Azzolini Riccardo 2019-10-08

Stili di design

1 Stili

Nei settori più maturi dell'ingegneria, esiste tipicamente un vocabolario comune di stili di design. Anche il software sta andando in questa direzione, ma ancora con meno maturità.

2 Componenti e connettori

Alcuni dei principali **componenti** dei sistemi software sono:

- client,
- server,
- filtri,
- strati,
- basi di dati,
- ecc.

Invece, alcuni dei principali **connettori** sono:

- chiamate di procedura/metodo,
- broadcast di eventi,
- protocolli dei database,
- pipe ("tubi"),
- ecc.

3 Architettura funzionale

In un'architettura funzionale, il sistema si decompone in operazioni astratte:

- per poter interagire, le operazioni conoscono (i nomi di) altre operazioni;
- i connettori sono le *chiamate di procedura* e i *dati condivisi* (quindi, l'interfaccia di ciascuna operazione deve comprendere anche i suoi effetti collaterali).

Quest'architettura può essere implementata mediante linguaggi procedurali (e, in tal caso, le operazioni sono procedure e i dati condivisi sono globali), ma si può applicare anche all'interno delle classi in un linguaggio ad oggetti:

- le operazioni sono i metodi della classe;
- i dati condivisi appartengono alla classe.

4 Architettura a oggetti

In questo tipo di architettura, gli oggetti conoscono i nomi di altri oggetti, con i quali interagiscono mediante chiamate di metodi (invio di messaggi).

5 Sistema a strati

In un sistema a strati, i moduli sono organizzati a strati/livelli. I moduli di un livello:

- usano solo i moduli del livello precedente;
- sono usati solo dai moduli del livello successivo.

Le relazioni tra i moduli formano quindi una gerarchia, e ogni livello può essere interpretato come una *macchina astratta*.

Quest'organizzazione è tipica dei sistemi operativi e dei protocolli di comunicazione.

6 Pipes and filters

Un sistema *pipes and filters* ("tubi e filtri") è costituito da un insieme di filtri, cioè componenti di trasformazione dei dati, collegati tra loro da dei flussi di dati chiamati *pipes*.

In quest'architettura, nessun filtro conosce gli altri: i collegamenti vengono realizzati in un secondo momento. Inoltre, i vari filtri possono essere eseguiti in modo batch sequenziale (applicando un filtro alla volta sull'intero insieme di dati), oppure in parallelo (riducendo quindi i tempi di esecuzione).

Vantaggi:

- è un'architettura *composizionale*, in quanto il comportamento complessivo si ottiene mediante la composizione di comportamenti individuali;
- è *orientata al riuso*, perché sistemi diversi si possono realizzare collegando in modi diversi gli stessi tipi di filtri;
- le *modifiche sono facili*, perché si possono aggiungere/sostituire dei filtri.

Svantaggi:

- quest'architettura non si presta alla *persistenza* dei dati;
- si ha una tendenza all'organizzazione batch, che rende l'architettura poco adatta per sistemi interattivi.

Un esempio di sistema pipes and filters è la shell di Unix, che permette di usare i singoli programmi come filtri, reindirizzando l'output di un qualsiasi programma per inviarlo come input a un altro.

7 Sistemi basati sugli eventi

In un sistema basato sugli eventi, esiste un componente centrale, chiamato **event manager**, al quale gli altri componenti si possono **registrare**. In seguito, ciascun evento che si verifica viene ricevuto dall'event manager e inoltrato a tutti i componenti registrati.

Solitamente, esiste un event manager separato per ciascun tipo di evento. In questo modo, gli eventi di un certo tipo vengono inviati solo ai componenti che devono reagire a essi. Uno stesso evento può essere trattato da più componenti.

Vantaggi:

- l'event manager non conosce esplicitamente i componenti che devono gestire gli eventi;
- si possono facilmente aggiungere/rimuovere componenti.

Svantaggi:

- l'ordine degli eventi può non essere ben definito;
- possono esserci problemi di scalabilità, cioè difficoltà nell'adattare questo tipo di architettura a sistemi di grandi dimensioni.

Questo stile è tipico delle applicazioni con interfaccia grafica, nelle quali gli eventi rappresentano solitamente le azioni di input svolte dall'utente.

8 Sistemi a repository

I componenti comunicano solo mediante lo scambio di dati attraverso un **repository**, che è un "deposito di dati" completamente passivo.

Quest'architettura può essere interpretata come un'estensione all'estremo dell'architettura funzionale, nella quale si eliminano le chiamate tra le operazioni, che quindi interagiscono solo mediante i dati condivisi.

8.1 Varianti

Database: si introduce tra i componenti e il repository un ulteriore componente attivo, chiamato **transaction handler**, che si occupa della gestione dei dati (evitando così che questa debba essere fatta direttamente dai vari componenti).

Blackboard: è un database attivo, che può attivare dei componenti in risposta ai cambiamenti del suo stato.