Premessa

Il titolo di questo libro è *Clean Architecture*. Un nome audace. Qualcuno direbbe perfino un nome arrogante. Allora perché ho scelto questo titolo e perché ho scritto questo libro? Ho scritto la mia prima riga di codice nel 1964, avevo 12 anni. A tutt'oggi, quindi, posso dire di aver scritto codice per più di mezzo secolo. In quest'arco di tempo, ho appreso alcune cose sul modo in cui strutturare i sistemi software, cose che, credo, anche altri troveranno probabilmente utili.

Ho imparato queste cose realizzando molti sistemi, grandi e piccoli. Ho realizzato alcuni piccoli sistemi embedded e alcuni grossi sistemi di elaborazione batch. Ho realizzato sistemi operanti in tempo reale e sistemi web. Ho realizzato app da console, app GUI, app di controllo dei processi, giochi, sistemi di contabilità, sistemi per telecomunicazioni, strumenti di progettazione, app di disegno e molto, molto altro.

Ho realizzato app mono-thread e multi-thread, app che gestivano pochi processi pesanti, app che avevano molti processi leggeri, app multiprocessore, app per database, app matematiche, app per il calcolo geometrico e molto, molto altro.

Ho realizzato molte app. Ho prodotto molti sistemi. E da tutti questi e prendendoli tutti in considerazione, ho appreso una vera lezione…

Le regole dell'architettura sono sempre le stesse!

Questo è davvero sorprendente, perché i sistemi che ho realizzato sono stati tutti radicalmente differenti. E allora perché sistemi così differenti condividono regole simili in termini di architettura? La mia conclusione è che le regole dell'architettura del software siano indipendenti da ogni altra variabile.

Questo è ancora più sorprendente se si considerano i cambiamenti intercorsi nell'hardware nel corso di questo stesso mezzo secolo. Ho iniziato a programmare su macchine delle dimensioni di un grosso frigorifero con un clock di mezzo megahertz, 4 KB di memoria centrale, 32 KB di spazio su disco e un'interfaccia a telescrivente da 10 caratteri al secondo. Oggi sto scrivendo questa prefazione su un autobus nel corso di un tour in Sudafrica su un MacBook con un i7 a quattro core, operanti ognuno a 2,8 GHz. Ho 16 GB di RAM, 1 TB di SSD e uno schermo Retina da 2880 × 1800 ad altissima definizione. La differenza in termini di potenza di calcolo è strabiliante. Qualsiasi analisi dimostrerebbe che questo MacBook è potente almeno 10^{22} volte i computer sui quali ho iniziato a lavorare mezzo secolo fa.

Ventidue ordini di grandezza sono un numero molto grande. È la distanza in angstrom dalla Terra ad Alfa-Centauri. È il numero di elettroni della materia che compone il vostro

borsellino, in tasca o in borsetta. E tuttavia tale numero è (come minimo) l'incremento di potenza di calcolo cui ho assistito nel corso della mia vita.

E dato questo immenso cambiamento in termini di potenza di calcolo, quale è stato l'effetto sul software che ho scritto? Certamente è cresciuto di dimensioni. Un tempo, 2000 righe rappresentavano un grosso programma. Dopotutto, si trattava di una scatola di schede perforate del peso di quasi 5 kg. Oggi, al contrario, un programma non si può dire "grosso" se non supera almeno le 100.000 righe.

Il software, inoltre, può fare molte più cose. Oggi possiamo fare cose che non ci saremmo neanche sognati negli anni Sessanta. I romanzi di fantascienza *Colossus* (1966), *La Luna è una severa maestra* (1966) e *2001: Odissea nello spazio* (1968) hanno tutti tentato di immaginare il nostro futuro, ma tutti hanno mancato il bersaglio piuttosto significativamente. Tutti hanno immaginato delle grandi macchine che acquisivano coscienza. Quello che abbiamo, invece, sono macchine sorprendentemente piccole che fanno… le macchine.

E vi è un'altra cosa da dire sul software di oggi, rispetto al software di allora: è fatto degli stessi componenti. Vi si trovano istruzioni `if`, istruzioni di assegnamento e cicli `while`.

Potreste obiettare che oggi abbiamo linguaggi molto migliori e paradigmi superiori. Dopotutto, programmiamo in Java o C# o Ruby e usiamo la progettazione a oggetti. Vero. Purtuttavia il codice resta comunque un assieme di sequenze, selezioni e iterazioni, esattamente quello che era negli anni Sessanta e Cinquanta.

Quando si osserva davvero attentamente la pratica della programmazione dei computer, ci si rende conto che ben poco è cambiato negli ultimi cinquant'anni. I linguaggi sono un po' migliorati. Gli strumenti di oggi sono fantastici. Ma i blocchi costruttivi di base di un programma non sono cambiati.

Se prendessi una programmatrice (molto probabilmente sarebbe una ragazza, dal momento che, allora, la programmazione era un'attività prevalentemente femminile) e dal 1966 la trasportassi nel 2016 e le mettessi davanti il mio MacBook con IntelliJ e le mostrassi Java, le basterebbero 24 ore per riprendersi dallo shock. Ma poi sarebbe in grado di programmare. Java non è poi così diverso dal C o anche dal Fortran.

E se vi trasportassi nel 1966 e vi mostrassi come scrivere e modificare del codice per un PDP-8, perforando un nastro di carta su una telescrivente da 10 caratteri al secondo, vi basterebbero 24 ore per riprendervi dallo sconforto. Ma poi sareste in grado di programmare. Il codice non è cambiato poi così tanto.

Questo è il segreto: questa immutabilità del codice è il motivo per il quale le regole dell'architettura del software sono così coerenti, indipendentemente dal tipo di sistema. Le regole dell'architettura del software sono le regole in base alle quali si dispongono e assemblano i blocchi costruttivi dei programmi. E poiché tali blocchi costruttivi sono universali e non sono cambiati, anche le regole per disporli sono altrettanto universali e non sono cambiate.

I giovani programmatori potranno anche non crederci. Potrebbero insistere sul fatto che oggi tutto è cambiato, che è molto differente, che le regole del passato sono ormai tramontate. Se questo è ciò che pensano, purtroppo si sbagliano. Le regole non sono cambiate. Nonostante tutti i nuovi linguaggi e tutti i nuovi framework e tutti i paradigmi, le regole di oggi sono le stesse impiegate da Alan Turing quando scrisse il suo primo codice macchina, nel 1946.

Ma una cosa è cambiata: a quel tempo non conoscevamo bene le regole. Di conseguenza, le violavamo, più e più volte. Oggi, con mezzo secolo di esperienza in più, conosciamo un po' meglio le buone regole.

Ed è di tali regole, senza tempo, immutabili, che parla questo libro.