



Introduzione

Tra codice e sorgente

È nato prima il sorgente o il binario? Questa volta il vecchio gioco dell'uovo e della gallina ha una soluzione semplice e chiara: è nato prima il codice, sono nate prima le parole che hanno dato voce al programma compilato.

Se facciamo attenzione al significato delle parole codice e sorgente possiamo richiamare alla memoria vecchi termini, oggetti antichi, umidi e coperti di polvere, che nel passato hanno portato il peso della cultura in un vecchio mondo sempre uguale a sé stesso, restio ai cambiamenti.

Il codice, ovvero l'insieme delle istruzioni di base che noi passiamo a un computer per generare un binario, il vero e proprio programma che eseguirà i compiti che noi abbiamo prestabilito, è in fin dei conti la reale fonte, la sorgente del sapere informatico. È un po' come se ritrovassimo un vecchio codice medioevale, un antico testo alchemico che ci insegnasse a creare l'oro dal piombo: apriamo il testo antico, leggiamone le parole, eseguiamo le operazioni descritte e infine otterremo l'oro. Meglio ancora se trovassimo un codice che ci consentisse di ottenere la pietra filosofale, ma accontentiamoci. E se avessimo direttamente tra le mani l'oro alchemico? Bene, se fosse veramente oro non saremmo molto più ricchi di prima, avremmo in mano un blocchetto di metallo prezioso e niente più. Non che ciò sia poco, ma ha un valore relativo. Se invece avessimo tra le mani il codice medioevale, con tanto di miniature e di testo, scritto in fitte minute? In questo caso, avremmo molto di più, sapremmo come ottenere l'oro, sempre, in qualsiasi luogo e momento contemplati dal codice, potremmo arricchirci oltre il semplice blocchetto d'oro che potrebbe capitarci dalle mani.

Cosa ci porta quindi il codice? Ci porta la conoscenza, la consapevolezza del come ottenere un risultato, di quale strada percorrere, di quali errori evitare. Pensate di avere a disposizione un genio, di quelli delle “Mille e una Notte”, una di quelle creature della fantasia mediorientale, capace di soddisfare ogni nostro desiderio, e immaginate di chiedergli di sapere ogni cosa di questo mondo. Cosa otterreste? Molto, forse? Ben poco, invece. Non basta il semplice sapere per operare, ma ci vuole la consapevolezza di ciò che si sta facendo, il che è decisamente diverso: è come avere una conoscenza racchiusa nel fondo della nostra memoria e non sapere di averla, lasciandola quindi inutilizzata e inesistente.

Il codice di un programma è quindi per noi la sorgente della conoscenza come consapevolezza del cammino da percorrere, degli espedienti tecnici e di programmazione che ci consentono di arrivare a costruire l'applicazione che ci permetterà di svolgere i compiti che ci prefiggiamo.

La distanza tra codice e binario è quella che ha sempre separato l'utilizzatore di un sistema operativo dal programmatore: il primo si avvale di applicazioni già pronte, il secondo è libero di crearne una che risponda alle proprie necessità; il primo utilizza una conoscenza non sua, il secondo fa proprio il sapere altrui per rielaborarlo in una creazione autonoma. Ma attenzione. L'importanza del codice non risiede solamente nella possibilità di creare un'applicazione autonoma, innovativa, ma nell'apertura dell'orizzonte temporale dell'ereditarietà: il programmatore immette nel proprio codice il sapere che ha acquisito per studio ed esperienza, creando codice autonomo, ma può in un certo senso anche “ereditare” una conoscenza acquisita da altri programmatori, utilizzando sorgenti già pronti, prendendo qualcosa di già creato e riplasmandolo per implementare nuove funzionalità.

La possibilità di accedere al codice sorgente di un'applicazione, quindi, ha come primo e più naturale effetto quello di inscrivere il proprio codice in una tradizione di accumulo e accrescimento della conoscenza, in questo caso informatica, comune a qualsiasi forma del sapere. Le informazioni che noi produciamo provengono quindi da una tradizione di informazioni, vi si stratificano e possono divenire il sostrato di nuove creazioni.

La libertà di accesso ai codici sorgente significa quindi una maggiore apertura del flusso delle informazioni, una più ampia possibilità di ricevere, rielaborare e trasmettere conoscenze, ma non solo.

Immaginiamo di avere fra le mani il codice sorgente di un'applicazione di quelle che ci "risolvono la vita", ovvero possono quasi del tutto soddisfare le nostre esigenze. Manca poco, solo qualche funzione in più, qualche ritocco fra le righe di istruzioni e ciò che abbiamo ricevuto potrebbe diventare il programma che stiamo tentando di costruire. Cosa succederebbe se il sorgente fosse posto sotto il vincolo del "Guardare ma non toccare", una di quelle proibizioni che risalgono ai tempi della nostra infanzia? Semplicemente, avremmo per le mani qualcosa con un grande valore potenziale, ma senza alcun valore reale: non potremmo usare il codice, modificarlo, adattarlo. Potremmo al massimo imparare da ciò che leggiamo e riformularlo in modo da aggirare il divieto iniziale, perdendo in questo espediente molto tempo e molte energie.

La possibilità data dall'Open Source di far circolare, modificare, riutilizzare rapidamente il software ha dato vita, con Linux, a una modalità di sviluppo decisamente innovativa nella sua ampiezza, se non nella sua origine, il cosiddetto bazaar. Uno dei più famosi testi sull'Open Source "La cattedrale e il bazaar" di Eric S. Raymond, attribuisce proprio a Torvalds, il creatore di Linux per intenderci, l'abilità di avere portato a un grado di estensione mai raggiunto prima una pratica comune nel mondo Unix, ovvero quella di distribuire versioni successive di codice sorgente in modo che altri programmatori rispetto a chi ha creato la prima versione, possano revisionarlo, modificarlo e inserire le variazioni nel codice iniziale. Con Linux questa pratica diventa molto diffusa: Torvalds rilascia versioni successive del sorgente del cuore di sistema operativo anche più volte al giorno e in breve si viene a creare una diffusa comunità di sviluppatori che iniziano a lavorare in parallelo al debug, ovvero alla correzione degli errori, e allo sviluppo del sistema operativo. Insomma, rispetto alla cattedrale rappresentata dal tradizionale metodo di sviluppo di altri applicativi, in cui poche persone avevano in mano la gestione di tutta l'architettura del programma, si viene a sviluppare un bazaar di persone, stili, idee, apporti diversi, in cui una rete in gran

parte decentrata di persone mettono mano a un progetto. Ciò consente di sviluppare con grande velocità un programma, di ripulirlo dagli errori, di integrarvi idee nuove e differenti, di migliorarlo sostanzialmente, in tempi e con costi ridotti, ma ha un limite. Come nota correttamente Raymond, un sistema decentrato non è in grado di costruire un progetto dal nulla: deve avere già qualcosa con il quale far lavorare gli sviluppatori sparsi per la rete, vi deve essere un'idea di base forte, una prospettiva plausibile e un progetto già strutturato sulla quale una comunità possa trovare un punto di coesione e confrontarsi. È così che è nato, per esempio, un sistema di grande successo come Linux, creato partendo da Minix e diventato tutt'altro grazie al lavoro di una vasta comunità di sviluppatori.

Tutto questo, però, ci porta a ulteriori considerazioni. Il modello di distribuzione Open Source consente a sua volta un modello di sviluppo notevolmente accelerato e decentrato, in cui non esiste un "Dipartimento di sviluppo" perfettamente localizzato in qualche ufficio, né esistono gerarchie di controllo sclerotizzate le quali mantengono un rigido controllo sul codice elaborato. Il passaggio da un mantainer all'altro, ovvero da un responsabile dello sviluppo all'altro, è più fluido, forse fin troppo, notevolmente semplificato dalla natura non ritentiva del modello a "sorgenti aperti". D'altro canto, il fenomeno Linux, come esempio di sviluppo a bazaar, non sarebbe stato possibile senza l'avvento e la diffusione prepotente di un medium di comunicazione capillare e a basso costo come Internet, che grazie alle sue varie applicazioni, posta elettronica, siti Web, cvs e molto altro ancora, ha consentito, e tutt'ora consente, agli sviluppatori di intrattenere quello scambio di informazioni e materiali vitali per costruire e mantenere l'identità di comunità degli sviluppatori legati a un progetto. Bisogna, però, mettere un freno al mito del bazaar: non è effettivamente possibile che non esista una gerarchia, per quanto fluida essa sia, che mantenga il controllo sulla direzione dello sviluppo, su ciò che possa essere accettato o meno nel codice. Una relazione uno a molti, o pochi a molti, in cui un numero ristretto di persone gestisca gli apporti di molti sviluppatori in una struttura a forte connotazione orizzontale, rischia di essere funzionale solo su progetti di dimensioni ridotte. E non basta il contrappeso di un progetto iniziale fortemente strutturato, di prospettive

limitate ma plausibili, a rendere più coerente il modello di sviluppo.

E il “business”? Una delle idee persistenti legate all’Open Source è che questo precluda la possibilità di fondare un’economia legata al software che si produce, che letteralmente renda vacanti dei posti di lavoro. Ma a cosa noi siamo abituati? Se dovessimo richiamare le vecchie, ma non per questo del tutto errate, categorie marxiste, potremmo dire che il valore del software è costituito dal lavoro che è stato impiegato per crearlo, aggiungiamoci i costi accessori, il mitico plusvalore e avremo il valore finale del programma che vogliamo commercializzare. Nel caso di un software Open Source, per il quale non siamo costretti a pagare il lavoro di tutti coloro che hanno in vario modo, direttamente o indirettamente, all’applicazione che ci troviamo fra le mani, come fare di tutto ciò un guadagno? Iniziamo con il liberare il campo da una incomprensione che spesso condiziona i giudizi su una possibile economia dell’Open Source: chi opera su codici aperti non è per questo costretto a rilasciare pubblicamente e gratuitamente il proprio lavoro. Le licenze del software libero sono diverse e vanno dalla variante “virale” chiamata GPL (General Public Licence), che costringe chiunque crei un programma derivato da sorgenti protetti da questa licenza a redistribuirlo negli stessi termini (e quindi la GPL letteralmente si autoapplica in maniera virale a ogni lavoro da essa “toccato”), alla Licenza MIT che consente di “to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software”, quindi anche espressamente di vendere copie del software. In effetti, il codice Open Source non è necessariamente né gratuito, né privo di diritto di autore: vi è una serie di licenze in grado di proteggere i diritti di chi ha lavorato sia sul software originale che sui programmi derivati e vi sono licenze che non limitano la possibilità di modificare applicazioni e di vendere il proprio lavoro. Ma l’Open Source pone in primo piano anche un diverso modello di mercato, nel quale il valore del lavoro non viene più posto nella merce come prodotto a sé stante, ma come servizio: se il software si svuota di un valore commerciale diretto, riporta all’individuo il valore del lavoro, ovvero rende l’attività diretta di consulenza, di vero e proprio servizio all’utente finale, la fonte di

un'economia sostenibile. Il programma Open Source ha indubbiamente, nella maggior parte dei casi, un minore costo iniziale di installazione, derivabile dalla mancanza di un corrispettivo economico legato alle licenze applicate, ma necessitano comunque di un'attività di installazione, configurazione, modifica delle funzionalità originarie, che possono essere affidate a singoli o a società. Ciò deriva anche dalla stessa natura del software Open Source: difficilmente vi troviamo alle spalle una società che ne concentra lo sviluppo, unica depositaria della conoscenza dei sorgenti del programma, ma anche unica in grado di offrire direttamente un'opera di formazione, assistenza e programmazione accessoria. Sorgenti aperti, in definitiva, consentono a chiunque di accedere alla conoscenza di un applicativo, formarsi su di esso, offrire consulenze, modificarlo secondo le esigenze dell'utente finale e, aspetto non trascurabile, farsi pagare per l'apporto del proprio lavoro.

Infine, un ulteriore aspetto dei sorgenti aperti che ci sta particolarmente a cuore è la libertà personale che questi proteggono. Con l'imperversare di troiani, malware e mille altre diavolerie nascoste dentro ai binari delle più innocenti applicazioni, ma anche con l'interesse sempre maggiore da parte di soggetti istituzionali ed economici verso l'identificazione degli utenti di particolari programmi o servizi, l'aprire i sorgenti allo sguardo di chiunque garantisce tutti nei confronti degli interessi di qualcuno.

Giorgio Zarrelli

zarrelli@linux.it