Prefazione

In quanto revisore, ho avuto la possibilità di leggere in anteprima il libro che avete fra le mani, ed era grande, già in bozza. Dave Thomas e Andy Hunt hanno qualcosa da dire, e sanno come dirlo. Ho visto quello che stavano facendo e sapevo che avrebbe funzionato. Così ho chiesto di scrivere questa prefazione, per poter spiegare perché.

Detto semplicemente, questo libro racconta come programmare in un modo che si può seguire. Forse non penserete che sia una cosa difficile da fare, ma lo è. Perché? Da un lato, non tutti i libri di programmazione sono scritti da programmatori. Molti sono compilati dai progettisti dei linguaggi, o dai giornalisti che lavorano con loro per promuovere le loro creazioni. Quei libri vi dicono come *parlare* in un linguaggio di programmazione, il che è certamente importante ma è solo una piccola parte di quel che fa un programmatore. Che cosa fa un programmatore oltre a parlare in un linguaggio di programmazione? Beh, questa è una faccenda più profonda. La maggior parte dei programmatori avrebbe qualche difficoltà a spiegare che cosa fa. La programmazione è un lavoro pieno di dettagli, e tener traccia di tutti quei dettagli richiede grande attenzione. Le ore passano e il codice compare. Lo guardate e ci sono tutti quegli enunciati. Se non pensate con grande attenzione, potreste credere che la programmazione sia solo scrivere enunciati in un linguaggio di programmazione. Avreste torto, ovviamente, ma non riuscireste a stabilirlo guardandovi intorno nella sezione "programmazione" della libreria.

In questo libro, Dave e Andy ci dicono come programmare, e lo dicono in un modo che possiamo seguire. Come sono riusciti a essere così abili? Non sono concentrati sui dettagli come gli altri programmatori? La risposta è che hanno fatto attenzione a quello che facevano mentre lo facevano, e poi hanno cercato di farlo meglio.

Immaginatevi di partecipare a una riunione. Magari state pensando che la riunione potrebbe andare avanti all'infinito e che preferireste programmare. Dave e Andy penserebbero perché stanno partecipando a una riunione, e si chiederebbero se ci sia qualche altra cosa che potrebbero fare che prenda il posto della riunione, e cercherebbero di decidere se quel qualcosa potrebbe essere automatizzato in modo che il lavoro della riunione "si faccia da solo" in futuro. Poi lo farebbero.

Questo è il modo in cui pensano Dave e Andy. Quella riunione non era una cosa che impedisse loro di programmare: era programmazione. Ed era programmazione che poteva essere migliorata. So che pensano in questo modo, perché è il loro secondo suggerimento: "Pensate al vostro lavoro".

Hanno pensato in questo modo per anni: come potete immaginare, presto hanno avuto una bella raccolta di soluzioni. Ora immaginateveli che usano quelle soluzioni nel loro lavoro per qualche anno ancora e buttano via quelle troppo pesanti o che non danno sempre buoni risultati. Beh, questo modo di procedere spiega sostanzialmente che cosa voglia dire *pragmatico*. Ora immaginateli che si prendono ancora un anno o due per scrivere le loro soluzioni. Penserete: *quelle informazioni sarebbero una miniera d'oro*. E avreste ragione. Gli autori ci dicono come programmano e ce lo dicono in un modo che possiamo seguire. Ma in questa affermazione c'è più di quello che forse pensate. Mi spiego.

Gli autori sono stati ben attenti a evitare di proporre una teoria dello sviluppo di software. È una fortuna, perché altrimenti sarebbero stati costretti a trasformare ogni capitolo per difendere la loro teoria. Il che sarebbe nella tradizione, per esempio, delle scienze fisiche, dove le teorie alla fine diventano leggi o vengono scartate senza clamore. La programmazione invece ha poche leggi (o forse nessuna). Perciò un consiglio di programmazione formulato in base a qualche presunta legge potrà suonare bene scritto, ma non soddisfa mai nella pratica. È quello che non va in tanti libri che parlano di metodologia.

Ho studiato il problema per una dozzina d'anni e ho trovato che la cosa più promettente era un dispositivo chiamato *linguaggio a pattern*. In breve, un *pattern* è una soluzione, e un linguaggio a pattern è un sistema di soluzioni che si rafforzano a vicenda. Si è formata un'intera comunità attorno alla ricerca di questi sistemi.

Questo libro è più di una raccolta di suggerimenti. È un linguaggio a pattern travestito. Lo dico perché ogni suggerimento è ricavato dall'esperienza, formulato come consiglio concreto e messo in rapporto con gli altri a formare un sistema. Sono le caratteristiche che ci permettono di apprendere e di seguire un linguaggio a pattern, e funzionano nello stesso modo qui.

Potete seguire i consigli di questo libro perché sono concreti. Non troverete astrazioni vaghe. Dave e Andy scrivono direttamente per voi, come se ogni suggerimento fosse una strategia vitale per infondere energia nella vostra carriera di programmatori. Restano semplici, raccontano una storia, usano un po' di umorismo e poi propongono le risposte alle domande che vi sorgono mentre provate.

E non è finita qui. Dopo aver letto dieci o quindici suggerimenti comincerete a vedere un'ulteriore dimensione di questo lavoro. A volte la chiamiamo QWAN, che sta per *quality without a name*, qualità senza nome. Il libro ha una filosofia che filtrerà e si mescolerà con la vostra coscienza. Non tiene sermoni; vi dice semplicemente quello che funziona. Ma nel dirlo passa anche molto altro. È la bellezza del libro: incarna la propria filosofia, e lo fa senza presunzione.

Perciò, eccolo qui: un libro completo sulla programmazione, facile da leggere e da mettere in pratica. Ho speso una gran quantità di tempo per spiegare il motivo che lo rende così speciale, ma a voi probabilmente interesserà solo che funzioni. Ed è così. Provare per credere.

Ward Cunningham