licenze possano essere usate assieme, avremmo un mondo più semplice.

Invece abbiamo licenze che in larga parte sono simili tra loro (appartengono solitamente a una delle grandi famiglie, di cui diremo), ma con differenze che possono a volte determinare ambiguità, passare inosservate e – quel che è peggio – causare incompatibilità reciproche, soprattutto quando si passa al copyleft. Seguendo le discussioni sulle nuove proposte di licenze all'OSI ci si rende conto che in larga parte l'esigenza che tali licenze soddisfano sono o l'ego di chi le propone, per avere il proprio nome legato a una licenza (che poi non userà quasi nessuno) oppure… l'ego di chi le propone, che pensa che le centinaia di menti che collettivamente hanno sviluppato quelle esistenti e adottate dalla maggior parte del software siano degli incompetenti. Solo in alcuni casi, alcune licenze servono a soddisfare esigenze specifiche e sono in parte giustificabili, nella maggior parte dei casi la mia opinione è che **meno è meglio**. Nel dubbio se usare una licenza e scriverne una, contare fino a diecimila e poi comunque **abbandonare l'idea**.[2]

3.4 Le famiglie delle licenze: divisione tra vari livelli di copyleft

Dicevo, esistono grandi famiglie di licenze, che coprono da sole la gran parte del software. Il resto è "coda lunga" (long tail), traducibile anche con "rumore di fondo": poco rilevante, ma sempre rumore.

Uno degli aspetti più importanti delle licenze, il primo che ricerco in una nuova licenza, è: "quanto copyleft?". Solitamente si definiscono tre livelli di copyleft, nel software: **copyleft forte, copyleft debole, nessun copyleft**. Nessuno è in grado di definire con precisione dove inizia uno e finisce l'altro, teniamo queste distinzioni come categorie di massima.

[2]Parlo come autore di una licenza sottoposta e mai approvata da OSI (la licenza MXM (https://www.linuxjournal.com/content/should-open-source-licence-ever-be-patent-agnostic), sviluppata con Leonardo Chiariglione per MPEG).

3.5 Licenze non copyleft

"Nessun copyleft" è abbastanza facile da capire: il codice sviluppato può essere preso, modificato, rilicenziato, senza che il codice così risultante debba soggiacere alla stessa licenza. Le licenze che seguono questo paradigma vengono dette anche "ultraliberali", nel senso che consentono di fare quel che si vuole (una si chiama appunto "WTF – *What The Fuck [you want] – license*": non traduco per decenza). In ciò rientra anche rendere il software interamente proprietario, secondo la parabola dei primi UNIX, di cui abbiamo parlato nello scorso capitolo su cui non torneremo.

In tali licenze, che per la tradizione storica vengono anche dette "accademiche" abbiamo licenze legate appunto all'ambito universitario: la **BSD** (dove "B" sta per "Berkeley"), e, meno diffusa, la **MIT**. La BSD in realtà esiste in tre sfumature, a seconda di quante clausole è composta (la più utilizzata è senz'altro la three-clauses). La MIT, invece, esiste in una serie quasi infinita di varianti. Sono licenze molto semplici, in cui le principali condizioni sono quelle di riconoscere l'assenza di responsabilità dello sviluppatore e quella di indicarne il nome dell'autore quando si distribuisce il codice sorgente. Queste licenze sono molto popolari tra alcuni sviluppatori perché sono molto semplici e non vi abbonda il "legalese".

Alle accademiche si accompagna un'altra licenza molto utilizzata, la **Apache Public License**, licenza usata dalla Apache Software Foundation, che si occupa del più famoso e utilizzato server web. La Apache, al contrario delle accademiche, è una licenza lunga, che comprende anche una clausola di risoluzione nel caso di uso aggressivo dei brevetti da parte di un licenziatario.

3.6 Licenze di copyeft forte

All'altro estremo abbiamo il copyleft forte. Tendenzialmente una licenza di copyleft forte tende a estendere il suo effetto vincolante il più possibile a qualunque "opera derivata" di software copyleft. L'estensione di tale effetto è una delle materie su cui sorgono più spesso le discussioni più accese tra coloro che si occupano della nostra materia. Il copyleft forte richiede anche una struttura normativa più complessa e stringente, perché essendo "restrittivo" (nel senso, tende a imporre condizioni più stringenti ed effettive al fine di mantenere le libertà del software), richiede che tutti i "buchi" siano tappati, tutte le scappatoie siano evitate, in modo che il copyleft forte, e il tentativo di tenere il software un "commons" resista a chi cerca di evitarlo. Creare una licenza di copyelft forte è molto difficile, crearne una fatta bene è cosa a portata di ben pochi.

Siccome le licenze di copyleft forte hanno condizioni più stringenti e peculiari, è molto facile che licenze di copyleft forte siano incompatibili con qualsiasi altra licenza di copyleft forte, dimodoché combinare software sotto due licenze di co-

pyleft forte è in pratica impossibile. Anche combinare copyleft forte con copyleft debole, e a volte anche con non copyleft, si rivela sovente impossibile. Se non si possono rispettare le condizioni di licenza –tutte le condizioni – non si può usare il software, non si possono trarre opere derivate.

La licenza di copyleft forte per antonomasia è la **GNU General Public License, o GPL** della FSF. Giunta alla terza versione, nella versione 2 è storicamente di gran lunga la più utilizzata licenza di copyleft forte, e la più utilizzata licenza in assoluto. Discutere di cosa c'è nella GPL richiederebbe un volume, non un articolo, figuriamoci pochi paragrafi. La versione 3 è incompatibile con la versione 2, a meno che lo sviluppatore non abbia utilizzato la clausola di "aggiornamento" ("o qualsiasi successiva versione") che consente a chiunque di trasformare una versione di licenza più vecchia con una nuova.

Esistono altre licenze di copyleft forte, come ad esempio la **EUPL**, nata in ambito delle istituzioni dell'Unione Europea. Creare una nuova licenza di copyleft forte, lo si capirà da quanto detto, non è una buona idea, la EUPL non lo è stata. Fortunatamente la EUPL contiene una clausola di compatibilità che consente di utilizzare, per le opere derivate, una lista di altre licenze, tra le quali la GNU GPL, per cui le conseguenze dannose sono in via pratica evitate.

3.7 Licenze di copyleft debole

Il copyleft debole è più recessivo rispetto al copyleft forte. Come idea, il copyleft debole opera a livello di file. Per cui se uno sviluppatore appone le proprie modifiche a quel file, quel file rimane sotto la stessa licenza. Mentre se codice sorgente di quel file viene compilato ("linkato") con altro codice sorgente, per creare un file oggetto che li comprenda (un'opera più ampia), il file oggetto risultante – pur incorporando codice copyleft – non necessita di essere rilasciato sotto le stesse condizioni, dunque non c'è interferenza in caso i due file sorgente siano sotto licenze incompatibili, e il prodotto risultante può anche essere sotto licenze proprietarie. Spesso infatti le licenze di copyleft debole sono usate per file destinato ad essere incorporato in altro software, tipicamente librerie (vedi sotto per la LGPL). Questo è detto in modo molto brutale, ma serve a dare l'idea.

Per il copyleft debole esistono due licenze paradigmatiche. La **GNU Lesser General Public License** (un tempo "Library General Public License", in quanto nata per le librerie GlibC, le librerie del compilatore C del progetto GNU) o **LGPL**; e la **Mozilla Public License** o **MPL**, la licenza di Firefox, per intenderci, uno dei più diffusi browser web. Entrambe sono compatibili con la GPL: anche la MPL, nella versione 2, in quanto contiene ha una espressa clausola di compatibilità con la GPL.

La caratteristica di essere compatibili con la GPL, vuoi per le condizioni, vuoi per un'espressa condizione di compatibilità, è una caratteristica molto importan-

te delle licenze, alla pari, quasi, con il fatto di essere copyleft di un certo tipo. La FSF pubblica una lista di tali licenze, secondo la propria interpretazione, ovviamente.

3.8 Concludendo

Solo riassumendo le licenze di cui abbiamo fatto menzione, abbiamo

- 3 tipi di BSD
- la MIT (multiforme)
- Due versioni di Apache
- Due versioni di LGPL
- Due versioni di Mozilla
- Due versioni di GPL
- Due versioni di EUPL

Dunque 14 licenze. In più, ci sono in circolazione diverse versioni della GPL con "eccezioni", studiate apposta per rendere compatibile il software con altro software sotto licenza diversa. Ce n'è veramente a sufficienza per non andare a cercare altre licenze e per dare materiale di studio ai legali. E questo senza contare la possibilità che il software sia sotto pubblico dominio.

E non abbiamo nemmeno accennato alla Affero GPL: una gemmazione della GPL nata per il Software as a Service, ora diventata una terza famiglia delle -GPL (AGPL, GPL, LGPL) e compatibile con la GPL solo grazie a una clausola di compatibilità espressa.

Pensavate fosse facile? Nulla in questo campo è facile. Ma è tutto affascinante. Ora, almeno, avete una mappa ragionevolmente completa delle licenze di software. Nel prossimo capitolo ci occuperemo di alcuni aspetti delle licenze di software copyleft, del dual licensing e di alcuni aspetti peculiari dello sviluppo di software sotto tali licenze. In altre parole, dei modelli di business legati alle o consentiti dai diversi tipi di licenze.

Capitolo 4

Licenze di software libero e modelli di business

Capita sovente che clienti si rivolgano a me per consigli su come scegliere una licenza piuttosto che un'altra, ma in realtà quello che chiedono riguarda la strategia "di business". Le prime volte dicevo "ditemi la vostra strategia di business, io vi dirò che licenze sono disponibili date le componenti che volete utilizzare". Ma mi sono reso conto molto presto che si tratta di un classico problema uovo-gallina.

4.1 Viene prima l'uovo o la gallina?

La scelta di un modello di business deriva dalla licenza che si intende adottare, o la licenza che si in tende adottare dipende dal modello di business che si è scelto?

Sovente, nessuna delle due opzioni. O tutte e due. Come in un puzzle, conviene iniziare dai pochi punti noti. Quasi mai si inizia da una situazione del tutto vergine. Quasi sempre si ha già un minimo di progetto abbozzato, se non un prototipo. A volte si ha addirittura non un vero e proprio prodotto sul mercato che si vuole rendere "open source". Allora la scelta dipende in larga misura da costrizioni esterne; oppure dalla voglia di disfare ciò che si è già fatto, se le costrizioni esterne non consentono seguire la strada che si vorrebbe prendere.

Nel gergo si parla di "**inbound**" (in ingresso) per definire il software già esistente e le relative condizioni di terzi che inseriamo nel nostro lavoro e "**outbound**" (in uscita) per il software che distribuiamo e la licenza che utilizzeremo per tale distribuzione.

4.2 Qualcosa di nuovo, qualcosa di vecchio, qualcosa di prestato, qualcosa di rosso

Le condizioni di licenza inbound possono richiedere che il software derivato sia sotto una determinata licenza o categoria di licenze (outbound), oppure possono addirittura escludere la licenziabilità del software sotto una o più licenze incompatibili. Dunque una qualsiasi decisione deve per forza partire da una ricognizione di quello che c'è già, per vedere di chi è, se va buttato e se va buttato cosa se ne deve andare con esso.

Se il software è interamente sviluppato dal nostro imprenditore, non c'è problema. O c'è? In teoria può accadere che qualcuno (uno stagista, un ambiente di sviluppo, una libreria che faceva comodo e di cui ci si è dimenticati) non abbia inserito nel codice qualcosa di altri, che viene sotto una licenza, le cui condizioni siano incompatibili con la licenza outbound. A seconda delle dimensioni del progetto e della sua anzianità, o del fatto che è stato acquistato da terzi (magari dopo l'acquisizione della società che l'aveva creato), queste informazioni possono non essere direttamente reperibili. Nel qual caso esistono strumenti di analisi del codice sorgente, tra i quali Black Duck è il più famoso e completo, a pagamento, e altri gratuiti. Questi strumenti restituiscono un rapporto più o meno accurato che costituisce una buona base di partenza.

Per dirla in termini tecnici, occorre sempre fare un minimo di attività di controllo, che sovente viene chiamata "*due diligence* tecnica", per scoprire *prima* quello che se si scopre *dopo* si piangono lacrime amare.

4.3 Scegliamo la licenza, no, il modello di business, no, cosa?

I modelli più gettonati sono: _dual licensing, *open* core, *freemium*, *subscription*, personalizzazioni e (scusate l'abuso di inglese) io speriamo che me la cavo.

4.3.1 Dual licensing

Il modello "*dual licensing*" è andato molto di moda in passato, faceva piuttosto gola agli investitori esterni (*venture capitalist*) perché sembrava l'uovo di Colombo. Avere i benefici del software libero e consentiva di monetizzare come se fosse software proprietario. Tombola!

Aggiungiamo che nel boom delle aziende tecnologiche c'era almeno un esempio vincente: MySQL AB, che con l'omonimo database campione di vendite nel boom delle *dotcom*, era stata comprata da Sun Microsystems per due miliardi di dollari. L'idea che il trucco potesse essere ripetuto ha fatto gola a molti.

Il funzionamento è semplice: stesso software, due licenze. La prima è una licenza di software libero. Invado il mercato, distribuisco a costo zero, creo un nome, creo effetti di rete, la gente sviluppa sul mio prodotto, lo faccio diventare appetibile. E lo distribuisco anche sotto licenza proprietaria. Perché uno dovrebbe pagare per qualcosa che già è disponibile gratuitamente? Per avere l'assistenza? Garanzie? Aggiornamenti? No, quello è il modello "*subscription*" (vedi sotto). Qui stiamo parlando di una licenza proprietaria.

Per far funzionare questo modello, la licenza di software libero deve essere una licenza copyleft. Più forte è il copyleft, più è difficile utilizzarla per un prodotto proprietario. Per cui chi vuole usare il prodotto con una licenza outbound proprietaria deve per forza ottenere un permesso ulteriore, un'**eccezione** al copyleft.

Il modello ha due debolezze: per quanto forte sia il copyleft, i suoi effetti si dispiegano su ciò che può essere considerato "prodotto derivato". Dunque occorre che il software incorporante sia inseparabile da quello incorporato, o che sia inefficiente disaccoppiare i due sistemi per renderli indipendenti.

La seconda debolezza è che per avere la possibilità di concedere eccezioni al copyleft, occorre **il permesso di tutti i titolari**. In uno sviluppo distribuito, è difficile ottenere tale permesso in capo a un solo soggetto, perché difficilmente un terzo investe in un prodotto che altri sfrutteranno. Manche così, occorre che chi contribuisce sottoscriva un accordo speciale, di assegnazione ("*Contribution Assignment Agreement*") in cui si cede il copyright sui contributi, ricevendo in cambio una licenza illimitata. Non una cosa molto appetibile per i più. Per cui si finisce per tornare a un modello di sviluppo "a silos" in cui tutto il lavoro è comunque sulle spalle di uno solo.

Un esempio del fallimento di tale modello, che ha portato a un "*forking*" (divisione) del progetto è OpenOffice.org, sempre di Sun Microsystem. Alcuni sviluppatori hanno creato la Document Foundation per un nuovo progetto: Libre Office. Il quale ora ha un numero enorme di sviluppatori indipendenti che contribuiscono regolarmente. OpenOffice.org ne riceve molti, molti meno.

4.3.2 Open Core

Il modello "*open core*" si basa sul concetto di creare un nucleo ("core") aperto, a cui possibilmente molti partecipanti contribuiscano in modo sostanziale.

In questo modo, si garantisce che almeno sull'infrastruttura centrale si possano ricevere vari contributi, visto che molti potranno comunque giovarsi. Per poi realizzare soluzioni a valore aggiunto nella parte "esterna", che può anche essere molto piccola, ma specializzata. Ad esempio, creare un motore di database, per poi realizzare database di livello industriale (es: PostgreSQL[1]), oppure un'infra-

[1] https://www.postgresql.org/

struttura di gestione dei flussi multimediali, utilizzabile da chi volesse costruirci sopra applicazioni multimediali di ogni tipo (esempio: GStreamer[2]).

Qui, al contrario del dual licensing, i prodotti proprietari e liberi sono diversi (spesso l'uno contiene l'altro) e la licenza può essere non copyleft, copyleft debole o copyleft forte con eccezioni. Proprio perché il "core" è destinato a essere compreso in un prodotto che potrà essere proprietario, e ci potrebbero essere incompatibilità.

Spesso questa è la strada scelta da progetti completamente liberi, in cui si inizia con un modello di licenza completamente libera. Poi nell'evoluzione si valuta che il modello iniziale non è sostenibile, perché da progetto semi-hobby è diventato qualcosa di serio e non è più gestibile, oppure perché per finanziare lo sviluppo necessita un introito derivante dalla messa a reddito.

4.3.3 Subscription/personalizzazioni ibridi

Il modello che monetizza mediante l'offerta di "*subscription*" e personalizzazioni è quello più tipico del software libero.

Non è incompatibile con un modello dual licensing, anzi, i progetti dual licensing che conosco essere più di successo in realtà combinano un modello duale, un open core e un modello subscription. Ad esempio, Zimbra[3], Alfresco[4], OpenBravo[5], Zarafa[6] non puntano tanto ad essere inclusi in progetti più ampi, quanto ad avere un canale "comunitario", nel quale la comunità è più di utilizzatori e di integratori che di sviluppatori, e offrire un servizio "premium" a chi necessità di livelli di supporto garantiti, includendo magari porzioni proprietarie di software per scopi precisi, non disponibili nella versione comunitaria.

Questo tipo di strategia ha alcuni vantaggi, ad esempio costituisce una garanzia di indipendenza per il cliente, che al massimo dovrà migrare verso una versione "comunitaria" tutto sommato simile, e rimpiazzare le parti proprietarie o con le versioni sviluppate dalla comunità, oppure da imprenditori concorrenti (nel caso di Zimbra, ad esempio, Zextras[7]), oppure create ad hoc da ditte specializzate sulla tecnologia. Inoltre si segmenta il mercato, offrendo una soluzione premium a chi è interessato a livelli di servizio e prestazioni più elevate, mentre non si rinuncia all'effetto di rete dato dall'utilizzo di gran parte della base di codice per chi non ha risorse o necessità per giustificare il livello premium.

[2] https://gstreamer.freedesktop.org/
[3] http://www.zimbra.com/
[4] https://www.alfresco.com/
[5] http://www.openbravo.com/
[6] https://www.zarafa.com/
[7] https://www.zextras.com/

4.3.4 Modelli di *subscription* "puri".

In realtà, il modello più aderente alla filosofia del software libero è quello "puro" di servizi.

Qui ci sono spesso fraintendimenti. Può accadere che da un punto di vista esteriore, la forma giuridica del rapporto tra l'operatore economico e il cliente sia, in modo un po' sorprendente di vendita di "licenze".

"Ma come", i più attenti avranno già pensato "comperare software libero non si può". Ni. L'acquisto di una copia del software conclude comunque un contratto di vendita in entrambi i casi. Nel contratto di acquisto di software proprietario si acquista, oltre alla copia in sé (il software, sia che venga consegnato su un supporto o da una risorsa online) anche un diritto d'uso di una o più copie, oltre a una serie di prestazioni accessorie. Acquistando invece una copia di software libero, non si acquista il diritto d'uso, perché quello è già garantito dalla licenza originale, e tale licenza viene concessa per effetto stesso della distribuzione.

Il distributore non può subordinare al pagamento di una somma di denaro o all'approvazione di clausole aggiuntive i diritti già concessi dalla licenza, in questo dunque la sorpresa del lettore è legittima. Ma nessuna licenza (salvo una limitata eccezione) impone di *distribuire* il software libero che si detiene, anche quello che si è modificato. Per cui legittimamente il detentore di quella versione del software può pretendere un compenso per consegnarne una copia. Il destinatario poi potrà in perpetuo esercitare tutti i diritti assegnatigli, compreso quello di distribuire altre copie o, nel caso di software copyleft, di ottenere il completo corrispondente codice sorgente della copia che ha ricevuto.

Indipendentemente dalla modalità di acquisizione, il rapporto tra le parti in un modello di *subscription* ha caratteristiche per certi versi simili a quello proprietario. Ad esempio è facile osservare un limite massimo di copie installabili oppure l'obbligo di installare il software su tutte le macchine di una certa categoria (Red Hat). Anche qui, la limitazione non deriva dalla licenza, ma dal contratto ed è una condizione per poter fruire delle altre prestazioni contrattuali. Ancora, può capitare di rinvenire nelle condizioni un divieto di modifica del software o di caricare versioni compilate da sé. Anche qui, si tratta di una condizione contrattuale, non di una condizione della licenza. La differenza principale è che in caso di violazione, la sanzione sarà di natura contrattuale, ad esempio l'inoperatività della garanzia, l'inesigibilità dell'obbligo di assistenza, la risoluzione contrattuale.

Al termine del contratto, il cliente sa che potrà conservare le versioni installate del software, modificarle, installarle su altre macchine. Insomma, sarà garantito contro il *lock-in* perché, venuto meno il contratto, il software rimarrà pur sempre software libero.

Il cliente potrà cedere copie a terzi? In base alla licenza, sicuramente. In base al contratto, probabilmente sì (altrimenti ci potrebbe essere un vincolo inconcilia-

bile con la licenza). Tuttavia c'è da considerare un elemento di cui ancora non abbiamo parlato, e che conclude la nostra trattazione.

4.4 Il marchio

Il marchio, in un modello di business legato al software libero, è forse il diritto di privativa più importante nel controllare e garantire sufficienti ritorni. E forse è anche l'unico che non ha grandi incompatibilità (se non in casi difficilmente concepibili) con nessuna licenza di software libero. Ed è per questo che non mi stancherò di ripetere che in qualsiasi progetto o attività legate al software libero un investimento sostanziale sui marchi è da considerare quanto più presto possibile.

Per quanto riguarda la monetizzazione dell'attività, il marchio ha ovviamente la funzione di incrementare il valore del fornitore, rendendolo riconoscibile e consentendo di riconnettere, nella visione degli utenti, i valori positivi del prodotto al suo produttore (anche quelli negativi, ma è un'altra storia) e viceversa. Con una politica accorta del marchio, si potrà distinguere il proprio prodotto e il proprio servizio da quello di concorrenti, anche nel caso in cui si tratta di software tutto sommato simile. E anche nel caso in cui il software sia proprio lo stesso, bit per bit. Un esempio chiarirà meglio: Red Hat realizza una delle distribuzioni di Linux più conosciute, soprattutto appetibile in installazioni di livello industriale. Oracle ha potuto prendere l'intera distribuzione di Red Hat e offrirla ai propri clienti tale-quale. L'unica differenza è che non ha potuto "venderlo" con il marchio Red Hat. In questo caso è difficile stabilire se valga di più il marchio Red Hat o quello di Oracle, quello che importa è che i prodotti abbiano un marchio *diverso* così ognuno "vale" per quanto sono i meriti della propria attività, e non ci si appropria di quella altrui.

Resta il fatto che il nome del prodotto da cui il prodotto rimarchiato deriva potrà essere comunque presente, ma in funzione descrittiva, per designarne l'origine, dunque in buona fede, in modo non apparente come il marchio del secondo operatore, in modo da non causare confusione quanto all'origine del prodotto e l'identità del produttore.

Allo stesso nel caso di un operatore che realizza software open core, in dual licensing o ibrido, con un marchio riconosciuto, un concorrente potrà affermare di supportare la versione community della propria controparte, usando per offrire i propri prodotti e servizi il marchio del produttore originale, in versione descrittiva.

Sarà possibile anche per il concorrente distribuire la versione community della propria controparte? La risposta è tendenzialmente negativa.

Il principio di esaurimento prevede che il marchio (ma anche il copyright) non possa essere utilizzato per controllare l'ulteriore diffusione di un prodotto da

parte di chi ne abbia acquistato una copia. Una volta acquistata tale copia, essa può dunque essere rivenduta (vedi la sentenza UsedSoft per il copyright). È piuttosto evidente però nel caso di una licenza di software libero, chi acquisisce la copia non abbia solo il diritto di usare quella copia, ma abbia molti diritti ulteriori. Facendo circolare copie del prodotto originale, non sta "rivendendo" la propria copia, non sta dunque ri-distribuendo un singolo individuo della specie, ma sta facendo un'attività di creazione e distribuzione di prodotti "nuovi" (le virgolette sono d'obbligo), attività che deve essere autorizzata dal titolare del marchio.

4.5 Conclusioni

Abbiamo solo scalfito la superficie, i modelli di business ottenibili dal software libero sono molteplici e le combinazioni infinite. Non abbiamo trattato dei modelli che usano il software per vendere hardware, o per popolare la propria offerta in cloud, non trattandosi di modelli in cui il software distribuito è la pietanza principale. Ma è un modello che non può essere sottaciuto, in un mondo in cui alcune delle più grandi società del mondo si basano quasi esclusivamente su software libero utilizzato per il cloud, da Amazon a Facebook, da eBay a Google.

Nota: nell'articolo sono stati utilizzati nomi di operatori commerciali e prodotti. Alcuni di essi sono o sono stati miei clienti. Nessuna delle valutazioni operate sul loro business model riflette conoscenze segrete o deve essere attribuita ai rispettivi operatori.

Capitolo 5

I contenuti open e le Creative Commons

Il software ha dato e continua a dare all'analisi dell'open-qualcosa un contributo essenziale e fondamentale, vuoi perché per primo ha sviluppato una serie coerente e completa di licenze, sia perché l'applicazione delle licenze di software libero addirittura è avvenuta prima della formalizzazione della protezione del software nell'ambito del diritto d'autore, per non parlare del diritto dei brevetti, che è arrivato molto tardi in tale settore.

Una grande popolarità ha però investito da parecchi anni un settore totalmente diverso dal software, ovvero quello più classico dei *contenuti creativi* (usiamo per convenzione "contenuti creativi" o "autorali" per letteratura, musica, fotografia, pittura, cinematografia, eccetera). Le licenze di software libero sono state pensate per il software e mal si adattano alle opere creative. Da un lato le licenze di software si soffermano su questioni tecniche, come la distinzione tra codice sorgente e codice oggetto, che in campo autorale non hanno senso. Dall'altro, per la natura stessa del software, parlano sostanzialmente di distribuzione file di dati, mentre i contenuti creativi hanno molte più e diverse forme di fruizione e distribuzione, e una serie di diritti aggiuntivi e connessi che non si riscontrano nel software.

Tolte queste differenze, comunque, parlando di openness nei contenuti creativi, le motivazioni e la filosofia di fondo non cambiano: si tratta di creare uno spazio di condivisione in un mondo in cui il diritto d'autore riserva al titolare tutti i diritti di esclusiva. E si tratta di concedere tanti diritti quanti servono per la fruizione del bene, ma non tanti da consentire – a scelta del titolare – la proprietarizzazione delle opere derivate, dunque creare uncopyleft. La principale, anzi, quasi totalitaria forma di licenziamento dei contenuti autorali si condensa in una famiglia di licenze: le **Creative Commons**.

5.1 Le Creative Commons: una famiglia di licenze "open" (ma anche non) per i contenuti creativi

Le Creative Commons sono una famiglia di licenze concepite da un gruppo di giuristi guidati da un professore di Harward, Lawrence Lessig[1] , precisamente per portare nel campo dei contenuti creativi gli stessi principi del software libero. Lessig ha non solo creato le licenze, ma anche un ente nonprofit[2] con lo stesso nome, che si occupa di gestire e creare nuove versioni della licenza.

Attualmente le licenze Creative Commons sono giunte alla **versione 4**, la prima che comprende espressamente tra i diritti concessi anche quello "*sui generis*" sui database, di cui parleremo quando ci occuperemo di open data, nonché la prima a non avere differenti **versioni nazionali** (esiste solo la versione "internazionale").

Parliamo di licenze al **plurale** perché non vi è una sola licenza, ma ve ne sono molte. Come nei giochi delle costruzioni, abbiamo elementi che possono combinarsi tra loro per creare una licenza con determinate caratteristiche. E poi abbiamo la **CC zero** di cui parleremo più avanti. Tali elementi sono (nome esteso e tra parentesi la sigla):

- "Attribution" (by)
- "Non commercial" (NC)
- "Share Alike" (SA)
- "No derivatives" (ND)

Così, se troviamo l'indicazione "CC by-SA" sapremo che si tratta di una Commons "Attribution-Share Alike". In altri casi potremmo avere "CC by-SA UK" o qualcosa di simile, vorrà dire che sarà la versione del Regno Unito, ma come detto ciò non è più un'opzione dalla versione 4.

Una caratteristica importante delle licenze è che sono **compatibili verso il futuro**, ovvero, contengono una clausola implicita grazie alla quale le opere cui sono soggette vengono sì licenziate in una determinata versione, ma possono essere distribuite rispettando le condizioni di una versione successiva che abbia gli stessi elementi fondamentali. Anche la licenza Mozilla per il software (MPL) ha una clausola molto simile, così come la GNU GPL, che però la presenta in modo opzionale.

Creative Commons fornisce anche un **insieme di elementi identificativi grafici** (in diverse funzioni) che rende immediatamente visibile a un essere umano la tipologia di licenza e il fatto che "alcuni diritti sono riservati" (ma non tutti). Fornisce[3] anche un **codice HTML-RDF** per far sì che anche un computer e

[1] https://en.wikipedia.org/wiki/Lawrence_Lessig/
[2] http://creativecommons.org/
[3] http://creativecommons.org/choose/

soprattutto un motore di ricerca possa comprendere automaticamente le condizioni di licenza e consentire rispettarne le condizioni di attribuzione in modo altrettanto automatico (vedi sotto).

5.2 Attribution (by)

Tutte le licenze Creative Commons prevedono questa clausola. Dunque tutte le licenze che troverete in giro saranno CC by o CC by-qualcosa. Dunque la CC by è la forma più semplice e insieme più liberale delle licenze, nel senso che non ha condizioni di copyleft. In tal senso è molto simile, operativamente, a una licenza BSD o MIT. Il materiale CC-by può dunque essere utilizzato per scopi commerciali, modificato, adattato, incluso in materiale proprietario, purché venga rispettata l'attribuzione originale.

Chi conosce il diritto d'autore nostrano si potrebbe aspettare che tale elemento sia imposto dall'inalienabilità del **diritto morale** di essere riconosciuto autore. In parte sì, ma *molto* in parte. Intanto, tale diritto morale non esiste in tutte le giurisdizioni: ad esempio né negli Stati Uniti né in Gran Bretagna esso non è presente. Inoltre, l'attribuzione può indicare l'autore, ma non sempre ciò è vero. Spesso viene indicata una società o un ente: quella è l'attribuzione che deve essere rispettata, anche se ovviamente, trattandosi di un diritto personale, non cedibile, ma solo esercitabile in via ereditaria da ascendenti e discendenti diretti, non può essere una ente giuridico "l'autore". Anche qui ho avuto modo di discettare con una visione totalmente errata della clausola "attribution" in materia di open data, ne parleremo a tempo debito. Ricordo infine che il diritto morale di essere riconosciuto autore, nel nostro diritto, non richiede comunque un'attribuzione esplicita della paternità, ma soltanto di non negarla quando venga rivendicata, ovvero di non attribuire l'opera a se stessi o a una persona diversa dall'autore, oppure, ancora, di non rimuovere degli elementi identificativi che siano stati apposti a tal fine.

Inoltre, la clausola Attribution richiede che venga riportata la dizione completa *se e in quanto viene riportata originalmente, e nella maniera indicata dal titolare*:

- Dell'indicazione dell'autore o di chi debba ricevere la menzione (incluso uno pseudonimo);
- La "*copyright notice*" (l'indicazione del copyright);
- L'esclusione delle garanzie;
- Un URI (Uniform Resource Indicator) o un hyperlink all'opera licenziata

Inoltre deve essere indicato che e in che misura (se fatto e se possibile) il materiale è stato modificato e l'indicazione che l'opera è licenziata sotto la licenza Creative Commons prescelta. Come si vede dunque, da una parte l'obbligo non coincide con (e non necessariamente rispetta!) il diritto morale di essere riconosciuto autore, dall'altra si prevedono una serie di elementi che vanno al di là della semplice paternità.

L'attribuzione si estende in particolar modo anche alla menzione della licenza. Ecco perché è un elemento essenziale della licenza stessa.

Una condizione particolare, presente in tutte le licenze Creative Commons e che costituisce una vera e propria clausola di copyleft (vedi il capitolo introduttivo, è quella per cui non è possibile apporre, né all'opera originale, né ad opere derivate, **ulteriori restrizioni tecnologiche** che limitino i diritti e le facoltà di utilizzo presso il pubblico. Ciò espressamente comprende misure tecnologiche, ovvero **DRM** (Digital Rights Management), strumenti anti-copia o che limitino in altro modo la possibilità di fruire liberamente dei contenuti.

5.3 Share alike (SA)

La clausola Share Alike, ("condividi allo stesso modo", ma tradotta anche più semplicemente "stessa licenza") è la condizione di principale **copyleft** nelle Creative Commons. Significa che ogni *opera derivata* deve essere obbligatoriamente licenziata sotto la stessa licenza o una licenza Creative Commons successiva con gli stessi elementi di quella "inbound".

Semplice, vero? Mica tanto. Come per il software, anche per i contenuti artistici a volte è difficile capire cosa sia un'opera derivata oppure no. Per fare un esempio, una recente sentenza[4] negli Stati Uniti ha ritenuto che la clausola Share Alike non costringe l'editore di un atlante a rilasciare l'atlante stesso sotto condizioni CC by-SA qualora sia stata usata per la copertina una foto altrui licenziata (su Flickr) sotto tale licenza, in quanto non si tratta di opera derivata, ma di semplice aggregato. Per avere un'opera derivata occorre che il materiale di provenienza sia trasformato, adattato o tradotto in maniera che fosse inscindibile con l'opera derivata, ma si è limitato a poche modifiche che non hanno alterato la natura della foto e richieste dalle tecniche di stampa.

Un caso è abbastanza chiaro: se immagini in movimento vengono poste in sincrono con un contenuto licenziato sotto condizioni Share Alike, per rispettare la licenza occorre che l'insieme venga licenziato allo stesso modo.

Con la Share Alike termina anche il novero delle clausole che – fatte le dovute correzioni – corrispondono alle quattro libertà del software libero, che vengono da Creative Commons contrassegnate come "Free Culture", ovvero che assicurino la libertà di riutilizzo pieno e illimitato dei contenuti in modo creativo e trasformativo. Le altre due condizioni, infatti, costituiscono una restrizione: una degli scopi di utilizzo, l'altra delle modalità di utilizzo. La definizione di Free Culture si trova in Freedomdefined.org[5].

[4] vedi il mio commento su MySolutionpost "Arriva una sentenza americana sulle Creative Commons (ma ci dice cose che sappiamo già)" http://www.mysolutionpost.it/blogs/it-law/piana/2015/09ve/creative-commons.aspx

[5] http://freedomdefined.org/Definition

5.4 Non Commercial (NC)

La clausola NC proibisce di utilizzare l'opera (o suoi derivati) per scopi di **vantaggio commerciale o diretto compenso monetario**. Cosa sia tale sfruttamento è – sorpresa sorpresa! – abbastanza difficile dirlo.

Cominciamo dai casi chiari. *Vendere* una copia è contrario alla clausola NC. Allo stesso modo utilizzare un contenuto in un servizio a pagamento (come una pay-radio o una pay-TV) è abbastanza sicuramente contrario alla clausola NC. Inserirlo in un servizio free-to-air, ma in un servizio commerciale che si sovvenziona grazie alla pubblicità è invece un caso piuttosto limite. Anche l'uso per un sito promozionale di una no-profit potrebbe essere considerato "commerciale" e dunque vietato, specie se le no-profit sono in concorrenza per sovvenzioni pubbliche o private (come il nostro 5 per mille).

Un caso che sicuramente non viola la clausola, invece, è quello in cui inserire contenuti è un requisito per partecipare a una rete di file-sharing, a condizione che non ci sia alcuna transazione monetaria nel processo.

Cosa dire invece della diffusione sonora in un centro commerciale? E cosa dire della musica diffusa in un locale dove si servono alcolici e si danza (con-senza consumazione obbligatoria)? E in una discoteca? Si va da usi probabilmente leciti a usi decisamente non consentiti.

Si noti che, essendo una limitazione di utilizzo, la clausola copre non solo gli atti di distribuzione (come quasi tutte le condizioni copyleft delle licenze di software tranne la Affero), ma anche gli utilizzi "interni" che presuppongono il necessario permesso dell'autore, inclusa la copia temporanea necessaria per fruire del contenuto. Pertanto anche la proiezione di un filmato a fini di istruzione di una classe pagante, ad esempio in un'università, violerebbe la clausola e determinerebbe dunque il mancato rispetto del diritto d'autore (salvo, come sempre, il caso in cui sia un utilizzo "libero" per legge, o un "fair use", dove tale principio si applica, come negli USA).

5.5 No derivatives

Qui invece la cosa è più semplice. Nessun'opera derivata è consentita. Punto. Prendi com'è, non cambiarla. Non tradurla, non trasformarla per inserirla in un contenuto più ampio, a meno che non si tratti di una semplice opera collettiva (una compilation, un'antologia).

Anche questa condizione contravviene alla definizione di Free Culture.

5.6 Così quante ne abbiamo?

Le combinazioni possibili delle licenze Creative Commons non sono, come ci si attenderebbe, tutte le permutazioni degli elementi disponibili, ma – per ragioni di incompatibilità tra le condizioni ND e SA, e perché "by" è sempre presente – solo le seguenti sono utilizzabili:

- CC by
- CC by-SA
- CC by-ND
- CC by-NC
- CC by-NC-ND
- CC by-NC-SA

Ovvero sei in tutto. O sette?

5.7 Creative Commons Zero (CC0)

Menzione a parte merita la Creative Commons Zero. Che non è una licenza "some rights reserved" come le licenze di cui abbiamo parlato sopra, e per cui non viene considerata omogenea rispetto alle altre. In effetti, per rimarcare ancora di più la differenza, ha un numero di versione diverso dalle coeve licenze CC.

Serve a creare il **pubblico dominio**, dunque a rimuovere tutte e qualsiasi le restrizioni che insistano su un'opera intellettuale (incluso sul software!), tale per cui chiunque è in grado di usare quell'opera per tutto ciò che vuole, senza chiedere il permesso. La CC0 è infatti definita in Inglese "waiver", che significa "atto di rinuncia"; infatti con essa il titolare dei diritti di proprietà intellettuale dichiara di voler rinunciare per sempre al loro esercizio, liberando quindi l'opera da ogni vincolo di privativa, prima che avvenga la naturale scadenza dei termini per la caduta in pubblico dominio. Negli ordinamenti giuridici nei quali non esiste un concetto di "pubblico dominio" la CC0 concede una licenza la più illimitata possibile.

Anche così l'effetto non è sempre quello di un vero e proprio pubblico dominio, perché qualche condizione rimarrà sempre appiccicata laddove sussistono diritti indisponibili, in principal modo quelli morali. Essendo indisponibili, qualsiasi contratto o dichiarazione intesa a spogliarsene o a promettere di non avvalersene non ha effetto, se non per quelle modifiche che l'autore ha conosciuto e accettate.

Ma tolte queste parti marginali, la CC0 è oggi lo strumento giuridico più affidabile per chi voglia completamente disfarsi in maniera pressoché definitiva di ogni diritto sulla propria creazione e consentirne ogni scempio. La CC0 rispetta la definizione di Free Culture.

È utilizzata anche per il software. Ma attenzione! Di tutti i diritti di cui ci si disfa con la CC0, uno di essi passa intatto dal filtro purificatore della nostra non-licenza: **i brevetti**.

Perché tale esclusione? Che io sia dannato se lo so. Chiedete a loro. Seriamente, il testo non è chiarissimo, afferma:

> No trademark or patent rights held by Affirmer are waived, abandoned, surrendered, licensed or otherwise affected by this document.

Il che sembra voler dire che tali diritti possono essere sempre utilizzati anche per l'oggetto della rinuncia, anche se sembra un controsenso quanto meno per i brevetti (per i marchi, invece, ha senso, ne ho già parlato nel capitolo d'esordio, i marchi non interferiscono con la creazione di commons intellettuali!).

5.8 Come applicare la licenza a un contenuto[6]

La prassi più diffusa e consigliabile è quella di aggiungere un chiaro disclaimer con il nome esteso della licenza e l'indirizzo web in cui è disponibile il testo integrale della licenza. Nel caso di opere in formato digitale e diffuse tramite internet il tutto risulta particolarmente facile, dato che è sufficiente aggiungere una nota nella pagina web in cui "risiede" il file dell'opera creativa. Il sito ufficiale di Creative Commons (come già accennato) offre un utile widget che, attraverso una serie di domande, guida l'utente nella scelta della licenza più opportuna e genera automaticamente il codice HTML con il disclaimer e il link alla licenza. Ancora, come già accennato, il codice fornito da Creative Commons ha anche la funzione di metatag, cioè inserisce nel codice sorgente della pagina web delle informazioni aggiuntive sul tipo di licenza scelta ma anche sull'autore e sul tipo di opera; queste informazioni, rispettando gli standard del cosiddetto "web semantico", permettono ai motori di ricerca di reperire più facilmente ed efficacemente le opere (HTML-RDF).

Se invece l'opera viene distribuita su supporto fisico, il disclaimer può essere apposto dove normalmente si trovano i dati di edizione e produzione dell'opera; per esempio nel colophon di un libro, nel booklet di un CD musicale, nella cover di un DVD video.

5.9 Conclusioni

Le Creative Commons sono le licenze dominanti dei contenuti creativi. Hanno assunto una forza tale che sono il golden standard per ogni rilascio pubblico,

[6] Paragrafo adattato da Simone Aliprandi, Le licenze open content: capirle e usarle correttamente http://aliprandi.blogspot.it/2015/11/licenze-open-content-capirle-usale.html, CC by-SA 4.0.

tanto che sono usate dai più affermati servizi online: da Flickr (foto) a Wikipedia, da OpenStreetMap a La Stampa, da Jamendo (musica) a Vimeo (video) ed è possibile ricercare contenuti espressamente sotto tali licenza su svariati servizi di ricerca, incluso Google (e ovviamente, con un meta-crawler, su Creative Commons[7]).

Per approfondimenti consiglio la lettura di un manuale per non addetti ai lavori (legali), scritto da Simone Aliprandi e rilasciato, *ça va sans dire!* sotto licenza Creative Commons: Creative Commons: manuale operativo[8]. Oppure, se preferite un video a un testo, il buon Simone ha realizzato anche un'utile e godibile videolezione[9].

Avete molto da vedere, ascoltare, usare. Buon divertimento!

[7] http://search.creativecommons.org/

[8] http://www.aliprandi.org/manuale-cc/

[9] http://aliprandi.blogspot.it/2015/11/creative-commons-introduzione-andrialearning.html

Capitolo 6

Standard e open standard, il diavolo si annida nei dettagli

> "*Sono uomo di mondo, ho fatto tre anni di militare a Cuneo*" – Principe Antonio De Curtis (in arte Totò)

Come il più noto commediante italiano, anch'io ho fatto la mia parte per servire la Patria in quel della Provincia Granda, anche se solo per qualche settimana. Lì ho imparato che senza gli standard[1], in ambito militare le cose si possono fare veramente difficili. Cominciamo con "*Alfa, Bravo, Charlie, Delta, Echo, Foxtrot, Golf, Hotel, India, Juliett, Kilo, Lima, Mike, November, Oscar, Papa, Quebec, Romeo, Sigma, Tango, Uniform, Victor, Whisky, X-ray, Yankee, Zulu*". È l'ABC degli standard, anzi, lo standard dell'ABC. È l'alfabeto fonetico, standard NATO. Se non lo conosci a memoria, non puoi comunicare via radio; così è per quasi tutto quanto accade nella vita militare: se sei un militare non hai spazio per ambiguità. Dai gradi (per sapere chi comanda su chi) fino al calibro delle pallottole, tutto è standard.

"Standard", effettivamente, viene dal francese antico "*estandart*" o "*estandard*", stendardo, bandiera, che in effetti richiama i campi di battaglia e gli eserciti schierati in file ordinate. Ma sbaglieremmo a pensare che solo in ambito militare esistono standard. Come misuriamo il tempo dipende da uno standard, come *scriviamo* il tempo dipende da uno standard. Come misuriamo qualsiasi cosa dipende da uno standard. È importante ovviamente che le misure siano conosciute e che qualsiasi necessità di conversione tra uno standard e l'altro sia precisa e

[1] Per ulteriori letture in tema di standard, vedi i vari contributi su Techeconomy da parte, tra gli altri, di Italo Vignoli http://www.techeconomy.it/author/italo-vignoli/. Vedi anche di Simone Aliprandi "Apriti Standard" http://www.aliprandi.org/apriti-standard/

dichiarata, altrimenti succedono disastri. Se non dico che l'ora in cui fisso una conferenza telefonica corro al massimo il rischio di perdere qualche partecipante; ma se non dico che una pressione è in libbre per pollici quadrati e l'altro pensa che sia (come dovrebbe) in kilogrammi per metro quadro, succedono disastri ben peggiori. Nel 1999 il Mars Polar Orbiter si schiantò al suolo perché alcuni dati di volo vennero inviati in libbre/secondo invece che in newton/secondo, buttando a mare quasi 330 milioni di Dollari USA. La stessa cosa capiterebbe anche a noi se nell'aviazione non ci fossero centinaia di standard, da quelli di sicurezza a quelli sulla navigazione e ai sistemi di identificazione e atterraggio.

Quasi tutto ciò che facciamo, in realtà, si basa sugli standard. È standard la presa con cui colleghiamo il computer alla corrente, è standard la corrente, è standard la codifica dei file, è standard il cavo di rete, è standard la presa di rete (o se usiamo il Wi-Fi, è standard pure quello), è standard il protocollo che stabilisce la connessione e assegna l'identificativo Internet; tutta Internet, da un punto di vista logico, non è che una somma di standard. È standard il formato di file con cui scriviamo il documento che contiene questo articolo. Se guardiamo la televisione, via satellite (DVB), via satellite (DVBT) è tutto uno standard; se guardiamo un filmato su Youtube, pure quello è uno standard (anzi due, uno per il video e uno per l'audio, anzi, tre, uno per il contenitore, WebM, anzi quattro, perché il se e il come il filmato viene mostrato dipende da un altro standard, HTML).

6.1 C'è standard e open standard

Chi fa gli standard? Fornire una risposta a questa domanda è praticamente impossibile. Abbiamo standard formali e standard informali (la posta elettronica e molti degli standard di Internet sono delle RFC, mai adottati formalmente e gestiti in larga parte dall'IETF[2]). Esistono poi enti di standardizzazione formale "istituzionali" a livello nazionale (ad esempio UNI in Italia, DIN in Germania, BSI nel Regno Unito, ANSI negli Stati Uniti), a livello europeo (ETSI, CEN), e internazionale (ISO, ITU, IEC). E poi vi sono quelli su base consortile (OASIS, ECMA, Bluetooth, W3C). Un bel guazzabuglio.

Ma a noi, in questa sede, non importa più di tanto come gli standard sono formati e da chi (se non quando succede un problema, come vedremo poi con il famigerato OOXML). Ci importa sapere **cosa accomuna** gli standard. Tutti gli standard, ufficiali e formalizzati (de jure) o non ufficiali e informali (de facto) si caratterizzano per essere una **norma**, e infatti un sinonimo dell'attività di creazione degli standard è "normazione" e così si specificano gli enti che se ne occupano. Si tratta di un norma di per sé **non cogente**, se non quando vi fa espresso riferimento una legge o un atto normativo dello stato o internazionale. Gli standard vengono rispettati perché e in quanto sono diffusamente adottati e non adottarli comporta problemi difficilmente insuperabili. La norma diventa

[2] https://ietf.org/

"cogente" se non posso fare altrimenti che seguirla, salvo trovarmi a mal partito (o peggio, violare la legge). Se entro in un ristorante francese e non so il francese, probabilmente non mangio, o corro il rischio di mangiare cose che non volevo.

Norma, dicevamo, dunque **regola**. Chi di solito fa le regole? Come nello sport, non è mai uno dei giocatori, o almeno non dovrebbe. E come seguire le regole se non le conosco? E mettiamo che le conosco, ma se una volta che le ho imparate, mi si cambiano le carte in tavola, e io non faccio in tempo a prepararmi, ma qualcun altro sì, perché lo sapeva prima? E se ancora, si impone di usare un pallone speciale, ma io non posso allenarmi con quel pallone, salvo poi trovarmelo in campo contro le altre squadre?

Le regole, devono essere fatte in un certo modo, altrimenti qualcuno è nei guai, e qualcun altro ci marcia. Entrano in campo gli standard veramente **imparziali** e non distorsivi, ovvero gli standard aperti. Siccome ci piace usare l'inglese "**Open Standard**".

6.2 Standard Aperti

Non esiste una definizione di open standard. Quando si cerca di adottarne una, ci si trova un fuoco di sbarramento fatto di lobbying selvaggio, *disinformazia*, agenti doppi, non un quadro edificante. Lo dico per esperienza personale, per esserci entrato in occasione di OOXML (ci arriviamo, ci arriviamo…) e in occasione di vari dibattiti in cui si è tentato di trovare una definizione efficace, come nel caso dell'European Interoperability Framework, che in effetti adottò una formulazione quasi accettabile di open standard, e infatti quando si è passati alla versione 2, quella parte è stata rimossa perché contro di essa si sono mossi mari e monti.

Ma chi è intellettualmente onesto non può che rinvenire alcuni tratti fondamentali su cosa definisce uno standard aperto. A partire dal fatto che uno standard aperto è… aperto alla sua adesione da parte di tutti e non crea indebiti vantaggi o posizioni di dominio da parte di qualcuno su qualcun altro. Qui do alcune indicazioni su ciò che uno standard aperto debba rispettare, nella definizione che ho contribuito a fissare per FSFE

6.3 Uno standard è aperto quando è accessibile

Questa è facile. Lo standard è una norma, la norma deve essere **conosciuta** per essere osservata. Lo standard deve essere dunque pienamente conoscibile. Per cui deve essere compiutamente documentato. Gli standard non debbono completamente documentare tutto, ma quello che non documentano deve essere facilmente derivabile da altri standard. Anzi, gli standard ***devono* riutilizzare**

gli altri standard, dove possibile, e non inventare nuovi modi per parti di esso che siano già standardizzate.

Corollario a questo principio è che uno standard **non può fare riferimento** a un prodotto, un servizio una tecnologia non standard, o peggio, **proprietaria**. Uno standard che dovesse indicare "fai questa cosa qui come la fa l'applicazione X del produttore M" non sarebbe uno standard aperto, e non sarebbe uno standard completo.

Tali documenti, poi, debbono essere resi "**pubblici**". Ciò non significa che essi non possono essere coperti da copyright e ceduti sotto condizioni proprietarie: tali condizioni non debbono essere **discriminatorie** o eccedere i costi di formazione dello standard e di produzione del documento, ovvero costi nominali dell'opera. Se munirsi di una copia dello standard dovesse costare eccessivamente, solo chi ha sufficienti disponibilità economiche potrebbero accedervi. Se solo chi ha una qualifica particolare può accedere il documento, questo sarebbe discriminatorio. L'accesso pubblico significa "a chiunque sia disposto a pagare una ragionevole somma, non eccessiva, se richiesta, e senza che qualcuno possa opporre un rifiuto".

6.4 Uno standard è aperto quando è gestito imparzialmente

La partecipazione alla formazione degli standard da parte delle imprese e degli altri operatori interessati è una elemento essenziale nella formazione degli stessi. Solo chi conosce un campo di applicazione può avere idea di cosa può essere standardizzato e come. Di solito si adottano le scelte operative più intelligenti dei primi che hanno affrontato il problema. Il principio base è dunque quello del raggiungimento di un **consenso** tra tutti gli operatori.

Uno standard "buono" è normalmente uno standard formato democraticamente, in cui tutti gli interessati hanno avuto modo di dire la loro e nessuno prevale in modo abnorme sugli altri partecipanti. In realtà questa non è una regola necessaria. Esistono buoni standard in cui la tecnologia è stata fornita unilateralmente da un operatore, il quale la ha compiutamente descritta in maniera standard (esiste uno standard su come si scrivono gli standard) e ha concesso a tutti il permesso di usare tale tecnologia. È il caso del PDF, che è stato "donato" da Adobe e formalizzato in uno standard ISO (ISO 19005). Qui la genesi non è certo imparziale, ma una volta che il formato è stato proposto come standard aperto, la conduzione ulteriore dello stesso è affidata a comitati tecnici in sede ISO, la cui partecipazione – come per tutti i comitati tecnici di ISO – è aperta a tutti.

Dunque, originalmente o successivamente a una "donazione", lo standard è affidato a un ente imparziale, non legato o dominato da un attore dominante, ad accesso democratico.

6.5 Uno standard è aperto quando non discrimina

Uno standard dovrebbe essere il tecnologicamente neutro possibile, ovvero non privilegiare ingiustificatamente una piattaforma rispetto a un'altra, una tecnologia rispetto a un'altra, un'impresa piuttosto che un'altra, per quanto possibile. Dunque dovrebbe osservare un principio di prudenza nel non prevedere l'adozione di tecnologie esterne e non standard, o peggio, la necessità di ottenere il permesso da qualcuno per utilizzare tutta o parte della tecnologia necessaria.

Condizione necessaria e sufficiente per implementare uno standard dovrebbe essere necessario unicamente conoscere lo standard (e gli standard di riferimento sui cui lo stesso si basa). Il resto dovrebbe essere solo "*delivery*",

6.6 Standard e brevetti: questo matrimonio non s'ha da fare (rinvio)

Siccome uno standard diventerà una norma, e soprattutto negli standard tecnologici più uno standard è utilizzato, più tende ad essere utilizzato (effetto di rete) a prescindere anche da quanto merito tecnico esso abbia, sono ovvie le interazioni tra standard e **concorrenza**. Pertanto, per quanto possibile, gli standard dovrebbero essere privi di condizioni legali che tendano a privilegiare le offerte di alcuni a scapito di altri, e soprattutto dovrebbero evitare quella che si chiama "*patent holdup*", ovvero la posizione di supremazia di chi ha brevetti essenziali per l'implementazione di uno standard.[3]

E qui, sovente, casca l'asino. In difetto di una precisa politica che consenta l'utilizzo degli standard a tutti, indipendentemente dal modello di business e di licenze, occorre che vi siano chiare regole sia per chi contribuisce agli standard (dichiarare l'esistenza di propri brevetti, impegnarsi a licenziarli sotto determinate regole), sia per chi approfitta delle lacune o delle imprecisioni degli enti di standardizzazione per porsi in posizione di controllo, spesso identica a quella dei famigerati *troll*.

[3]Per un'ottima analisi, si veda Dolmans, Maurits (2010) 'A Tale of Two Tragedies – A plea for open standards, and some comments on the RAND report', IFOSS L. Rev., 2(2), pp 115 – 138 DOI: 10.5033/ifosslr.v2i2.46 (http://dx.doi.org/10.5033/ifosslr.v2i2.46) con una mia introduzione.

Questo è un campo di indagine molto complesso, che sospendiamo solo momentaneamente.

6.7 OOXML e altri orrori

OOXML è lo standard documentale XML spinto (per usare un eufemismo) da Microsoft. La storia è molto lunga ed è anche in certo modo la *summa* di ciò che non si dovrebbe fare in una standardizzazione. Lo standard serve a codificare i documenti di Microsoft Office in formato XML (altro standard).

Lo standard soprattutto evidenzia difetti in tutte le aree di cui ho parlato sopra, forse (e dico *forse*) con l'unica esclusione dell'interferenza di brevetti.

Andiamo in ordine sparso. Il primo peccato capitale è che si tratta di uno standard che viene adottato su proposta di una e una sola azienda, senza che vi sia una che sia una implementazione di esso che sia in qualche modo completa. E ciò non lo è stato per diversi anni. Soprattutto quando esiste un diverso standard (ISO IEC IS 26300 – ODF) per esattamente lo stesso dominio. Le buone pratiche vorrebbero che Microsoft si mettesse in gioco e spingesse perché tale standard evolva fino a coprire le esigenze non coperte dallo stesso. Invece lo rimpiazza con un secondo standard completamente nuovo.

Lo standard poi si limita a riflettere pari-pari il comportamento delle applicazioni di Microsoft, non a descrivere le funzioni in astratto e a risolverle in modo astratto e imparziale. Arriva persino a codificare gli errori, fenomenale fu quello di ripetere l'omissione dell'anno non bisestile negli anni zero dei secoli non divisibili per quattrocento, che è la modalità in cui si contano le date nel calendario Gregoriano (che è anch'esso uno standard: ISO 8601.[4] Ciò solo per un *bug* mai corretto di Microsoft Excel.

Lo standard venne proposto con una procedura accelerata ("*Fast Track*") nonostante ci fossero molte perplessità sull'esperibilità di tale procedura e in vari enti nazionali (ISO è un ente "federato" in cui partecipano gli enti nazionali, in Italia l'ente nazionale competente era Uninfo, in seno a UNI) si sollevarono centinaia di osservazioni, che vennero risolte in modo molto grossolano in un "*Ballot Resolution Meeting*" in quel di Ginevra, segno che lo standard doveva essere approvato più o meno ad ogni costo. Le stesse procedure nazionali lasciarono molti perplessi, me compreso. In Italia, ad esempio, nei mesi precedenti alla votazione, si osservò un'impennata di iscrizioni ad Uninfo a cui molti hanno attribuito un significato preciso: quello di un tentativo di modificare gli equilibri in gioco per far passare (o non passare, ma in misura assai minore) lo standard, al di là del suo merito tecnico. Lo standard stesso era composto da circa 6000 pagine, che probabilmente nessuno dei partecipanti (se mai qualcuno) ha letto integralmente.

[4] Per un'interessante disamina, se interessati, si può consultare la pagina di Wikipedia https://it.wikipedia.org/wiki/ISO_8601.

Ma al di là di tali episodi, su cui ognuno manterrà le proprie opinioni, è il concetto stesso che si possa standardizzare a partire da un'applicazione singola per creare uno standard che non è condiviso, ma replica in maniera ossessivamente pedissequa ciò che fa un'applicazione, a detrimento di tutte le altre, a fare a pugni con la stessa concezione di standard. Il fatto che il principale sponsor di quello standard fosse una società che proprio in quel periodo era stata condannata per condotte abusive esattamente per la creazione di standard di fatto usati in modo anticoncorrenziale, anche se in ambiti non sovrapponibili, lascia alquanto perplessi circa tale standard.

Non che gli altri siano indenni da critiche. Lo vedremo nel prossimo capitolo.

Capitolo 7

Open standard e brevetti

Con l'ipertrofia dei brevetti originatasi soprattutto negli Stati Uniti al volgere del millennio, e poi importata a rimorchio in Europa, ormai è pressoché impossibile realizzare una qualsiasi applicazione tecnologica senza violare un qualche tipo di brevetto. Non importa se si tratti di uno standard di comunicazione, di uno standard sui formati, di uno standard multimediale, ogniqualvolta si tratti di normare un settore tecnologico, l'interferenza con (spesso molti) brevetti è inevitabile.

È un problema? Sicuramente lo è. Per capire meglio di cosa si tratti, facciamo un piccolo passo indietro e interroghiamoci a cosa servano i brevetti.

7.1 La funzione dei brevetti nell'economia

7.1.1 La teoria economica fondamentale

I brevetti vengono riconosciuti come un **incentivo all'innovazione**. La teoria economica è sufficientemente chiara e semplice. Tizio inventa qualcosa, ci mette anni e anni, investe ingenti capitali in ricerca e sviluppo, va sul mercato. Il suo prodotto, costa dieci euro. Ma se lo vende a dieci euro e dieci centesimi, cosa ne è di tutto l'investimento fatto? Per cui ammortizza il costo su un numero sperato di prodotti venduti. Il prodotto dunque esce a quindici euro.

Caio vede il prodotto di Tizio, pensa che sia una bella invenzione. Ne compra uno, lo osserva, capisce come è fatto e decide che anche lui è in grado di fare la stessa cosa. Il prodotto gli costa dieci euro, esattamente come a Tizio. Lo vende a undici, guadagna uno. Il prodotto di Tizio è fuori mercato. Allora, vista la concorrenza, Tizio abbassa il costo, ma anche così lui vende solo la metà dei prodotti che sperava, da un certo punto in poi, e per giunta con un margine

molto inferiore. Tizio perde, Caio guadagna sulle spalle di Tizio. Sapendo di correre questo rischio, si dice, è probabile che Tizio nemmeno si metta a investire. È un **gioco somma negativa**. Non solo per Tizio e per Caio, ma per tutto il sistema, che così non riceve i benefici delle possibili invenzioni che così non verranno immesse sul mercato.

La soluzione trovata a questo dilemma è stata sinora quella di garantire a chi dimostri di aver dato un contributo all'avanzamento della tecnica, ovvero aver creato e concepito qualche idea che prima non esisteva, un **limitato monopolio** sull'utilizzo di quell'idea sul mercato. Il monopolio gli consente di estrarre la rendita del monopolista, sia direttamente, sia vendendo il permesso di praticare la stessa invenzione.

7.1.2 La limitazione della protezione è funzionale allo sviluppo della tecnologia nel settore protetto.

Il monopolio è limitato in due sensi: non si brevetta un intero prodotto (come comunemente si tende a pensare), ma solo sulla parte innovativa di esso, sull'idea che non esisteva, è dunque limitato **nell'oggetto**; inoltre, è limitato nel **tempo**: la privativa dura vent'anni, poi l'idea diventa "**di pubblico dominio**" e chiunque può praticarla commercialmente. Questo connubio tra la fase di privativa e la fase di pubblico dominio è importante, senza l'una non potrebbe esistere l'altra. Per ottenere questo risultato, una delle condizioni per ottenere il brevetto è quella di **compiutamente descrivere** l'invenzione, cosicché un esperto del ramo sia in grado di riprodurla solo leggendone la descrizione. Brevetto e **segreto** sono **incompatibili** perché l'invenzione deve entrare nello **stato della tecnica**.

La limitazione temporale è chiara, ma anche quella sull'oggetto lo è, nella teoria economica. Concentrandoci solo sulla parte che è rilevante al nostro discorso, il fatto che la protezione sia limitata solo alla parte strettamente necessaria a praticare in pace la propria invenzione e nulla più, ovvero non sia debordante ("*overbroad*") serve allo scopo di **incentivare l'ulteriore innovazione**. Se il primo arrivato fosse in grado di ottenere il monopolio su tutto ciò che ha a che fare con la propria invenzione, ben difficilmente un secondo inventore investirebbe nello stesso campo, sapendo di essere alla mercè dell'arbitrio del monopolista. [1]

Il gioco concorrenziale che (teoricamente) si crea è anch'esso (sempre in teoria) semplice. Chi volesse fare concorrenza a Tizio ha due strategie fondamentali: pagare il permesso (= licenza) o **trovare un altro modo** di praticare la stessa invenzione ("***to invent around***"). Se Tizio chiede troppo come compenso per

[1] Cosa che accadde ed è molto studiata nel caso dei brevetti di Watt sulla macchina a vapore. Per approfondimenti, si veda il lavoro di Michele Boldrin e David K Levine "Against Intellectual Monopoly" Cambridge University Press 978-0-521-87928-6. L'introduzione è disponibile tramite il MISES Blog [2]

praticare la propria invenzione, Caio sarà incentivato a trovare una soluzione diversa tra le varie possibili, e dunque creerà magari un modo più intelligente di risolvere lo stesso problema. In questo modo, si lascia spazio al **mercato**, e all'ulteriore innovazione, per cui Tizio tenderà a chiedere un prezzo ragionevole per la licenza; se non lo farà, Caio avrà un'alternativa praticabile, aggirando l'ostacolo della privativa, e in ogni caso si incentiva l'incremento tecnologico. Il gioco qui è a somma positiva e tende a calmierare i prezzi e a incentivare l'ulteriore ricerca.

Teniamo a mente il concetto di *invent around*, che ci servirà nella discussione più oltre.

7.1.3 Il cross licensing, il primo fallimento

La teoria economica classica dei brevetti va avanti con una possibilità ulteriore. Poniamo che Tizio e Caio si accordino. Ancora, Caio sarà incentivato a innovare ulteriormente, perché se da un lato ora può praticare l'invenzione che gli serve, dall'altro ogni volta che vende fa un favore anche a Tizio, che guadagna con la vendita di entrambi. Se però Caio riesce a trovare un modo più raffinato di utilizzare l'invenzione originale ("**invenzione di sviluppo**") egli potrà a sua volta ottenere un brevetto su quella parte innovativa a valle, e praticare in esclusiva, a parti invertite con Tizio.

A questo punto un terzo che dovesse cercare di entrare sul mercato, dovrebbe chiedere il permesso sia a Tizio che a Caio.

Ma tra Tizio e Caio, cosa succede? Molto probabilmente essi si metteranno d'accordo affinché, invece di pagarsi le licenze l'un l'altro, si paghi solo la differenza tra quello che l'uno deve all'altro e viceversa. Anche stabilire quanto Tizio debba a Caio per ciascun prodotto è difficile: vale di più l'invenzione di uno o quella dell'altro, visto che entrambe sono necessarie e non sufficienti a costruire il prodotto più avanzato? Molto spesso Tizio e Caio si mettono d'accordo per non "pesare" i rispettivi brevetti, semplicemente si dicono che i portafogli di entrambi valgono uguale e amici come prima. Questo sistema si chiama ***cross licensing***.

Tutti contenti? Il terzo non è contento, perché a questo punto egli ha due concorrenti che hanno i brevetti, e lui no. Non avendo brevetti, egli non avrà moneta di scambio, non potrà entrare nel sistema del *cross licensing*. Si è creata una situazione che tende all'oligopolio, un club in cui chi è dentro tende a tenere fuori chi non lo è. Certo, anche il terzo operatore potrà mettersi di buzzo buono e inventare qualcosa di nuovo, o di cercare di aggirare i brevetti, ma più questo gioco va avanti, meno saranno le possibilità di entrare nel club, soprattutto perché i primi brevetti tendono a essere i più generici, quelli successivi ad avere qualche tipo di interferenza con i primi (più ad essere di sviluppo, che totalmente alternativi).

Insomma, si creano barriere all'entrata.

7.2 I brevetti tecnologici

Nella tecnologia moderna il susseguirsi di innovazioni è sempre più tumultuoso, e ogni campo tende a essere **denso di brevetti**, un vero e proprio *campo minato*. Allo stesso tempo, l'innovazione sempre più spinta tende a rendere **obsoleti** i principi poco tempo prima ritenuti innovativi.

Da un primo punto di vista questa situazione crea un primo rilevante fallimento, quello di non avere più uno spazio di esercizio del **pubblico dominio**, che diventa privo di senso: un'invenzione di vent'anni prima difficilmente sarà da sola sufficiente ad essere praticata.

Il protrarsi della protezione porta anche a un secondo, *paradossale*, effetto: quello del **disincentivare** l'innovazione. Riconoscendo una posizione di rendita ai primi arrivati, in quanto i loro brevetti sono fondamentali per tutti gli altri, per costoro è meno conveniente concentrarsi sull'innovazione, che comunque aggiungerà poco e costerà tanto, rispetto a concentrarsi sulla massimizzazione dei proventi ottenuti dalle licenze, che non richiede più investimenti. Ciò fa sì che i primi siano in posizione asimmetrica avendo un maggior potere interdittivo generale rispetto ai nuovi venuti, coloro che hanno i brevetti più innovativi (perché più recenti) ma anche più specialistici e limitati nello scopo, se pur non necessariamente meno importanti o costosi da ottenere.

Nella tecnologia avanzata, infine, si tende a moltiplicare all'infinito la complessità della rete di brevetti che insistono su ogni minimo componente tecnologico, tanto da creare quello che viene definito un "roveto di brevetti" ("***patent thicket***")[3], in cui non è più tanto importante la qualità dei brevetti che hai, ma la quantità di essi, il fatto di possedere quella massa critica che ti fa arrivare al tavolo di negoziazione per avere un *cross licensing* alla pari con gli altri.

Si è da varie parti evidenziato, infine, come nel software la parte inventiva sia ridicolmente meno importante della parte di esecuzione (sviluppo, testing, supporto), nel valore delle singole applicazioni, e spesso i brevetti sono un artefatto postumo, ricavati *ex post* dal realizzato, senza precisi e separati sforzi di **ricerca**, ma solo di **sviluppo**, per giunta non dedicati tanto all'innovazione, quanto all'applicazione in sé, parti innovative come parti non innovative.[4]

[3] La parola "thicket" indica una zona impenetrabile a causa di una crescita disordinata ed eccessiva di vegetazione.

[4] Una posizione simile è quella di Richard Posner, Scuola di Chicago, dunque liberista ed estremamente *pro-market*, in *Do patent and copyright law restrict competition and creativity excessively? Posner*, Becker-Posner Blog (http://www.becker-posner-blog.com/2012/09/do-patent-and-copyright-law-restrict-competition-and-creativity-excessively-posner.html)

7.3 Entrino gli standard (nel software)

Se nella teoria classica il concetto di *invent around* è importante, nella realtà la possibilità di ricorrervi è molto minore di quella teorica. Cosa succede se tale possibilità è **inesistente**? Se vi fosse un impedimento assoluto ad aggirarli, non avremmo dato un eccessivo potere ai singoli brevetti? Cosa sarebbe della limitata **concorrenza** che il sistema brevettuale pur concede, al fine di non concedere la possibilità di innovare solo al primo venuto? La teoria economica alla base dei brevetti, verrebbe di sicuro a cadere.

Questa situazione avviene quando una tecnologia brevettata entra in uno **standard**.

Una volta che uno standard è stato creato e diventa di fatto inevitabile, ecco che la teorica possibilità di aggirare l'ostacolo viene meno. L'alternativa è infatti molto poco appetibile, e cioè ignorare lo standard, o crearne uno alternativo. Ma se i grandi operatori si sono messi d'accordo per seguire quello standard, la *chance* che un piccolo operatore, magari enormemente innovativo, possa farci qualcosa è spesso minima. Non tutti possiamo avere la *grandeur* che faceva dire ai Britannici, all'epoca dell'Impero:

> "la Manica è in burrasca oggi, il Continente è isolato".

7.3.1 Il gioco degli standard e dei brevetti "necessariamente violati"

Se non puoi passargli sopra (aspettare che scadano) o girargli intorno (trovare strade alternative), ecco che concedi a chi ha brevetti che sono "**necessariamente violati**" da chi implementa lo standard una posizione di vero e proprio **monopolio assoluto** non mitigato da una limitata concorrenza tra monopoli.

In passato ci si è posti un serio problema di **antitrust** nei confronti dei partecipanti alla standardizzazione e agli stessi enti standardizzatori. D'altronde è facile intravedere nella ricerca di un *consenso* tra concorrenti (gli standard formali richiedono il consenso degli *stakeholder*) la possibilità di un accordo per sfruttare in modo anticoncorrenziale l'esistenza di brevetti. Tra un accordo, di per sé del tutto legittimo, di standardizzazione e un **cartello** la differenza è minima. Per quanto riguarda i brevetti, tale differenza sta nel fatto che i partecipanti allo standard si impegnino più o meno formalmente a concedere una licenza sotto condizioni ragionevoli e non discriminatorie, ovvero, con un acronimo abbastanza conosciuto **RAND**.

7.3.2 Le licenze RAND e i *patent pool*

Chi frequenta i trasferimenti di tecnologia e gli accordi di licenza sugli standard, conosce assai bene sia il termine "RAND" (frequentemente "FRAND") e "*patent pool*". Cominciamo con il primo termine.

(F)R ((Fair), Reasonable) :Le condizioni di licenza debbono essere ragionevoli, ovvero non richiedere un compenso eccessivo rispetto al "valore" del singolo brevetto o del portafoglio di brevetti che si ottiene. Questa valutazione deve essere effettuata *ex ante*, non tenendo conto del valore strategico che le privative licenziate hanno *in quanto* inserite nello standard (c'è chi dissente). "Fair" e "Reasonable" sono largamente sinonimi.

A (And) :Entrambi i requisiti legati da "and" debbono essere presenti.

ND (Non Discriminatory) :Le condizioni debbono essere offerte a **chiunque** e in modo equanime, **senza favorire nessuno**. Un'eccezione sovente praticata è che chi fa parte del *patent pool* (perché licenziante) non sia soggetto al pagamento di *royalty*.

Se il principio di non discriminazione dà luogo a pochi dissidi, il concetto di "ragionevole" è fonte sovente di diverse interpretazioni. Va ricordato che le regole sulle licenze dei brevetti "vincolano" (attraverso varie teorie, che omettiamo per brevità) il titolare dei brevetti a rispettare la sua dichiarazione. Ma la genericità di tale affermazione è tale da frustrare sin dall'inizio ogni velleità di contestarla, se non in alcuni casi. [5]

Il **fallimento** di tale gioco può avvenire però quando uno o più titolari di brevetti non siano, almeno formalmente, coinvolti nelle attività di standardizzazione, e dunque non risultino vincolati dalla dichiarazione effettuata all'ente standardizzatore. Nel caso Rambus,[6] la società è stata accusata di "*patent ambush*", ovvero di aver teso un'imboscata, facendo sì che lo standard implementasse una sua tecnologia, per poi imporre *royalty* eccessive per il permesso di praticarla, senza che contro di essa fosse invocabile un obbligo RAND. Ma anche nel caso in cui un soggetto abbia effettivamente sottoscritto le condizioni RAND, vi sono spazi per abusi. Un caso piuttosto chiaro (e che richiama il caso Apple v. Motorola) mi è capitato recentemente.

Un noto standard, in cui si è formato un altrettanto noto *patent pool*, licenzia più di un migliaio di brevetti per una cifra poco più che simbolica (pochi centesimi a copia, con un'esenzione per un primo quantitativo annuale di copie); un noto operatore ha chiesto a un mio cliente il pagamento di una somma non molto differente per meno di una decina di brevetti. In difetto di una prova convincente,

[5] Esemplare il caso Apple v. Motorola. Una buona fonte per approfondimenti è la pagina di Wikipedia: https://en.wikipedia.org/wiki/Motorola_Mobility_v._Apple_Inc. Un'altra buona fonte è, a cura della Direzione generale antitrust della Commissione Europea, Standard-essential patents, in *Competition Policy Brief*, Issue 8, June 2014 http://ec.europa.eu/competition/publications/cpb/2014/008_en.pdf

[6] Vedi il già citato *paper* della Commissione nelle note precedenti.

è abbastanza difficile pensare che quei brevetti valgano ciascuno più di quanto valgano in aggregato i cento brevetti più importanti conferiti nel *patent pool*, anche a voler pensare che il 90% di essi sia *fuffa*.[7] L'obiezione che i membri del *patent pool* possano essere filantropi e aver licenziato i propri brevetti sottocosto sembra ben poco valida: proprio il fatto che si sia creato un *patent pool* che ha calmierato le pretese di ciascuno (abbia impedito il ***royalty stacking***) ha probabilmente decretato il successo dello standard (di tale fenomeno si discute in Apple v. Motorola).

Come sia conclusa la vicenda parrebbe ovvio: irretito dalle mie puntuali obiezioni, convinto dall'irrefutabilità dei riferimenti alle norme e alle sentenze, il noto operatore ha receduto dal suo proposito e si è ritirato in buon ordine, accontentandosi di briciole, giusto?

Sbagliato. Il titolare non ha nemmeno fatto lo sforzo di replicare, ben sapendo che nessuno sano di mente avrebbe speso quello che avrebbe potuto costare una causa, con la prospettiva di essere coinvolto anche in un'ingiunzione che avrebbero impedito la commercializzazione dei prodotti che implementano lo standard, per risparmiare una somma che può rappresentare una frazione dei costi legali.[8]

7.3.3 Le licenze RAND e le licenze di Software Libero / Open Source

Abbiamo tralasciato per un attimo il discorso della non discriminatorietà. Se Tizio viene e mi chiede una licenza, gliela do. Se Caio viene e mi chiede la stessa licenza, gliela do alle stesse condizioni, non dovrei essere discriminatorio, anche se Tizio ha uno un margine del 30% su ogni vendita, e Caio a malapena fa il *break even* e con la licenza il suo prodotto diventa diseconomico, magari perché i due sono sottoposti a pressioni concorrenziali notevolmente differenti.

Questo è almeno il discorso che viene effettuato tutte le volte che si avanza l'obiezione secondo cui le condizioni RAND, anche qualora prevedano *royalty* molto basse, sono incompatibili con il Software Libero / Open Source. Queste sono licenze pubbliche, in cui non esiste il concetto di "venditore" e "acquirente", per cui è impossibile controllare quante copie del software siano in circolazione.

Ci si dice "è una vostra scelta quella di adottare un modello di licenze incompatibile con la nostra scelta di cosa offrire". Tale affermazione è accettabile se la applichiamo in un modo dove la concorrenza è solo all'interno di un singolo modello di business e laddove tale modello di business sia quello di "vendere copie di software" (se applicato al software). Ma è evidente che in un modo dove il modello di vendita di licenze di software diventa sempre *meno* importante, laddove anzi nel software (e altrove!) il modello aperto è molto spesso la

[7] Termine rigorosamente tecnico

[8] L'imprecisione dei riferimenti e delle cifre è intenzionale, serve solo a dare gli ordini di grandezza, ma i dettagli e le identità sono ovviamente confidenziali.

regola, un sistema di standardizzazione legato a modelli passati e unici risulta discriminatorio.

Sicuramente, ad ogni modo, un sistema che discrimina e impedisce l'accesso alla tecnologia verso una delle strategie di sviluppo e di *licensing* di maggior successo non può dirsi **aperto**, in quanto non può ritenersi aperto e neutrale un sistema che impedisca l'acceso a certe tecnologie e operatori legittimamente sul mercato, imponendo specifiche modalità di business. Un modello di standard che si limiti a prevedere un obbligo di (F)RAND basato su *royalties* e comunque sulla necessità di controllare il numero di copie distribuite è in aperta antitesi con un ampio settore di operatori che non solo legittimamente, ma anche con grande beneficio comune, hanno adottato un modello di rilevante successo, è chiaramente un'esclusione ingiustificata, quando parliamo di standard.

Capitolo 8

Brevetti e software: per chi suona la campana?

Nel capitolo precedente abbiamo discusso di brevetti applicati agli standard (nel campo del software principalmente). Mi chiedo però se il software debba essere oggetto di tutela brevettuale per sempre, oppure ci sia spazio per una loro abolizione o sostanziale restrizione? Già la Corte Suprema degli Stati Uniti ha dato diversi colpi alla lassità con cui le corti federali avevano concesso strada libera alla brevettazione del software e dei modelli di business e praticamente di qualsiasi cosa di cui sotto il sole di agosto qualcuno potesse inventarsi la brevettazione. Le sentenze Bilski, Alice (soprattutto, vedi paragrafo successivo, Mayo, pur non affrontando il cuore del problema hanno però fornito materiale per le corti inferiori per demolire almeno i brevetti più "deboli".

Chiunque esperto nel ramo vi dirà "*software patents are here to stay*" ("i brevetti software son qui per restare"), ma recentemente si è aperto qualche spiraglio per una definitiva risistemazione del sistema brevettuale, che lasci libero da brevetti il software in sé e per sé, dunque non solo limitando e restringendo i requisiti di brevettazione.

La Corte d'Appello per il Circuito Federale ha emesso una sentenza piuttosto interessante nella causa **Intellectual Ventures v. Symantec**, che mi dà lo spunto per discutere perché i brevetti non dovrebbero mai essere concessi sul software, fornendo dunque un'opinione ben più autorevole della mia. La sentenza in cui Intellctual Ventures pretendeva un bel po' di milioni, ha deciso che nessuno dei tre brevetti (software) su cui si basava la pretesa meritava la protezione brevettuale perché si tratta di idee astratte. Ma la parte più interessante, come capita spesso, è nella *dissenting opinion* del Giudice Mayer, il quale chiede che venga suonata la campana a morto per i brevetti software.

8.1 la sentenza e i precedenti

La Corte d'Appello per il Circuito Federale (CAFC) è stata di recente molte volte al centro di decisioni molto importanti riguardo al tipo di protezione da dare al software, come nel caso Oracle v. Google, che almeno apparentemente si scontra[1] con la giurisprudenza e la normativa europea circa il copyright sulle intefacce. Questo caso è in apparenza meno centrale, in quanto per due su tre dei brevetti anche la corte inferiore aveva riconosciuto una non brevettabilità poiché essi contenevano in maniera abbastanza evidente semplici idee astratte di cui si era pretesa e ottenuta protezione semplicemente perché per applicarle c'era di mezzo un computer. Sul terzo aveva *in extremis* rinvenuto una parvenza di brevettabilità e su quel presupposto aveva condannato i convenuti a un lauto risarcimento dei danni per non avere ottenuto una valida licenza.

Sulla base di quanto deciso dalla **Corte Suprema** degli Stati Uniti, nel caso **Alice Corp v. CLS Bank**[2] (e in Mayo Collaborative Servs. v. Prometheus Labs in un campo adiacente quale quello della brevettazione di principi di funzionamento della biologia umana) il risultato appariva sufficientemente scontato per tutti e tre. Come ricordato sopra, la corte di primo grado aveva già invalidato due di essi, ma aveva fatto rientrare dalla finestra uno dei tre brevetti in quanto, nonostante fosse basato su idee astratte, e queste idee erano semplicemente implementate in un computer, si riteneva contenesse comunque materiale sufficiente per risolvere un problema tecnico in modo non ovvio.

Tuttavia la Corte spazza via anche questo residuo:

> Mayo Collaborative Servs. v. Prometheus Labs.,Claims that "amount to nothing significantly more than an instruction to apply [an] abstract idea . . . using some unspecified, generic computer" and in which "each step does no more than require a generic computer to perform generic computer functions" do not make an abstract idea patent-eligible, Alice, 134 S. Ct. at 2359–60 (citations and internal quotation marks omitted), because "claiming the improved speed or efficiency inherent with applying the abstract idea on a computer" does not "provide a sufficient inventive concept.

Nemmeno questo residuo dunque merita attenzione per la CAFC, in quanto – sostiene – pretende di realizzare un incremento di protezione per il sistema antivirus spostando il livello di proteione da un computer a una linea telefonica (e per traslazione a Internet), in maniera del tutto convenzionale e senza aspetti innovativi, pur essendo presente una descrizione di numerosi dettagli tecnici sulla possibile implementazione nella descrizione del brevetto, dettagli che però non rilevano per l'ambito di tutela.

[1] http://arstechnica.com/tech-policy/2016/05/round-2-of-oracle-v-google-is-an-unpredictable-trial-over-api-fair-use/
[2] http://www.scotusblog.com/case-files/cases/alice-corporation-pty-ltd-v-cls-bank-international/

Fin qui niente di innovativo, si tratta dell'applicazione di una serie di sentenze che limitano il ruolo dei brevetti in settori adiacenti al software e come per il software denotati da una certa astrattezza e intangibilità, come i metodi di business o i metodi matematici per massimizzare i risultati economici di transazioni finanziarie tramite algoritmi implementati in computer (Alice) o altri metodi di business (Bilski v. Kappos).

Ciò che è innovativo, che non si era mai visto, è un giudice che si scaglia lancia in resta contro la stessa idea che possano essere concessi brevetti sul software, cosa che la Corte Suprema aveva sempre accuratamente evitato di fare quando è stata richiesta di farlo e poteva farlo.

8.2 *Dissenting* e *concurring opinion*

Le sentenza USA hanno questa peculiarità: sono delle "*opinion*" (opinioni, pareri) in cui i giudici affrontano in maniera molto elaborata e piuttosto dottrinale i punti giuridici discussi, in riferimento ai precedenti. I giudici sono infatti molto attenti a discutere espressamente i precedenti in quanto questi, sulla base del principio *stare decisis* ("ci si attenga ai precedenti"), essi contribuiscono a costituire la legge per tutti i casi simili, principio fondante della *Common Law*, a cui il sistema giuridico USA si attiene. Dipartire senza una valida ragione da precedenti sufficientemente autorevoli costituisce quella che da noi si chiamerebbe una vera e propria violazione di legge.

Le sentenze sono elaborate collegialmente sulla base di una votazione che può essere unanime o con giudici dissenzienti. Sulla base della decisione, uno dei giudici è chiamato a esprimere l'intenzione della corte, redigendo materialmente la parte motivazionale, ovvero l'*opinion*, che ovviamente sarà coerente con la decisione presa. Così nella sentenza si troveranno tutti i punti fondamentali che hanno guidato il giudice e che guideranno altri giudici di quel distraetto ed eventualmente di altri distretti nella soluzione di casi simili.

Il giudice che provvede alla redazione della sentenza si fa interprete della volontà collegiale. Ci possono essere voti contrari; a differenza di quanto avviene da noi, il fatto che ci siano voti contrari (in tutto o in parte) viene reso pubblico. Un giudice dissenziente che si voglia prendere la briga di farlo, può con altrettanta autorevolezza giuridica, anche se senza vincolare i giudici successivi, esprimere la propria opinione affermando le ragioni per il suo voto contrario, in quella che si chiama "*dissenting opinion*" (opinione dissenziente).

La bellezza e stranezza del sistema americano non si ferma qui: anche chi vota a favore può farci conoscere il suo pensiero, pur concordando con la maggioranza ("*concurring*"), onde dare il suo personale contributo e guida ai futuri giudici, offrendo spunti utili a meglio motivare (e agli studenti maggiori oneri di studio…).

Molto spesso le novità più interessanti nella giurisprudenza vengono appunto dalle opinioni dissenzienti e da quelle *concurring*.

8.3 Il caso contro i brevetti software del Giudice Mayer

Mayer si spinge più in là dei suoi colleghi. Dice che è ora di far scendere un pietoso velo sui brevetti che ricadono su "semplice" software che fa riferimento a computer generici, e lo fa sotto due speciali argomenti: il diritto di esprimere liberamente le proprie opinioni (Primo Emendamento alla Costituzione USA); una esenzione "per categoria" del software dall'ambito della protezione brevettuale.

La parte sul **diritto di parola** (*Freedom of Speech*, uno dei più sacri, non solo perché oggetto del primo emendamento), è molto interessante, ma per la sua peculiarità è anche la più legata alle peculiarità della giurisdizione. Il secondo, riguardante l'esenzione del software dalla brevettabilità come categoria in quanto per natura **materia astratta**, invece si applica pari-pari a tutti i luoghi, compreso il territorio europeo, dove i brevetti software sono concessi a piene mani. È importante per la nostra analisi, in generale, perché coglie una caratteristica importante del software: esso è una serie di algoritmi, un'idea astratta, che tecnicamente può essere realizzata saldando circuiti fatti da semiconduttori, scrivendo bit su un supporto magnetico, ma anche prendendo una penna e scrivendo il codice sulla carta. Difficile, lungo, complesso, ma non impossibile. Mentre di un farmaco posso scrivere la *formula* su un pezzo di carta, ma per realizzarlo mi ci vuole un laboratorio. Di un nuovo materiale a memoria di forma posso scrivere il processo per ottenerlo da materie prime, ma mi serve un impianto per crearlo. Di un microprocessore posso disegnare su carta la topografia con gli elementi e le connessioni, ma mi serve una fabbrica da cento milioni di Dollari per crearlo. Il software no, quello che scrivo sulla carta ***è*** l'oggetto della protezione, ***è*** il software.

L'opinion cita e passa in rassegna in maniera rimarchevole praticamente tutte le ragioni che da anni il fronte anti brevetti software va predicando, fronte a cui non è un mistero anch'io mi iscrivo. Non è il primo giudice di corte d'appello a farlo: già Richard Posner[3] aveva sottolineato come il presupposto economico dei brevetti non si combina con settori in cui l'innovazione è tumultuosa ed effimera, e dove il tasso di investimenti in ricerca e sviluppo e trascurabile rispetto al costo di implementazione dell'idea predicata dal brevetto. Ma si trattava di un blog, qui si tratta di una *concurring opinion* in una sentenza di Corte d'Appello che si occupa in via praticamente esclusiva di brevetti.

[3] http://www.becker-posner-blog.com/2012/09/do-patent-and-copyright-law-restrict-competition-and-creativity-excess
html

Il Giudice Mayer individua in modo analitico almeno quattro dei peccati capitali dei brevetti software.

8.3.1 L'ambito di protezione è sproporzionato al valore di ciò che viene "rivelato"

I brevetti offrono, come è noto e anche da noi ricordato, un incentivo a investire per elaborare prima e a rivelare poi un'idea inventiva che va ad accumularsi allo stato dell'arte al fine di avanzare lo stato della tecnica. È un *do ut des*: ti consento di estrarre profitti del monopolista da un settore, perché tu aumenti lo stato delle conoscenze dandoci modo di fare qualcosa che prima non potevamo fare perché ci *insegni* (letteralmente) **come farlo**. I brevetti software **mancano quasi totalmente** di una descrizione di come si mettono in pratica, descrizione che va per forza di cose effettuata con codice sorgente, che però, quando esiste (spesso non esiste nemmeno, vedi punto successivo) non viene affatto rivelato.

I brevetti contengono solo una descrizione funzionale ad alto livello di ciò che l'applicazione deve fare, non un metodo direttamente implementabile da un esperto, che sarebbe possibile con il codice. Non insegna niente di utile. Non dice come tradurre l'idea in pratica. Per dirla in altri termini, **si brevetta il problema, non la soluzione**.

La banalità dei brevetti e il fatto che anche un non programmatore dotato di sufficiente buon senso e conoscenze molto rudimentali potrebbe scriverli, è un elemento molto soprendente per chi si trova a leggerli. Personalmente devo ancora trovare un brevetto software (e ne ho letti molti) che mi abbia sorpreso con una soluzione che – dato il problema da risolvere – costituisse un qualcosa di sorprendentemente non evidente o inarrivabile senza leggere il brevetto, con l'eccezione forse di *alcuni* brevetti sulla compressione audiovideo, che però hanno ampiamente fatto il loro tempo (il resto è molto minestra riscaldata).

8.3.2 L'incentivo all'innovazione viene dato in un momento sbagliato

Questo punto è una conseguenza del precedente, in un certo senso. Poiché non è richiesta alcuna rivelazione di codice sorgente che implementi il trovato brevettato, e il brevetto viene concesso sulla base di una vaga descrizione funzionale, la tutela viene garantita di molto **in anticipo** rispetto a qualsiasi dimostrazione di avere un'implementazione minimamente funzionante. Realizzare un'implementazione funzionante significa utilizzare molto tempo e risorse di programmazione, e niente, in questo passo, viene aiutato dalla presenza di una domanda di brevetto, che solitamente è anche in regime di segretezza,per cui probabilmente al momento dello sviluppo nessuno è nemmeno *cosciente* della sua esistenza. Siccome il sistema premia chi arriva per primo, vi è un forte incentivo ad ottenere il

brevetto ancora prima di aver *iniziato* a creare una tale implementazione. Una volta ottenuto il brevetto, la privativa consente in teoria di lucrare sull'investimento, rilevante, in ricerca e sviluppo (che nel software è sostanzialmente scrittura di codice e prova di corretto funzionamento) **di terzi**, prima ancora di averlo fatto in proprio. Ciò è tra l'altro dimostrato dall'abbondare di "*non practicing entities*", soggetti che non hanno magari mai nemmeno scritto una linea di codice, ma hanno saputo intercettare dove nuove aree di progettazione del software si sarebbero mosse. Dunque l'incentivo non va a chi dovrebbe investire, anzi, sottrae risorse agli investimenti che dovrebbe promuovere.

Se devo muovere una critica al ragionamento, spesso il problema è l'inverso: si fornisce un incentivo quando è troppo tardi, nel senso che il "trovato" è un sottoprodotto di un'attività di programmazione che ci sarebbe comunque stata, qualcosa che è emerso come un "*afterthought*" alla fine del processo di scrittura del codice, senza uno sforzo preciso e finalizzato, e senza che dunque il premio di un brevetto abbia potuto svolgere il suo effetto incentivante, se non come mera possibilità. Un po' come un premio Oscar alla carriera. Esattamente come la possibilità di ottenere profitti dopo la propria morte difficilmente incentiva a creare più opere musicali,letterarie cinematografiche.

8.3.3 Il loro mero numero ne fa un problema in sé

Il Giudice prosegue segnalando come il sistema dei brevetti sul software stia causando da solo la maggioranza dei nuovi brevetti. Il loro numero, da solo, è un problema per l'economia e una minaccia per le imprese innovative più giovani e piccole, ma anche per tutto il settore della tecnologia. Ricorda che il numero di brevetti insistenti su un moderno smartphone sono un numero inimmaginabile, oltre 250.000. secondo alcuni calcoli. Per cui si tratta di un elemento di disturbo non da poco

8.3.4 I brevetti software per loro natura mancano dei naturali confini che i brevetti in altri campi offrono

Infine, per come sono concepiti, i brevetti sul software tendono ad essere **intollerabilmente vaghi**. Un brevetto dovrebbe avere una sua delimitazione sicura, in modo che un operatore esperto nel settore sia messo in grado di conoscere cosa sia oggetto del brevetto e cosa non lo sia. In un sistema in cui si brevettano sostanzialmente idee astratte e algoritmi, non processi e materiali esistenti in natura, di cui si dà solitamente una descrizione funzionale ad alto livello, non è possibile **sapere in anticipo** con un sufficiente margine di sicurezza cosa sia brevettato e cosa no.

Tutto ciò crea incertezza e *frena* l'innovazione.

8.4 È tutto?

La *concurring opinion* analizzata è sicuramente l'atto di accusa più autorevole contenuto in un atto ufficiale di un giudice chiamato ad applicare le norme sui brevetti. Non sviscera interamente l'intero parco delle critiche che si possono muovere ai brevetti sul software, ma vi assesta numerosi colpi che potranno in futuro essere colti da altri giudici per scardinare un sistema malato, una volta per tutte.

Il punto più importante è comunque colto: il software è un ambito in cui si aveva da tempo una protezione ben più che sufficiente a creare un'intera industria, come la storia fino alla fine degli anni '90 dimostra, prima che nell'equazione venissero calati i brevetti. Questa protezione consente di remunerare chi si impegna a realizzare qualcosa con importanti investimenti e creare qualcosa di valore per il mercato, *dopo* che l'investimento è stato fatto e un "prodotto" è sul mercato. Qualcosa che consente di creare il fenomeno più rilevante e sconvolgente degli ultimi trent'anni: il software libero / open source. Questa protezione è il copyright, pur con tutti i suoi difetti. Soprattutto, siccome il sistema di proprietà intellettuale trova la sua ragion d'essere nel fornire un incentivo all'innovazione, i cui vantaggi per la collettività superano l'inconveniente di fornire un piccolo monopolio, non vi è un'analisi seria e convincente (o anche minimamente tale) che tale incentivo e vantaggio siano reali, in presenza di un "*controfattuale*" opposto proprio nella storia del software a cui si è fatto cenno.

La campana a morto per i brevetti software non suonerà mai troppo presto.

Capitolo 9

I dati aperti (open data)

Il movimento degli open data attira l'attenzione principalmente nel mondo pubblico, ma non è affatto necessariamente limitato a tale settore. La tematica riguarda anche il settore privato, anche se con regole leggermente diverse e lasciate più alla convenienza che a un obbligo, salvo casi particolari.

Come per ogni altro campo di indagine nella nostra disamina del mondo open, parlare di dati aperti vuol dire che vi sono dati chiusi. Come può essere chiuso un dato? Le possibilità sono molte, esaminiamo dunque i vari modi con cui i dati possono essere tenuti chiusi, per poi vedere come si possono aprire e perché.

9.1 Come si chiudono i dati

9.1.1 Dati chiusi perché non rivelati, o segreti

La prima forma di chiusura dei dati è la segretezza, o la non divulgazione, degli stessi. Spesso, nell'affrontare il tema dei dati aperti si inizia con la discussione sulla licenza. Non ha però senso parlare di una licenza se il dato viene tenuto privato.

Non vi è una regola generale per la quale i dati debbano essere resi pubblici. Anzi, semmai la regola è l'opposta: l'ordinamento tutela infatti le informazioni aziendali con due sostanziali norme del Codice della proprietà Industriale, ovvero gli articoli 98 e 99. Essi sono stati inseriti nel nostro ordinamento in esecuzione dei trattati TRIPS, dei quali riprendono in modo quasi letterale il contenuto. Prima di essi, si era giunti a una tutela comunque generale dei dati aziendali segreti, attraverso il divieto di concorrenza sleale, contenuto nell'art. 2598 del Codice Civile. I requisiti per ottenere tale tutela sono tutto sommato limitati: deve trattarsi di dati tenuti segreti, ovvero chi li possiede deve aver adottato

sufficienti misure per impedirne la divulgazione, e devono avere un qualche valore in sé, all'interno dei beni aziendali.

In ambito pubblico, invece, è difficile parlare di tutela del segreto aziendale. Valgono però due generali principi che potrebbero in astratto interferire. Il primo e più evidente è che i dati detenuti possono essere **dati personali** e perciò soggetti a divieti di divulgazione, o comunque a regole strette. Dati non personali possono comunque essere rilevanti per un ambito di tutela simile a quello aziendale, a tutela del **buon andamento** amministrativo.

La raccolta, la conservazione, il controllo e la gestione dei dati hanno un costo per l'amministrazione. Essi hanno anche un potenziale valore commerciale: l'amministrazione potrebbe infatti cederli dietro compenso. Anche l'attività di diffusione ha un costo, renderli disponibili senza ottenerne un ristoro potrebbe comportare un depauperamento del patrimonio dell'ente detentore. Si è in passato affermato che la diffusione di dati senza ottenere almeno il costo di copia, ma più propriamente un compenso sotto forma di *royalty*, e il divieto di ulteriore diffusione, fosse una potenziale fonte di **responsabilità amministrativa**. Alla luce di quanto diremo in seguito, tale preoccupazione direi che è venuta meno.

9.1.2 Dati chiusi perché oggetto di un diritto di privativa

La comunicazione a terzi di dati può non essere incompatibile con la tutela data dal segreto. Terzi possono ricevere i dati sotto condizione di segretezza, ed essere costretti a detenerli e usarli adottando tutele simili a quelle adottate dal titolare. Tale però una condizione difficile da ottenere e funziona solo a condizione che la diffusione sia limitata.

Almeno in Europa, tuttavia, è stata realizzata una forma di tutela diversa. Chi dimostri di aver effettuato rilevanti investimenti nella raccolta, ordinamento, verifica dei dati ha il diritto di vietare a terzi la copia, l'estrazione di parti sostanziali (anche tramite più estrazioni parziali) e la diffusione di tali dati, anche quando questi venissero in suo possesso. E ciò per quindici anni dalla data in cui la base di dati è stata costituita (prolungabili in caso di dimostrabili ulteriori rilevanti investimenti).

Questo è il diritto fondamentale sul quale le licenze, di cui diremo in seguito, si concentrano. Questo diritto è infatti simile al **copyright**.

9.1.3 Dati chiusi perché illeggibili

Il dato può essere reso solo teoricamente accessibile, ma di fatto inutilizzabile, perché chi lo riceve non può utilizzarlo in maniera efficiente.

I dati possono essere resi disponibili su un **supporto** non informatico, come ad esempio un foglio di carta, o una copia fotografica digitale. Tali dati richie-

derebbero in tal caso un'attività intensa di acquisizione e verifica del risultato dell'acquisizione. Potrebbero essere forniti su un supporto sì informatico, ma in un **formato** di archiviazione non standard (vedi il capitolo sugli open standard) e quindi utilizzabile solo con una particolare applicazione o piattaforma, o ancora da un solo soggetto che conosce lo standard e lo implementa. Infine, i dati potrebbero essere forniti su un formato aperto, ma totalmente incomprensibili perché la loro **presentazione** manca delle necessarie meta- informazioni sulla concreta **codifica** dei dati stessi. Ad esempio, un set di dati potrebbe essere fornito in un formato XML (che è uno standard aperto), ma in un sottoformato XML non completamente o affatto descritto quanto alla sua **semantica** (cosa vuol dire un dato inserito in una data posizione e con un dato attributo) e alla sua **sintassi** (come si interpreta un dato o un insieme di dati, ad esempio se un numero esprime l'età, il numero civico, il codice fiscale, o altro, di una data persona).

9.1.4 Dati semi-chiusi perché diffusi solo in forma aggregata o con insufficiente dettaglio

A cavallo della leggibilità e della mancata comunicazione, il fatto che i dati siano forniti soltanto in forma aggregata, mentre restano indisponibili i dati disaggregati, o la disaggregazione è a un livello insufficiente per consentire una elaborazione e una verifica di affidabilità, o una qualche forma di regressione statistica. Ad esempio, una ASL può diffondere i dati di mortalità per un determinato tipo cancro della popolazione comune per comune, ma non essere disponibile una serie di dati per età alla morte, per sesso, per familiarità, per condizione genetica rilevante, per occupazione.

È ovvio che il livello di dettaglio necessario perché vi sia un'utilità dei dati dipende dal tipo di analisi che deve essere compiuta e dal fatto che quei dati siano stati raccolti a monte per ciascun individuo o per una porzione significativa della popolazione.

9.2 Perché offrire dati aperti

9.2.1 La scelta del "se" pubblicare

Abbiamo già detto che per le aziende e i soggetti privati in genere, fornire dati aperti è una scelta inoppugnabile. Solo alcuni soggetti hanno il diritto di pretendere di ottenere tali dati, in genere si tratta di autorità pubbliche, come l'Amministrazione delle finanze o l'ISTAT o le autorità amministrative indipendenti.

Per i soggetti pubblici tale facoltà di non divulgare i dati è molto più limitata. Per alcuni dati, intanto, una forma di pubblicità è imposta. Gli obblighi pos-

sono essere di varia natura. I dati reddituali delle persone fisiche sono infatti ad accesso pubblico, anche se ottenerli in forma di dati aperti è impossibile, se non in forma anonima. In altri casi di sono veri e propri obblighi di fornire i dataset, imposti da normative, come ad esempio i dati delle imprese conservati dal Registro delle imprese, oppure gli indirizzi di posta elettronica certificata (PEC) utilizzati come domicilio elettronico da imprese e professionisti. Esistono pure direttive europee, come la direttiva PSI[1] (Public Sector Information), che impone condizioni di accesso paritario e pubblico, esteso all'utilizzo commerciale, di un esteso insieme di dati pubblici. Esiste poi la normativa interna sulla cosiddetta "**trasparenza**".

In molti casi, dunque, la pubblica amministrazione non è arbitra di decidere il "se" rendere disponibili a terzi alcuni set di dati. Può però essere arbitra di stabilire il "come"? In senso assoluto no. La stessa direttiva PSI pone alcuni paletti e requisiti (tendenziale gratuità o rimborso del costo marginale di diffusione, non discriminazione, divieto di accordi esclusivi). Inoltre vale nel nostro paese il principio "***open by default***", ovvero, qualora un dataset è pubblicato, e non è espressa una licenza, si deve presumere che tali dati sono liberi per ogni utilizzo, compreso quello commerciale. Si tratta dell'art. 52[2] del Codice dell'Amministrazione Digitale, "CAD", che fa riferimento, per la definizione di dato aperto, alla lettera b) del comma 3 dell'art. 68[3] del CAD, il quale tra l'altro impone che i dati aperti siano offerti in "formato disaggregato".

9.2.2 La scelta del "come" pubblicare: la licenza

Abbiamo detto sopra che i dati sono "protetti" in modo simile al copyright, ma non è copyright. Dunque si tratta di una forma di privativa legata al rilevante investimento fatto dal costitutore nella creazione della banca dati. Tale privativa assegna al suo titolare il diritto di proibire la copia e l'estrazione di parti sostanziali della banca dati, ma non l'uso dei dati quale fonte di informazione, se un terzo ne è a disposizione. Ogni operazione di trasformazione delle banche dati comporta solitamente anche una copia degli stessi; anche il semplice agganciarsi a una fonte esterna tramite strumenti di interrogazione (del tipo webservice) messi a disposizione da un terzo, comporta estrazione sistematica. Pertanto se un terzo volesse usare una banca dati di terzi per includerla in un proprio servizio deve di regola ottenere il permesso del titolare.

Il permesso, come sempre, può essere concesso caso per caso (con tutto il sovrappeso burocratico connesso) o una volta per tutte con una **licenza pubblica**. La licenza pubblica a sua volta può contenere condizioni, modalità, limitazioni. Può insomma essere una licenza "proprietaria". Ma può più verosimilmente essere una **licenza aperta**, e allora possiamo parlare, almeno sotto il profilo legale, di "dati aperti".

[1] https://ec.europa.eu/digital-single-market/en/european-legislation-reuse-public-sector-information
[2] http://www.agid.gov.it/cad/accesso-telematico-riutilizzo-dati-pubbliche-amministrazioni
[3] http://www.agid.gov.it/cad/analisi-comparativa-soluzioni

Esistono licenze espressamente dedicate ai dati, e sono solitamente licenze dotate di un qualche tipo di copyleft. Una di queste è la IODL[4], che nella versione 2.0 ha di molto attenuato il suo copyleft rispetto alla versione 1.0. Pur essendo una licenza tecnicamente ben fatta, io personalmente ho espresso piuttosto chiaramente già dallo studio effettuato con Simone Aliprandi nel progetto freegis.net[5] l'opinione che i dati aperti della pubblica amministrazione vadano pubblicati con una licenza il più possibile vicina al pubblico dominio, indicando nella licenza Creativa Commons Zero (CC0), di cui abbiamo già parlato nell'articolo dedicato ai contenuti liberi.

Esistono opinioni difformi. L'Agenzia per l'Italia Digitale (AGID) ha pubblicato una guida[6] per la pubblicazione dei dati aperti, in cui, pur citando incompletamente uno studio mio e di Aliprandi (che riprende quello citato qui sopra), perviene a condizioni differenti. In particolare ha citato la necessità di usare una licenza di attribuzione come la CC BY, e non la CC0, perché è l'unica che rispetta i diritti morali. A pagina 80 si legge:

> Infine, occorre ricordare che alla maggior parte dei dati e dei documenti necessari per lo svolgimento delle funzioni tipiche delle pubbliche amministrazioni non è opportuno applicare la CC0, in quanto questa prevede il rilascio dei diritti morali che sono inalienabili, indisponibili, imprescrittibili secondo le norme nazionali ed europee

Mi si permetta una piccola nota polemica. L'affermazione citata contiene ben **due errori** di importanza capitale.

La privativa sulle basi dati **non prevede diritti morali**.

Infatti, non si parla di "autore", ma di "costitutore". Evidentemente l'estensore si è fatto trarre in inganno dall'articolo 64-quinquies della Legge sul Diritto d'Autore, che parla appunto di "autore", ma la legge italiana ha trasposto in maniera insufficientemente chiara la direttiva[7] sulla protezione dei database, che distingue tra il copyright sulla banca dati, qualora la stessa "per la scelta o la disposizione del materiale costituiscono una creazione dell'ingegno propria del loro autore" abbia protezione sotto il copyright (art. 3.1). Ma qui si parla di diritti *sui generis*, previsti dall'art. 7 della direttiva, pacificamente diretta ai "costitutori" della banca dati, e dunque si parla del diritto previsto dall'art. 102-ter della LDA. Se si pensa, l'attribuzione del diritto d'autore è effettivamente a una o più persone fisiche, mentre il "rilevante investimento" per solito è effettuato da una **persona giuridica**.

E qui viene in considerazione la **seconda** topica presa da chi ha scritto quel pezzo. La licenza CC BY, ne abbiamo parlato nel relativo capitolo, rimandando a questo contributo, non tutela affatto il diritto morale, ma *l'attribuzione di*

[4] https://it.wikipedia.org/wiki/Italian_Open_Data_License

[5] https://freegis.net/documents/10157/14646/FreeGIS+data+licence+1

[6] http://www.agid.gov.it/sites/default/files/linee_guida/patrimoniopubblicolg2014_v0.7finale.pdf

[7] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31996L0009:IT:HTML

provenienza, che è concetto assolutamente distinto dal diritto morale di vedersi riconosciuta la paternità. L'attribuzione della fonte sarebbe molto probabilmente fatta, inoltre, all'ente costitutore, e non alle singole persone (chi? I dataset pubblicati contengono i nomi delle singole persone, forse?) le quali vi hanno lavorato (ma che non sono autori). L'ente, quale – in quanto **persona giuridica** – non può affatto vantare diritti morali. Si tratta evidentemente di una ragione inesistente.

CC0 va benissimo.

Per approfondimenti suggerisco la lettura di Simone Aliprandi, Il fenomeno open data [8]

[8] http://www.aliprandi.org/fenomeno-opendata/

Capitolo 10

API e nuvole, la faccia chiusa del web

L'openness ha una faccia oscura, che si fa vedere solo attraverso un messaggero fumoso e impercettibile. È il "cloud computing", tanto di moda nei circoli di quelli che parlano di innovazione, non sempre a sproposito. Il cloud, però, non è altro che software dei cui servizi ci avvaliamo tramite **interfacce e protocolli via rete**. Software che sta da qualche parte, sovente su un computer di qualcun altro, a cui quindi non abbiamo accesso, se non tramite "servizi" che vengono esposti via rete. Questi servizi vengono fruiti o direttamente, tramite una pagina web accessibile in un browser, o "dietro le scene", tramite altri protocolli e interfacce in cui sono i computer e i relativi programmi a parlarsi tra loro. Protocolli e interfacce che vanno sotto il nome di "API". Ecco spiegato il titolo, almeno la prima parte.

10.1 Un problema di disponibilità

Nei capitoli precedenti abbiamo discusso come le questioni dell'openness nascano principalmente dalla "chiusura" data da diritti di privativa (diritto d'autore, brevetti, diritto *sui generis* del database). In certi casi, come per il software distribuito come codice oggetto, la mancanza di accesso al codice sorgente impedisce un'adeguata modificabiltà (dunque adattabilità) del software. Mancanza tutelata dal diritto attraverso la tutela del segreto (il codice sorgente è considerato segreto, cercare di rivelarlo tramite decompilazione è generalmente vietato, salve eccezioni: una di queste è appunto la ricerca dell'interoperabilità, di cui diremo oltre). Nel caso di software accessibile solo via rete – come nel cloud computing, ma anche in tutti i casi in cui occorre **interfacciarsi** con altro software detenuto da altri e reso accessibile via rete – la ragione di possibile mancanza di

apertura deriva principalmente dal fatto che il software è da qualche altra parte a cui non si ha fisicamente accesso, e comunque che non si controlla.

Non che ciò sia necessariamente una cosa negativa, semplicemente è uno scenario diverso da quello concettualmente più semplice, di un sistema IT di cui si "**controlla**" tutto, dall'hardware fino agli strati superiori. Per controllo intendo quanto meno avere la potestà della decisione di installare o disinstallare una componente (hardware o software, libera o proprietaria che sia). Se il software è installato altrove, la decisione non è mia, posso solo usare quello che mi viene messo a disposizione via rete.

10.2 Un problema di API

API sta per "Application Programming Interface", ed è un termine che indica l'insieme di modalità di interazione tra due componenti software. Si distinguono dalle GUI (o semplcemente UI) acronimo di "Graphic User Interface", ovvero l'interfaccia uomo-computer, perché – come ovvio – l'elemento umano non interviene, tutto avviene in modalità automatica; nel cloud computing, e in genere nelle applicazioni "distribuite" su varie risorse, l'interazione avviene via rete (RPC calls, web services, eccetera). Nel prosieguo parleremo indifferentemente di "specifiche", "protocolli", "interfacce", "API", i quali, benché concettualmente differenti, ai fini della nostra analisi possono essere usati in modo intercambiabile.

Se dunque io voglio creare un'applicazione che "parli" con una applicazione che espone delle API, ho almeno due ordini di problemi: di accesso (logico e fisico) e di ordine giuridico (copyright e brevetti, in primo luogo).

10.3 Il segreto: specifiche non documentate, che ballano il Samba

Le interfacce possono essere perfettamente accessibili, ad esempio nel caso in cui il software sia tutto locale e le API siano "esposte" (ovvero non serva una chiave o un altro componente per interagire), ma essere sconosciuta la lingua che parlano.

Un caso che ha fatto scuola, e che conosco sufficientemente bene per aver partecipato direttamente ai vari processi alla Corte di Giustizia dell'Unione Europea, è il caso Microsoft. Parlarne compiutamente sarebbe troppo lungo, ma il caso verteva sulla possibilità che un operatore indipendente reimplementasse i protocolli e le interfacce di rete dei sistemi Microsoft Windows, facendo "finta" di essere dall'altra parte un sistema Windows. Le API dei sistemi di rete di Windows non sono pubbliche, a differenza delle API che consentono ai programmi

di interfacciarsi con il sistema operativo locale: esse sono (erano) tenute segrete, almeno da una certa data in poi (per coincidenza, più o meno da quando Microsoft era diventata "dominante" nei sistemi "workgroup").

Un sistema operativo diverso, per esempio Linux, non aveva dunque la possibilità di "inserirsi" in una rete Windows. Non poteva fare il server in una rete di client Windows, non poteva fare il server in un gruppo di server Windows, non poteva fare il client in una rete di server e client Windows, se non tramite protocolli diversi e meno adatti. Era nella stessa situazione di un cliente in un caffè francese che non sapeva parlare francese, aveva i soldi, ma non sapeva come ordinare qualcosa (né esattamente cosa ordinare). La metafora viene da una descrizione dell'autore ˆhttps://www.samba.org/ftp/tridge/misc/french_cafe.txt] su come è nato il suo progetto Samba[1] .

Samba è una reimplementazione dei servizi di rete di Windows, nata dal protocolo SMB (da cui il nome "Samba") ad opera di Andrew "Tridge" Tridgell, una delle persone più geniali che abbia mai conosciuto. Per aggirare il segreto che a un certo punto era sceso sui protocolli nelle nuove versioni della parte di servizi di workgroup server (condivisione di file, servizi di autenticazione e stampa), egli si mise ad "annusare" il traffico di rete tra due macchine Windows, per scoprire – riusiamo l'analogia – come si chiama quel pezzo di pane lungo e strano, o quella bevanda in quella tazza, come si fa una domanda invece di un'affermazione, cercando di capire i pezzi della "lingua" che compone il protocollo, il ruolo di essi, la giusta sequenza, come combinarli per fare una domanda e per dare una risposta e così via. Fino a creare un componente che, visto dalla parte del server "Windows", sembrasse un altro Windows. E poi sostituire la macchina Windows con Samba, per vedere i messaggi di errore generati intenzionalmente e imparare ancora di più.

Alla fine, il risultato delle azioni a cui ho partecipato è stato di ritenere che il segreto imposto da Microsoft sui propri protocolli fosse contrario alle normative antitrust, perché consentiva alla stessa di controllare in modo indebito uno standard "di fatto", che era essenziale per competere nel mercato. Il segreto infatti non necessariamente doveva essere su tutto per frustrare **l'interoperabilità**, ma poteva anche limitarsi a pochi dettagli tenuti nascosti, un granello di sabbia gettato negli ingranaggi dei propri concorrenti, che interrompesse la piena interoperabilità. La condanna a fornire ai concorrenti complete e tempestive informazioni sui protocolli è stata confermata dal tribunale europeo.

10.4 I brevetti

Risolta la parte sui brevetti, il Team Samba (e io per loro) si è ritrovato nella situazione di avere pieno accesso alle specifiche dei protocolli di Microsoft, perfettamente documentati in modo tempestivo e completo, come richiesto dalla

[1] https://www.samba.org/

decisione della Commissione (il ritardo nella fornitura di tale documentazione è costato a Microsoft qusi tre miliardi di multa).

Il problema è che la documentazione non è del tutto sufficiente per poter implementare i protocolli senza problemi legali. Infatti, i protocolli di Microsoft sono tutelati da brevetto. Questo significa che chiunque voglia implementare gli stessi protocolli in un'applicazione che si vuole interoperabile (ad esempio Samba), necessariamente rientra nell'ambito di protezione dei brevetti. Nel caso di Microsoft i brevetti facevano comunque parte del pacchetto-condanna, ma sono poi stati concessi in larga parte sotto l'Open Specification Promise, un impegno a non utillizzarli "aggressivamente", il che ha effetti simili a una licenza generale. Alcuni protocolli, tuttavia, rimangono fuori, come ad esempio Active Sync, un protocollo che consente di sincronizzare informazioni tra device differenti. Altri casi potrebbero essere rilevanti.

10.5 Copyright

Per il diritto d'autore, la situazione è più sfumata. Infatti è idea comune che le interfacce non siano protette da copyright. Questo perché il copyright copre, come abbiamo già detto varie volte, solo la forma di espressione originale, e non l'idea. Corollario è che tutte le volte in cui la forma di espressione coincide con l'idea, perché esiste solo un modo per "dire" una determinata cosa, esprimere un determinato concetto, usare il copyright sull'espressione equivale a usarlo sull'idea.

In Europa questo concetto è stato espresso in modo chiaro e inequivocabile nel caso SAS Institute v. World Programming Language (Caso C-406/10. In tale caso un concorrente ha reimplementato in modo perfetto le interfacce e i protocolli (e i formati di file, ma è un'altra questione) di SAS, una società che produce un famosissimo e costoso programma di statistica. L'obiettivo dichiarato era quello di far sì che i programmi statistici sviluppati per SAS dagli utilizzatori (dunque non materiale di SAS) potessero essere utilizzati in modo corretto nel programma concorrente. Sostanzialmente la stessa cosa che vuole fare Samba, raggiungere quel livello in interoperabilità che si chiama "drop-in replaceability": la capacità di prendere un pezzo del sistema, rimpiazzarlo con un pezzo diverso, senza che nessuno se ne accorga. La stessa cosa che facciamo quando una lampadina si rompe e la rimpiazziamo con un'altra: il risultato è luce, perché la seconda ha lo stesso passo, usa la stessa tensione e frequenza dell'altra.

La sentenza della Corte è stata appunto nel senso di consentire questa ricostruzione, nonostante nell'implementare i protocolli il concorrente abbia dovuto riusare un "dizionario" di variabili e costrutti *identici* a quelli di SAS, i quali, visti da soli, sarebbero stati da chiunque considerati proteggibili. Tuttavia, in quanto usati come interfacce, essi sono stati considerati non oggetto di copyright. La stessa cosa sembra dover poter essere la situazione giuridica delle interfacce

negli USA, solo che in quel sistema si giunge a situazioni simili attraverso il *fair use right*, il quale (è una sottigliezza importante) è una causa di giustificazione, non una condizione di esenzione dal copyright. Questa differenza è importante ed è al momento in cui scrivo discussa nel caso Oracle v. Google sulla reimplementazione in Android delle interfacce e protocolli di Java (e delle relative librerie). Pur essendo la norma, ai fini pratici, sostanzialmente identica, il grado di certezza con cui ne parliamo è quindi diversa, in quanto la sua applicazione concreta è ancora, almeno in parte, *sub judice*).

10.6 Andiamo nelle nuvole

Abbiamo visto che ci sono vari diritti e situazioni protette da diritto coinvolte. Tutte però presuppongono che le interfacce *esistano* e siano rese disponibili. Niente impone a nessuno di crearle e renderle disponibili, in quanto nessuno (o quasi) può imporre al titolare di un servizio di creare un pezzo di software, ma al massimo la legge può limitare il livello di controllo tramite il diritto (incluso il segreto, anch'esso tutelato da un diritto, ovvero il divieto di decompilazione).

Il sistema usato da Samba per ricostruire i protocolli e interoperare con le interfacce presuppone che le interfacce possano essere utilizzate e il traffico tra di esse ("*through the wires*", attraverso i cavi) possa essere intercettato e analizzato, tramite un'operazione particolare di *reverse engineering* (di tipo osservazionale, non di decompilazione del software). Il *reverse engineering* presuppone però di avere accesso all'oggetto dell'analisi. Se chi ospita il software in cloud non rende disponibili determinati servizi, non esiste niente che possa aggirare tale mancanza. Se non è possibile esportare i dati attraverso l'applicazione, non è possibile nemmeno attraverso il sistema operativo o il motore di database su cui magari il software si appoggia, come nel caso in cui il software sia installato su una macchina del cliente.

Ciò ha conseguenze in caso di cessazione del contratto con l'operatore dei servizi in cloud. Il rimedio a questa situazione sembra essere puramente contrattuale, o anche semplicemente giudiziario. Dunque occorre rendere obbligatorio e contrattualizzato, ad esempio, un sistema per garantirsi di non perdere dati e l'operatività anche senza la collaborazione dell'altra parte. Una combinazione di presidi tecnici e contrattuali che garantiscano (ragionevolmente) tutto ciò. Altrimenti in un ambiente cloud gestito da altri si è molto più vincolati al fornitore di un sistema in cui si controlla tutto.

10.7 Un cenno all'antitrust e conclusioni

Nel caso Microsoft, però, la società americana è stata condannata a fare qualcosa: produrre una documentazione accurata, completa e tempestiva (aggiornata) in

cui pubblicava le informazioni di interoperabilità dei propri protocolli. È un rimedio esperibile solo nel caso in cui ci sia un operatore in posizione dominante sul mercato e questi ne abusi. Un caso raro, si dirà? Sarà, ma a me è capitato di occuparmene almeno in tre casi diversi, in tutti e tre i casi con l'apertura di una indagine formale; nel caso Microsoft, con una condanna, in un caso diverso – riguardante i servizi innovativi nella scuola) – con impegni formali (un patteggiamento) delle imprese dominanti, nel terzo caso, appena diventato di dominio pubblico, nel caso del fornitore di servizi del processo civile telematico.

Questo dimostra che la tentazione di usare la posizione di chiusura data dal controllo delle interfacce è molto alta, il conflitto di interessi rischioso e il rimedio antitrust disponibile solo in determinati casi. Occorre dunque vigilare e tenerlo presente allorché si affida ad altri una parte del nostro sistema informatico, avendo sempre presente una *exit strategy* nel caso incerto solo nel quando, non nel se, di abbandono dell'attuale fornitore. Il rischio di *lock-in* in questo caso è estremamente alto.

Capitolo 11

Nuvole aperte, nuvole chiuse e nuvole nere

Proseguiamo il discorso iniziato – con particolare riferimento alle API e alla loro disponibilità – sul **cloud computing**, in particolare sul cosiddetto **cloud pubblico** e alle relative problematiche concorrenziali e di interoperabilità, per affrontare più in generale le declinazioni e le problematiche di *openness* nei sistemi cloud.

11.1 Cos'è il cloud

Il cloud non esiste.

Esiste una serie di tecnologie di virtualizzazione, di condivisione e di interscambio di dati e di servizi che vanno sotto il nome comune di "cloud", ma da un punto di vista tecnico e giuridico sono molto diverse tra loro. Qui ci occupiamo esclusivamente di quella tipologia di cloud denominata "**pubblica**", ovvero in cui il software e i dati sono in tutto o in parte "ospitati" da fornitori di servizi non controllati da chi li utilizza, e sono regolati da rapporti contrattuali in cui non vi è (se non marginalmente) consegna di software. Le cloud private sono, invece, solamente un modo di organizzare il software, con tecnologie magari identiche a quelle usate nel pubblico, ma in cui non vi è una **separazione tra il fornitore e l'utilizzatore** del servizio a sua volta fornito dal software, che è in gran parte la ragione per cui ci occupiamo specificamente di cloud con gli accenti a volte negativi che troveremo in seguito.

"Pubblico", in questa accezione, non riguarda dunque la natura legale del soggetto che si avvale o che fornisce i servizi in cloud, ma si riferisce al fatto che i

servizi siano **offerti** "**pubblicamente**". Dunque "pubblico" nel senso di "sulla (metaforica) pubblica piazza".

In particolarmente ci occupiamo di "**Software as a Service**" (SaaS) e di "**Platform as a Service**" (Paas), secondo una nomenclatura ormai accettata universalmente. Servizi *in luogo* di software.

11.2 Le libertà del cloud

Nel cloud, e da qui avanti useremo questo termine per intendere solo PaaS e SaaS, tutto quello che abbiamo detto in precedenza in termini di openness nel software **non vale**. Nel software ottenuto via cloud, infatti, non vi è distribuzione di software, ma soltanto un'esposizione via rete dei servizi che il fornitore decide di mettere a disposizione, e solo con le interfacce e i protocolli da questi predisposti. Pertanto, il "cliente" non può certamente beneficiare delle quattro libertà[1]: non può usare il software, se non per quello che gli è consentito dal fornitore; non può distribuire il software, perché non lo riceve lui stesso; non può studiare né modificare il software, perché non ha né il codice oggetto, né il codice sorgente; non può infine distribuire copie modificate, perché non le può modificare.

Con una disponibilità di banda sempre più elevata a costi sempre più bassi, lo spostamento di un sempre maggior numero di servizi dal software gestito in locale a software gestito in remoto via Internet, dunque "nella nuvola", diventa un paradigma con il quale fare i conti, ma verso il quale gli strumenti ormai consolidati e che abbiamo analizzato in precedenza hanno poco impatto.

È dunque tutto perduto, tutto inutile? Non dobbiamo usare il cloud o rassegnarci a un mondo non libero? Andiamoci piano.

Se nel cloud dunque non possiamo utilizzare le categorie logiche e giuridiche del software "distribuito", abbiamo tuttavia alcune caratteristiche che ci consentono di discriminare un "buon" cloud da uno non poi troppo buono.Vi sono una serie di "libertà" che dobbiamo tenere presente per valutare se, giuridicamente e tecnicamente, si possa ottenere una situazione simile a quella che avremmo in una situazione di software libero.

11.3 Migrazione: anche una questione di costi

Una cosa è certa: non staremo per sempre con lo stesso fornitore di servizi. Dunque i costi di migrazione sono "*sunk cost*". Significa che si tratta di costi inevitabili, in quanto che non dipendono da una decisione futura, ma dipendono da una decisione decisione passata e non più modificabile. Ad esempio, i costi di migrazione devono essere valutati sin dal giorno in cui concludo un contratto

[1] https://www.gnu.org/philosophy/free-sw.it.html

con il nuovo fornitore. "Addebitare" i costi di migrazione al vecchio fornitore comporta scelte irrazionali, in quanto induce a scegliere il fornitore con i costi di ingresso più bassi, ma i costi di uscita alti. Il che è irrazionale.

Vi è infatti un incentivo per il fornitore ad alzare i costi di uscita, abbassando l'interoperabilità e la migrabilità dei servizi. Ciò crea una **dipendenza dal fornitore** in capo al cliente: anche se vi fosse una soluzione più efficiente ed economica, aggiungendo i costi di migrazione la nuova soluzione sarebbe sempre handicappata rispetto a quella attuale, perché sopporterebbe costi di migrazione che restare presso l'attuale fornitore non affronta.

Come evitare i costi della migrazione? Purtroppo non è possibile se non affrontando sin da subito gli aspetti **contrattuali**, ad esempio, addossando al fornitore attuale il costo e gli oneri di migrare i servizi verso una soluzione standard. La migrazione non deve essere dalla soluzione "proprietaria" ad ogni n soluzioni proprietarie di operatori, ciò sarebbe assurdo e impossibile. La migrazione a carico del fornitore uscente deve essere contrattualmente stabilita fino a un "**Punto di raccolta**". Questo punto di raccolta deve essere visto in una situazione standard di dati e di servizi. Usando questa strategia il futuro fornitore uscente è incentivato a usare sin da subito **open standard**, o quantomeno a usare soluzioni che consentano di realizzare in modo sufficientemente spedito ed economico soluzioni basate su standard aperti che garantiscano una piena interoperabilità. Invece di avere incentivi congruenti verso il *lock-in*, abbiamo un incentivo naturale verso il *lock-in* equilibrato da un incentivo opposto ad evitarlo, perché il costo del *lock-in* viene pagato alla fine dal fornitore attuale.

11.4 Interoperabilità

Anche nei servizi cloud esiste una tematica di interoperabilità, ovvero di far "parlare" i servizi per scambiarsi dati e funzionalità. Per avere interoperabilità occorrerà dunque avere anche qui servizi standard, o quantomeno adeguatamente documentati (standard *de facto*), e formati altrettanto standard (per i documenti latamente intesi).

L'interoperabilità deve essere valutata sia durante la fase di conduzione *normale* del servizio, sia durante la fase di **migrazione** (o cosiddetta "ripresa" dei dati e di servizi). L'interoperabilità in fase di conduzione consente di utilizzare i servizi del fornitore considerato *in congiunzione* con i servizi di un terzo. Ciò riduce la dipendenza in caso di aumento delle esigenze e di nuovi *workload* che dovessero rendersi necessari.

L'interoperabilità in fase di migrazione, invece, consente di esportare e importare sia i dati che i servizi (il concetto di migrazione dei servizi può sembrare ostico, ma è una cosa abbastanza naturale) senza perdita di fedeltà né nei dati, né nella *business logic* associata ai servizi. Nei costi di migrazione da considerare è ovviamente da comprendere quello di indisponibilità del sistema nella fase

di migrazione (totale o parziale). È appunto di questa situazione che abbiamo parlato nel punto precedente.

Certo, non sarà possibile ottenere una piena interoperabilità e migrabilità sulla base di interfacce e soluzioni di interscambio valide per tutti, ma sempre di più si stanno affacciando tecnologie mature ed esempi di successo di approcci basati su interfacce standard e persino open source nei servizi cloud. La loro adozione diretta è una soluzione per determinare l'interoperabilità, certo, ma la presenza di tali iniziative fa sì che ci sia un sistema (o più sistemi) di riferimento verso i quali i produttori di soluzioni proprietarie possono creare interfacce. Una di queste soluzioni è il progetto Open Stack[2].

11.5 Sicurezza e privacy

Mentre scrivevo queste righe sul post originale, arrivò la notizia che Amazon (AWS) stava sperimentando[3] estese interruzioni del servizio. Come noto, sui servizi di Amazon sono basati migliaia di siti di tutti i tipi. Un adagio vuole che non si mettano tutte le uova nello stesso cesto, ma è esattamente ciò che stiamo facendo. Certo, è conveniente. Certo, è comodo. Certo, è sicuro. Dipende però da ciò che si considera "sicuro". La sicurezza non è una dimensione lineare, è un insieme di fattori che vanno bilanciati. Si suole dire che la sicurezza di un sistema è simile alla robustezza di una catena: si misura dal più debole dei suoi anelli.

Con la concentrazione di servizi e funzioni ("*workload*"), si fa esattamente questo: si mettono un sacco di uova in uno stesso cesto. Il problema è che quando i sistemi su cui questo cesto si fonda si interrompono, un pezzo rilevante di Internet se ne va a fare un giro per un po'. E con esso un sacco di gente che ci lavora sopra. Il rischio di cui parliamo è dunque quello di **indisponibilità** (*business continuity*), che è un rischio esattamente come lo è quello di perdita e di diffusione involontaria di dati.

Parlando di diffusione, o meglio, di rivelazione involontaria, essa è un fatto indipendente dalla conoscenza "psicologica" di una persona determinata, è sufficiente che i dati siano conoscibili da un soggetto, ecco che per le leggi sulla protezione dei dati personali non sono conservati correttamente. Il nuovo Regolamento sulla privacy obbligherà in molti casi a fare un auto-accertamento dei livelli di rischio (frequenza, gravità dell'incidente e delle potenziali conseguenze) e la sicurezza andrà parametrata al livello auto-accertato, senza una soluzione unica per tutti. Per questo, i livelli di sicurezza da adottare potranno essere minori, ma in moltissimi casi saranno più elevati.

[2] https://www.openstack.org/ "Open Stack"

[3] http://www.theverge.com/2017/2/28/14765042/amazon-s3-outage-causing-trouble "Amazon down"

La possibilità che un terzo venga a conoscenza dei dati conservati è solo una delle possibili conseguenze di una cattiva conservazione. Anche la mancanza di "**trasparenza**" sulla **catena di responsabilità** nella gestione dei servizi è un elemento da tenere presente. Per dare trasparenza (ai propri interessati) occorre avere trasparenza in proprio. In altre parole, è necessario conoscere chi è investito dei compiti di rendere le varie componenti dei servizi resi, in tutta la catena di fornitura.

Tornando alla questione delle uova tutte in un cesto, però, ancora l'interoperabilità e la possibilità di migrare, o meglio, **federare** sevizi diversi, senza avere degradata la qualità del servizio, è ciò che riduce di gran lunga anche il rischio di dipendenza da un unico fornitore esterno. I servizi che per loro natura sono "commodity", ovvero forniti da un numero non limitato di soggetti e intercambiabile, danno intrinsecamente maggiore sicurezza e indipendenza, consentendo la ridondanza dei sistemi e dei fornitori. Si pensi alla connessione Internet, ai DNS, all'archiviazione dei dati, ai servizi di database eccetera. Tutti questi servizi sono disponibili e accessibili in quanto sono basati su **standard interoperabili**. Il che è un modo di definire una caratteristica degli standard aperti.

11.6 Conclusioni

L'uso di sistemi basati su paradigmi nuovi porta sfide e possibilità prima non concepibili. La tendenza a sfruttare sempre di più SaaS e PaaS nei servizi ad ogni livello, anche da parte della Pubblica Amministrazione, crea una serie di possibili punti di fallimento che – pur anticipati da molti – vengono accettati come reali solo quando gli eventi, come quello di questa sera in cui concludo queste righe, rendono evidente.

Non si tratta di essere Cassandre o teorici del complotto. Né si tratta di essere faciloni improvvisati. Conoscere le sfide, appunto, è il modo migliore di affrontarle. Siamo ancora in larga parte in **terra incognita**, questo fatto da solo merita rispetto e attenzione, e fa sì che solo un atteggiamento prudente – né luddista, né tantomeno stupidamente entusiasta e incosciente – sia quello da tenere.

Il faro che ci ha guidati sin qui, quello di prediligere sempre e comunque una scelta di apertura nella tecnologia e di tendenziale condivisione, dove possibile, sembra non aver ancora esaurito la sua luce. Se c'è una cosa su cui mi sento di scommettere è che questa luce ci guiderà ancora per tanto tempo, con buona pace di chi, decenni fa come oggi, pensa di avere la verità in tasca e scuote la testa quando parliamo di questi temi.

La storia ci ha dato ragione in passato, ci darà ragione anche in futuro.

Lista delle abbreviazioni

(GNU) GPL GNU General Public License. La principale e più famosa licenza copyleft, gestita dalla FSF, oggi alla Versione 3.

FSF Free Software Foundation. Istituzione fondata da Richard Stallman per la gestione del primo progetto di Software Libero (Free Software), ovvero GNU

GNU GNU's Not Unix. Progetto di Software Libero di FSF per la creazione di un sistema operativo UNIX-like senza dipendenze da software proprietario. L'acronimo è evidentemente ricorsivo.

API Application Programming Interface. Regole e interfacce stabiliti per interoperare tra due differenti servizi o programmi.

SaaS Software as a Service. Software che non viene eseguito sulla macchina dell'utilizzatore, ma da un terzo su macchine proprie, che ne mette a disposizione i servizi.