Tendenze nello sviluppo dei sistemi

Dopo che l'analista ha completato e formalizzato l'analisi dei requisiti funzionali e non funzionali, il progettista, o *architetto*, e il capo progetto possono definire i dettagli dell'implementazione del progetto informatico.
Il progettista osserva i requisiti funzionali e non funzionali e propone un'architettura informatica in grado di eseguire i programmi con i requisiti richiesti.

Un'**architettura informatica** è un insieme di sistemi, sottosistemi e componenti, che supportano i programmi per offrire un sistema complessivo funzionante e mantenibile.

Esistono diverse soluzioni architetturali, che vengono applicate a tipologie comuni di progetti informatici, dette *schemi architetturali* o *pattern*.

Un **pattern architetturale** (*architectural pattern*) è una soluzione architetturale conosciuta e consolidata, che fornisce una soluzione adottabile per problemi tipici e riconoscibili.

Esistono molti pattern architetturali, classificati in diverse categorie, in particolare i pattern utilizzati nell'ambito della gestione dell'informazione aziendale, che comprende tematiche e soluzioni sia organizzative che informatiche.

Il pattern **Layers** è utilizzato per suddividere le responsabilità dei componenti di un software. Deriva dalla tecnica, frequente utilizzata nell'informatica, di stratificare un problema complesso in *livelli*. I livelli sono concettualmente disposti in una pila, o *stack*. Ogni livello ha le seguenti caratteristiche:
- svolge compiti ben definiti;
- utilizza servizi, funzioni e programmi definiti nel livello immediatamente inferiore;
- fornisce servizi, funzioni e programmi al livello immediatamente superiore.

Un esempio di pattern basato su *Layers* è la **three-tier architecture** (architettura a tre strati). In essa sono definiti tre livelli:
- livello dati;
- livello della logica applicativa;
- livello di presentazione.

Un pattern, che pone l'accento sulla possibilità di costruire sistemi distribuiti, si chiama **SOA** (*Service Oriented Architecture*, architettura orientata ai servizi).
L'elemento centrale dell'architettura *SOA* è il **servizio**. Il risultato complessivo che il software deve raggiungere è ottenuto grazie alla collaborazione tra i servizi.

Un *servizio* ha le seguenti caratteristiche:

- è un singolo programma autocontenuto, cioè non dipende dal contesto, né dallo stato, di altri servizi;
- è in grado di funzionare indipendentemente dalla presenza o dal funzionamento di altri servizi;
- è *esposto*, cioè può essere contattato per fornire le proprie funzionalità ad altri programmi, o servizi, che siano in grado di utilizzarle.

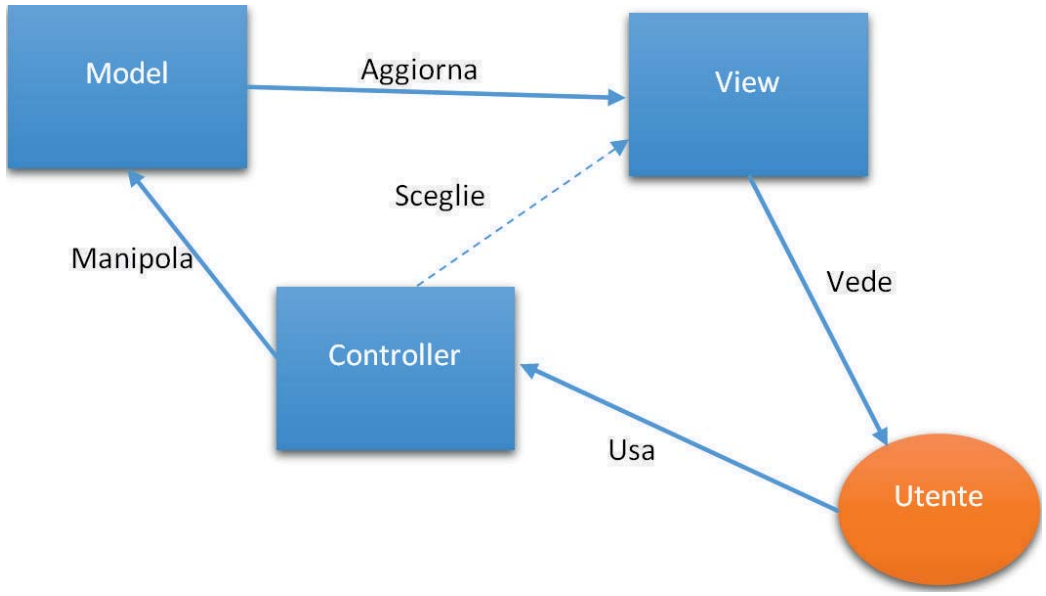
Ognuno dei servizi costituisce un problema informatico di dimensione minore che può essere risolto utilizzando un altro pattern architetturale, come la *three-tier architecture*, vista poco sopra. Il risultato complessivo dell'architettura di un sistema software è quindi una miscela di soluzioni architetturali che risolvono problemi di dimensione più controllabile. Anche in questo caso, nella progettazione del software, viene applicata la tecnica di riduzione della complessità.

Per costruire il livello di presentazione di un'architettura *three-tier* viene utilizzato il pattern architetturale **MVC** (*Model View Controller*, modello visualizzazione controllore).
In questo pattern si pone l'accento sulla separazione dei compiti, necessaria per realizzare un'interfaccia utente.
Nella creazione di interfacce utente si devono produrre elementi grafici interattivi in grado di raccogliere o di mostrare informazioni, sulla base delle richieste dell'utente. Gli elementi grafici solitamente appartengono a un modello di visualizzazione, detto **template**, che abbia la struttura idonea per essere arricchito con i dati variabili, prodotti ed elaborati da un programma.
Si pensi, per esempio, a una pagina Web: nel *template,* realizzato con il linguaggio HTML, vengono inseriti i dati elaborati dal server.

Nel pattern MVC, le responsabilità sono così suddivise:

- Il **Controller** è una tipologia di programma che si occupa di rilevare le azioni dell'utente al fine di individuare l'azione corretta da intraprendere.
- Il **Model** è una tipologia di programma che, istruito dal *Controller*, si occupa di rintracciare i dati che devono essere presentati all'utente.
- La **View** è una tipologia di programma che, invocato dal *Controller*, riesce a creare una visualizzazione dei dati ottenuti dal *Model*, avvalendosi di un *template*.

Schematicamente, il pattern può essere rappresentato come nella figura seguente:



Il ciclo di elaborazione di una richiesta dell'utente, si svolge ordinatamente nel seguente modo:

1. L'utente usa un *Controller* per richiedere la visualizzazione di un'informazione.
2. Il *Controller* manipola il *Model*, demandando ad esso il compito di produrre l'informazione richiesta.
3. Il *Controller* sceglie la *View* che è in grado di produrre una visualizzazione dell'informazione prodotta dal *Model*.
4. Il *Model* aggiorna la *View*, inviando l'informazione.
5. L'utente vede la visualizzazione finale prodotta dalla *View*.