Prefazione alla prima edizione

Il "refactoring" è stato concepito nella cerchia dello Smalltalk, ma non c'è voluto molto perché entrasse anche in altri ambiti dei linguaggi di programmazione. Poiché il refactoring è parte integrante dello sviluppo di framework, il termine spunta rapidamente quando i "frameworker" parlano della loro arte. Spunta quando rifiniscono le loro gerarchie di classi e quando si vantano del numero di righe di codice che sono stati capaci di eliminare. I frameworker sanno che un framework non sarà perfetto al primo colpo: deve evolvere mentre loro acquisiscono esperienza. Sanno anche che il codice verrà letto e modificato più spesso di quanto non venga scritto. La chiave per mantenere il codice leggibile e modificabile è il refactoring, in particolare per i framework ma anche per tutto il software in generale.

Qual è allora il problema? Semplicemente questo: il refactoring è rischioso. Comporta modifiche al codice che possono introdurre errori sottili. Il refactoring, se non effettuato correttamente, può far perdere giorni, se non addirittura settimane e diventa ancora più rischioso quando viene praticato informalmente o *ad hoc.* Si comincia a scavare nel codice; presto si scoprono nuove opportunità di cambiamento e si scava ancora più a fondo; quanto più si scava, tante più cose si trovano. Alla fine si scava un buco da cui non si è più in grado di uscire. Per evitare di scavarsi la fossa, il refactoring va effettuato in modo sistematico. Quando i miei coautori e io abbiamo scritto *Design Patterns*, abbiamo detto che i pattern di design sono obiettivi per il refactoring. Identificare gli obiettivi però è solo una parte del problema; trasformare il codice nel modo giusto è un altro paio di maniche.

Martin Fowler e gli autori che hanno collaborato al volume danno un contributo prezioso al software orientato agli oggetti gettando luce sul processo di refactoring. Questo libro spiega i principi e le buone pratiche del refactoring e mostra quando e dove iniziare a scavare nel codice per migliorarlo. Il cuore del libro è un ampio catalogo di rifattorizzazioni. Ciascuna descrive la motivazione e la meccanica di una trasformazione collaudata del codice. Alcune rifattorizzazioni, come Extract Method o Move Field, possono sembrare ovvie.

Non fatevi ingannare, però. Comprendere la meccanica di queste rifattorizzazioni è la chiave per il refactoring disciplinato. Le rifattorizzazioni in questo libro vi aiuteranno a modificare il vostro codice un piccolo passo alla volta, riducendo così i rischi dell'evoluzione del vostro design. In poco tempo aggiungerete queste rifattorizzazioni e i loro nomi al vostro vocabolario di sviluppo.

La mia prima esperienza con il refactoring rigoroso, "un passo alla volta", è stata quando programmavo in coppia con Kent Beck. Kent ha preteso che applicassimo le rifattorizzazioni dal catalogo di questo libro un passo alla volta e sono rimasto stupito da quanto questo esercizio funzionasse bene. Non solo è aumentata la mia fiducia nel codice risultante, mi sono sentito anche meno stressato. Vi consiglio caldamente di provare: voi e il vostro codice vi sentirete molto meglio.

Erich Gamma, Object Technology International, Inc.
gennaio 1999