Sistemi Orientati agli Oggetti

Lo sviluppo di sistemi orientati agli oggetti mira a descrivere un software nelle sue diverse fasi (analisi, progettazione, codifica) in termini di ***classi***. Tali sistemi sono documentati focalizzando sulle classi ed usando diagrammi di classi. La classe è spesso pensata come l'entità riusabile più piccola (ovvero la ***granularità*** del riuso è a livello di classe).

Svantaggi della programmazione OO :-(

- Le conseguenze dell'uso delle classi possono non essere chiare.
- La progettazione può essere non corretta ai fini del riuso o di altre proprietà.
- Anche se ben progettate, le classi sono sì entità riusabili ma esse sono troppo piccole per ottenere grandi vantaggi dal loro riuso.

Quello che si spera (e si tenta) di riusare è un insieme di classi, ovvero la *soluzione ad un certo problema*, che molto spesso è costituita da più di una classe.

Entità con granularità più grande, ***architetture di sistemi ad oggetti,*** permettono di ridurre i costi di sviluppo, poiché cercano di riusare insiemi di classi e di produrre sistemi più facili da evolvere.

L'***architettura software*** di un sistema è un artefatto, frutto della attività di progettazione. Una architettura software è descritta dai suoi *componenti* e dalle *relazioni* tra i componenti.

- ***Componenti,*** costituiscono i 'building block' di un sistema (es: moduli, classi, funzioni)
- ***Relazioni,*** denotano una connessione tra componenti: aggregazione, eredità, interazione

Design Pattern

Nella progettazione del software, riferirsi a componenti con granularità più grande della classe, permette di ottenere un migliore riuso ed una maggiore astrazione.

I ***design pattern*** sono strutture software (ovvero micro-architetture) per un piccolo numero di classi che descrivono soluzioni di successo per problemi ricorrenti.

Tali micro-architetture specificano le diverse classi ed oggetti coinvolti e le loro interazioni.

Esistono tanti cataloghi di design pattern, per vari contesti (sistemi centralizzati, concorrenti, distribuiti, real-time, etc.)

Documentazione sul web:

hillside.net

fornisce liste articoli e libri su design pattern, conferenze, link, ...

www.patterndepot.com

Libro su Design Pattern con codice Java (come Gamma, vedi dopo)

www.bruceeckel.com

Libro "Thinking in Patterns" in via di preparazione (come Gamma)

Libri su Design Pattern

Gamma, Helm, Johnson, Vlissides. “Design Patterns – Elements of Reusable Object-Oriented Software”. 1994

E’ il primo ed il più conosciuto libro sui design pattern. Cataloga 23 pattern per sistemi centralizzati.

Buschmann, Meunier, Rohnert, Sommerlad, Stal. “Pattern-Oriented Software Architecture, Volume 1: A System of Patterns. 1996.

Cataloga pattern per sistemi centralizzati e distribuiti.

Lea. “Concurrent Programming in Java: Design Principles and Patterns (2nd Edition)”. 1999.

Cataloga pattern per sistemi concorrenti.

Schmidt, Stal, Rohnert, Buschmann. “Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects. 2000.

Cataloga pattern per sistemi concorrenti e distribuiti.

Vlissides, Coplien, Kert. “Pattern Languages of Program Design”

Altri: vedi Coplien, Vlissides

Design Pattern: Definizione e Usi

I design pattern descrivono un ***problema di design ricorrente*** che si incontra in ***specifici contesti*** di progettazione e presentano una ***soluzione collaudata generica*** ma specializzabile.

In altre parole, sono soluzioni riusabili per una certa classe di problemi ricorrenti.

Usi:

Documentano soluzioni già applicate che si sono rivelate di successo per certi problemi e che si sono evolute nel tempo.

Aiutano i principianti ad agire come se fossero esperti.

Supportano gli esperti nella progettazione di software su grande scala.

Evitano di re-inventare concetti e soluzioni, riducendo il costo del software.

Forniscono un ***vocabolario*** comune e permettono una ***comprensione dei principi del design.***

Analizzano le loro proprietà ***non-funzionali***: ovvero, come una funzione è portata a termine, es. affidabilità, cambiabilità, sicurezza, testabilità, riuso.

Design Pattern - Descrizione

Un design pattern nomina, astrae ed identifica gli aspetti chiave di un problema di progettazione:

- Le classi e le istanze che vi partecipano
- I loro ruoli e come collaborano
- La distribuzione delle responsabilità

Include 5 parti fondamentali:

- **Nome**: permette di identificare il design pattern con una parola e di lavorare con un alto livello di astrazione, indica lo scopo del pattern
- **Intento**: descrive brevemente le funzionalità e lo scopo
- **Problema (Motivazione+Applicabilità)**: descrive il problema a cui il pattern è applicato e le condizioni necessarie per applicarlo
- **Soluzione**: descrive gli elementi (classi) che costituiscono il design pattern, le loro responsabilità e le loro relazioni
- **Conseguenze**: indicano risultati, compromessi, vantaggi e svantaggi nell'uso del design pattern

Nella sezione 'Problema' della descrizione di un design pattern si parla di ***forze*** (come in fisica), in pratica ***obiettivi e vincoli,*** spesso contrastanti, che si incontrano nel contesto di quel design pattern.

Altre parti possono essere presenti nella descrizione:

- **Esempi di utilizzo**: illustrano dove il design pattern è stato usato
- **Codice**: fornisce porzioni di codice che lo implementano