





# Parte II

# Design pattern

“Once you understand the design patterns and have had an ‘Aha!’ experience with them, you won’t ever think about object-oriented design in the same way.”

(Una volta che i pattern di design siano stati compresi e sperimentati al punto di dire “Aha!”, non sarà più possibile pensare il design object-oriented nella maniera in cui lo si faceva precedentemente.)

ERICH GAMMA, RICHARD HELM,  
RALPH JOHNSON, JOHN VLISSIDES







# Capitolo 7

## Introduzione all'idea di pattern

Nei capitoli di questa seconda parte vediamo finalmente di approfondire alcuni concetti legati alla concezione e all'uso dei design pattern.



### "Il" libro

Il libro di riferimento "storico" e più importante per tutti gli studi attorno ai design pattern è senza dubbio quello pubblicato nel 1995 da Gamma, Helm, Johnson e Vlissides [Gamma-1995].

Oltre a un'introduzione pratica, vengono mostrati, commentati e contestualizzati 23 diversi modelli di progettazione (*design pattern*, appunto), frutto dell'esperienza e della conoscenza del codice di grossi progetti acquisita dai quattro autori.

L'idea alla base di tale impostazione è quella che ogni sviluppatore deve poter sfruttare non solo la propria conoscenza ed esperienza nel design e nel redesign, ma anche quella acquisita da altri sviluppatori su grandi progetti. Il catalogo serve proprio a questo: a condividere l'esperienza di design.

Si torna quindi a parlare di esperienza. Ho già avuto modo di sottolineare come capitalizzare esperienza sia fondamentale nel design e nello sviluppo software. I design pattern servono a



questo, ma non solo. Lo studio dei pattern permette di capire a fondo la programmazione a oggetti. E questa è una motivazione più che sufficiente per parlare di pattern.

I design pattern da soli spiegano perché la programmazione a oggetti è stata un passo in avanti rispetto ai paradigmi precedenti. Dimostrano cos'è la genericità e come va sfruttato al meglio il polimorfismo.

#### Studio dei pattern di design

Nei primi anni in cui introducevo questo tema ai miei studenti, i commenti alla fine del semestre erano di questo tipo: "Non sono sicuro di aver afferrato esattamente l'importanza dei pattern, ma cercando di studiarli ho capito molto meglio quali sono gli aspetti importanti della programmazione a oggetti".

Non ero sicuro, allora, se essere soddisfatto o meno dell'osservazione. Da una parte si riconosceva l'utilità di parlare di pattern e di design all'interno di un corso di ingegneria del software, dall'altra arrivava la conferma che alla fine del corso agli studenti mancava ancora qualcosa per poter essere veramente appagati. Infatti, pur riconoscendo al corso sui pattern il merito di approfondire gli aspetti della programmazione a oggetti, non era quello l'unico scopo delle lezioni.

In realtà, mi sono accorto col tempo che i pattern vengono capiti. Ciò che invece manca, dopo un corso del genere, è l'esperienza del loro utilizzo. Sapere quando e in che contesto va utilizzato un modello piuttosto di un altro. Si torna quindi al tema dell'esperienza, già accennato in precedenza. Esperienza che va acquisita solo con la pratica e con il tempo.

## Visione del software

I pattern hanno contribuito a dare una nuova visione del software e a rendere l'attività dello sviluppo software in un certo senso più "nobile". Hanno cioè permesso di elevare il livello del discorso, pur mantenendo, anzi aumentando, l'importanza della programmazione, che all'interno di un progetto software non può in nessun caso essere considerata "manovalanza".

La concezione molto gerarchica della gestione di un progetto, facilitata da modelli di progettazione molto sequenziali e dal modesto livello di formazione dei programmatori, considerati meri esecutori del progetto, fa spazio a una suddivisione più orizzontale, in cui ogni partecipante si occupa di design, programmazione e test di unità. Ogni collaboratore contribuisce in modo completo alla definizione, pianificazione e realizzazione del progetto.

#### Prime bozze

Attraverso Marc Domenig, mio attuale CEO alla Canoo, con il cui gruppo di ricerca ho lavorato all'Università di Basilea quando lui era professore all'Istituto di Informatica, ho avuto il privilegio di ricevere una delle prime bozze del libro sui pattern: credo che fosse nel 1993. Erich Gamma aveva lavorato con Marc durante il suo dottorato all'università di Zurigo ed erano rimasti in contatto anche dopo che Erich si era trasferito temporaneamente negli USA al progetto Taligent. Ricevetti quindi una copia di tali bozze da Marc, a cui erano state inviate affinché le leggesse e commentasse. Ricordo che inizialmente non compresi l'importanza di certi argomenti. Il solo

fatto che l'avesse scritto Erich (già conosciuto per il progetto ET++) mi convinse che doveva contenere argomenti rilevanti. Ciò mi permise di leggerlo più volte e di capire, infine, la vera portata dei temi, a dimostrazione che a volte, per un primo approccio a un nuovo tema, serve più fidarsi della bontà dell'autore che del proprio interesse sul contenuto.

Lo stesso utilizzo dei pattern a livello di progettazione ha subito delle evoluzioni dalla sua prima introduzione, soprattutto se consideriamo il fenomeno dal punto di vista dei metodi di programmazione agile, quelli cioè che consideriamo come riferimento in questo libro.

Una delle grosse differenze dell'approccio agile rispetto a un approccio più tradizionale (come ad esempio quello descritto dal modello a cascata) nei confronti della progettazione, consiste nel rivedere continuamente design e implementazione per integrare in modo continuo nuovi requisiti. Se, nell'approccio tradizionale, il primo design è il punto centrale di tutta la progettazione e va quindi studiato a fondo nei minimi dettagli prima di passare alla fase di programmazione, nell'approccio agile non è più così.

I metodi cosiddetti agili, suggeriscono di partire da una funzionalità minima, costruiscono il design e l'implementazione per questa funzionalità minima e solo in un secondo tempo, con cicli molto brevi, introducono man mano nuove funzionalità, frutto anche di nuovi requisiti, adattando il design alle mutate esigenze, gestendo l'evoluzione del codice e della struttura di classi con metodi di refactoring e test.

Senza voler entrare troppo nei dettagli della filosofia che sta alla base dei metodi agili, su cui esistono ottimi libri soprattutto per quanto riguarda XP (eXtreme Programming), possiamo affermare che questi hanno conseguenze sull'utilizzo dei pattern in fase di progettazione.

Se, con i metodi tradizionali, si cerca di determinare l'intera architettura (e quindi anche i pattern coinvolti) già nella prima fase di progettazione, con quelli agili si tende ad inserire i pattern a mano a mano nell'architettura che evolve.

Vengono con maggiore frequenza introdotti nell'architettura durante i passaggi di refactoring, proprio perché è durante le considerazioni che si rendono necessarie per l'aggiunta di nuove funzionalità che si scoprono elementi di generalizzazione, di condivisione, di flessibilità, e così via; in altre parole si scopre come il design dovrebbe evolvere nel modo migliore. La conferma di questo si intuisce dal fatto che i metodi di refactoring più complessi (e più interessanti), hanno come obiettivo l'introduzione di un design pattern.

Del resto, proprio usando il modello tradizionale si scopre come sia difficile "indovinare" al primo colpo un design flessibile e riutilizzabile. Con un approccio diverso, per certi versi minimalista, si mette al centro del processo di progettazione e sviluppo software il refactoring, con il relativo redesign.

## Flessibilità

L'inserimento di pattern durante la fase di redesign (e di design, naturalmente: nessuno impedisce di farlo se la direzione è già chiara fin dall'inizio) serve a stabilizzare il design e a introdurre flessibilità e genericità attorno alle mini architetture rappresentate dai pattern.

## Definizione

I design pattern descrivono soluzioni semplici ed eleganti per problemi specifici della progettazione object-oriented.

Questa può essere considerata una prima definizione di design pattern. Altre seguiranno nel corso di questa seconda parte a loro dedicata.

## Documentazione

I design pattern sono senza dubbio un importante elemento di comprensione e di chiarezza del design.

Un ulteriore aspetto interessante è però quello della documentazione. L'utilizzo di pattern ci permette infatti di documentare il design direttamente nel codice, evitando lunghi commenti di documentazione che rischiano di essere vecchi già dopo le prime modifiche. Definiscono infatti un vocabolario per parlare e discutere di design.

Anche questo aspetto si adatta bene alla filosofia della progettazione agile, che vede il codice e la capacità di modificare il codice al centro del processo di progettazione e sviluppo. La documentazione dev'essere sufficiente a permettere di comprendere un programma, ma non deve essere ingombrante e rappresentare un freno allo sviluppo ulteriore e al redesign di un sistema. I pattern permettono di minimizzare la documentazione nel codice, facilitando in tal modo i cicli di refactoring.

## In questo capitolo...

Questo breve capitolo ci ha permesso di contestualizzare l'idea di pattern e riutilizzo di design. A partire dal prossimo vedremo in modo approfondito i concetti e l'utilizzo dei design pattern nella programmazione a oggetti.