Introduzione

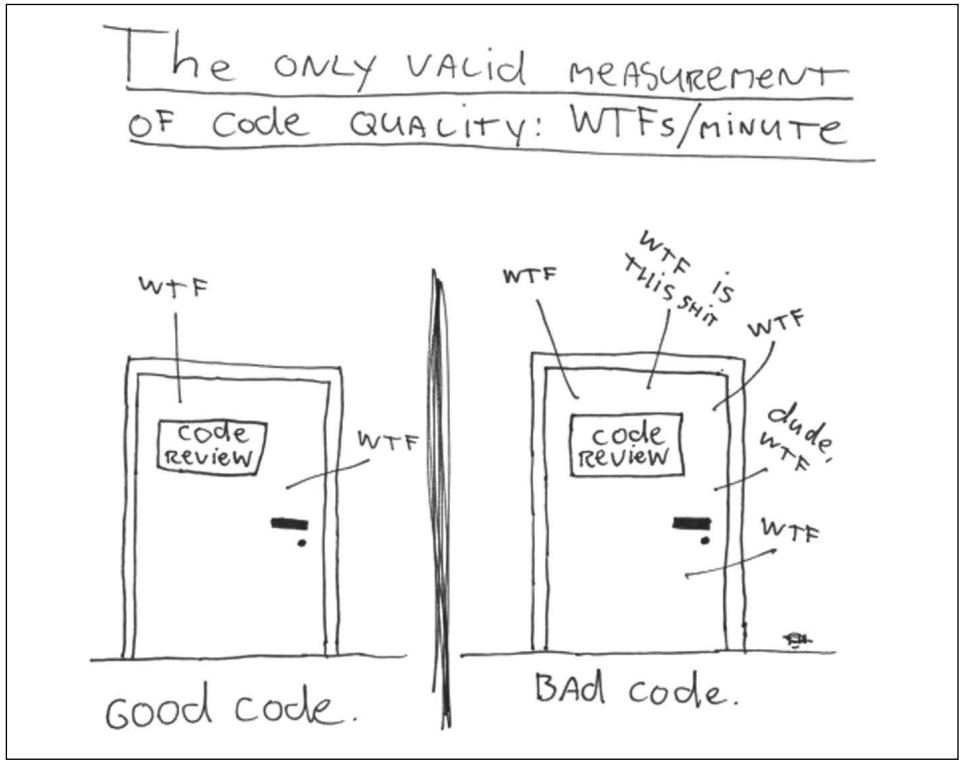

Figura I.1 Riprodotta per gentile concessione di Thom Holwerda: http://www.osnews.com/story/19266/WTFs_m.

Quale porta rappresenta il vostro codice? Quale porta rappresenta il vostro team o la vostra azienda? Perché siamo chiusi dentro quella stanza? È solo una normale revisione di codice o abbiamo trovato un'intera sequenza di terribili problemi poco dopo l'uscita? Stiamo eseguendo il debugging nel panico, vagando nel codice sul quale abbiamo lavorato così duramente? I clienti se ne stanno andando a frotte e i manager ci stanno

col fiato sul collo? Come possiamo assicurarci di trovarci dietro la porta giusta quando il gioco si fa duro? La risposta è: l'esperienza.
Sono due i componenti che permettono di acquisire esperienza: la conoscenza e il lavoro. Occorre la conoscenza dei principi, dei pattern, delle tecniche e delle euristiche "artigianali"; e poi occorre anche mettere in pratica tale conoscenza, lavorando duramente e provando.
Potrei insegnarvi tutta la fisica su cui si basa l'andare in bicicletta. In effetti, gli aspetti matematici sono relativamente semplici. La gravità, l'attrito, il momento angolare, il centro di massa e così via, sono tutti aspetti illustrabili in una paginetta di equazioni. Sulla base di queste formule potrei dimostrarvi che andare in bicicletta è possibile e fornirvi tutta la conoscenza necessaria per farlo. E inevitabilmente, la prima volta che provaste a farlo, cadreste.
Anche programmare non è differente. Potremmo scrivere tutti i "migliori" principi di programmazione (*clean code*) e forse pensereste di poter procedere (in altre parole, vi faremmo solo cadere dalla bicicletta), ma allora che razza di insegnanti saremmo, e quale utilità avrebbe tutto ciò per voi studenti?
No. Questa non è la strada seguita da questo libro.
Imparare a scrivere codice "pulito", *clean code*, non è facile. Richiede qualcosa di più della semplice conoscenza di principi e tecniche. Ci vuole anche il "sudore". Dovete fare pratica, e anche andare incontro a insuccessi. Dovete osservare gli altri mentre provano e falliscono. Dovete vederli inciampare e tornare sui loro passi. Dovete vederli soffrire per le loro decisioni e vedere il prezzo che pagano per aver preso decisioni sbagliate.
Preparatevi a lavorare duramente mentre leggerete questo libro. Questo non è un "libro facile" che potete leggere in aereo e terminare prima dell'atterraggio. Questo libro vi farà lavorare, ma lavorare duramente. Quale tipo di lavoro dovrete svolgere? Leggerete del codice, grandi quantità di codice. E vi verrà chiesto di riflettere su quello che c'è di giusto e quello che c'è di sbagliato in tale codice. Vi verrà chiesto di seguire i passaggi mentre vengono estratti dei moduli, che poi vengono ricombinati in modo differente. Questo richiederà tempo e impegno; ma pensiamo che ne valga la pena.
Abbiamo diviso questo libro in tre parti. I primi capitoli descrivono i principi, i pattern e le tecniche del codice pulito. Vi troverete una grande quantità di codice, e riuscire a comprenderlo non sarà sempre facile. Questa prima parte vi prepara alla seconda. E se doveste arrendervi dopo aver letto i primi capitoli… buona fortuna a voi!
La seconda parte del libro è più solida. È costituita da numerosi casi di studio di complessità crescente. Ogni caso di studio è un esercizio di "pulizia" di un frammento di codice o di trasformazione di codice problematico in codice meno problematico. L'attenzione al dettaglio sarà notevole. Vi troverete a scorrere, avanti e indietro, fra descrizione e i listati. Dovrete analizzare e comprendere il codice sul quale stiamo lavorando e seguire i ragionamenti proposti e le motivazioni che hanno spinto a effettuare ogni singola modifica. Dedicategli del tempo, perché l'impegno vi richiederà giorni.
La terza parte di questo libro è il lascito finale. Si tratta di un unico capitolo contenente un elenco di euristiche e intuizioni raccolte nel corso della creazione dei casi di studio. Mentre elaboravamo e raffinavamo il codice dei casi di studio, abbiamo documentato ogni motivazione delle nostre azioni sotto forma di euristiche e avvertenze. Abbiamo tentato di studiare le nostre stesse reazioni al codice che stavamo leggendo e modificando, e abbiamo cercato di catturare il perché abbiamo ragionato in un certo modo e abbiamo

agito di conseguenza. Il risultato è una base di conoscenze che descrive il modo in cui ragioniamo quando scriviamo, leggiamo e raffiniamo il codice.
Questa base di conoscenze non avrà particolare valore se non avrete svolto il lavoro di leggere con cura i casi di studio presentati nella seconda parte di questo libro, nei quali abbiamo annotato con cura ogni modifica apportata, con riferimenti in avanti alle euristiche. Questi riferimenti in avanti sono specificati fra parentesi quadre nel seguente modo: [H22]. Ciò vi permette di vedere il contesto in cui tali euristiche sono state applicate e scritte! Le euristiche, da sole, non sono sufficientemente utili: a contare davvero è la relazione fra tali euristiche e le singole decisioni che abbiamo preso mentre raffinavamo il codice dei casi di studio.
Per agevolarvi ulteriormente a proposito di queste relazioni, abbiamo previsto a fine libro un indice di questi riferimenti. Potete quindi usarlo per trovare agevolmente la posizione in cui è stata applicata una determinata euristica.
Se leggeste solo la prima e la terza sezione, saltando del tutto i casi di studio, vi ritrovereste a leggere l'ennesimo libro "leggero" sulla scrittura del buon software. Se invece dedicherete un tempo adeguato ai casi di studio, seguendo ogni singolo passo, ogni più piccola decisione, se vi metterete nei nostri panni e vi costringerete a seguire i nostri stessi percorsi mentali, acquisirete una comprensione molto più ricca di questi principi, pattern, tecniche ed euristiche. Non si tratterà più di una conoscenza "leggera". Vi sarà entrata nelle dita, negli occhi e nella mente. Diventerà un automatismo, così come lo è andare in bicicletta.

Ringraziamenti

Grazie alle mie due illustratrici, Jeniffer Kohnke e Angela Brooks. Jennifer è autrice delle simpatiche e creative immagini che aprono ogni capitolo e anche dei ritratti di Kent Beck, Ward Cunningham, Bjarne Stroustrup, Ron Jeffries, Grady Booch, Dave Thomas, Michael Feathers e mio.
Angela, invece, è autrice delle illustrazioni interne dei capitoli. Aveva già preparato per me varie immagini nel corso degli anni, fra le quali molte immagini presenti nel libro *Agile Software Develpment: Principles, Patterns, and Practices*. È anche la mia primogenita, della quale sono molto orgoglioso.
Un ringraziamento particolare va ai miei revisori, Bob Bogetti, George Bullock, Jeffrey Overbey e in particolare Matt Heusser. Sono stati brutali, crudeli, implacabili. Grazie a ciò mi hanno permesso di apportare i miglioramenti necessari.
Ringrazio il mio editor, Chris Guzikowski, per il suo sostegno, incoraggiamento e per la sua giovialità. Grazie anche al personale editoriale di Pearson, fra cui Raina Chrobak, per avermi permesso di essere tranquillo e puntuale.
Grazie a Micah Martin e a tutto il gruppo di *8th Light* (`www.8thlight.com`) per le loro indicazioni e il loro incoraggiamento.
Grazie agli *Object Mentors*, di ieri, di oggi e di domani, fra i quali: Bob Koss, Michael Feathers, Michael Hill, Erik Meade, Jeff Langr, Pascal Roy, David Farber, Brett Schuchert, Dean Wampler, Tim Ottinger, Dave Thomas, James Grenning, Brian Button, Ron Jeffries, Lowell Lindstrom, Angelique Martin, Cindy Sprague, Libby Ottinger, Joleen Craig, Janice Brown, Susan Rosso e tanti altri.

Grazie a Jim Newkirk, mio amico e partner professionale, che mi ha insegnato molto di più di quanto possa immaginare. Grazie a Kent Beck, Martin Fowler, Ward Cunningham, Bjarne Stroustrup, Grady Booch e a tutti gli altri miei mentori, compatrioti e colleghi. Grazie a John Vlissides per esserci stato nei momenti che contano. Grazie ai ragazzi di Zebra per aver sopportato i miei vaneggiamenti sulla lunghezza "giusta" di una funzione. E, infine, grazie a voi per aver letto tutti questi ringraziamenti.