



Progettazione (ingegneria del software)

Questa voce o sezione sull'argomento ingegneria del software non cita le fonti necessarie o quelle presenti sono insufficienti.

In ingegneria del software, la **progettazione** (chiamata talvolta **progetto** o con il falso amico **disegno**, dall'inglese *design*) è una fase del ciclo di vita del software. Sulla base della specifica dei requisiti prodotta dall'analisi, il progetto definisce *come* tali requisiti saranno soddisfatti, entrando nel merito della struttura che dovrà essere data al sistema software che deve essere realizzato.

Descrizione

La progettazione rimane comunque una fase distinta dalla programmazione o codifica, che corrisponde alla traduzione in un particolare linguaggio di programmazione delle decisioni prese in sede di progettazione, cioè dunque la fase implementativa.

Le distinzioni fra le attività fin qui menzionate non sono sempre chiare come vorrebbero le teorie classiche dell'ingegneria del software. La progettazione, in particolare, può descrivere il funzionamento interno di un sistema a diversi livelli di dettaglio, ciascuno dei quali si colloca in una posizione intermedia fra analisi e codifica.

Normalmente si intende con *progettazione dell'architettura* (o *progettazione architettuale*) la progettazione "ad altissimo livello", in cui si definisce solo la struttura complessiva del sistema in termini dei principali moduli di cui esso è composto e delle relazioni macroscopiche fra di essi. A questo livello di progettazione appartengono formule come client-server o three-tier, o più in generale decisioni sull'uso di particolari architetture hardware, sistemi operativi, DBMS, protocolli di rete, e così via.

Un livello intermedio di dettaglio definisce ancora la scomposizione del sistema in moduli, ma questa volta con riferimento più o meno esplicito alle modalità di scomposizione offerte dal particolare linguaggio di programmazione con cui avverrà lo sviluppo; per esempio, in una progettazione condotta con tecnologie a oggetti, il progetto potrebbe descrivere il sistema in termini delle principali classi e delle loro interrelazioni.

Il *progetto di dettaglio*, infine, rappresenta una descrizione del sistema *molto vicina* alla codifica, ovvero che la vincola in maniera sostanziale (per esempio, descrivendo non solo le classi in astratto ma anche i loro attributi e metodi, con relativi tipi e "firma").

A causa della natura "impalpabile" del software, e a seconda degli strumenti che si utilizzano nel processo, il confine fra progettazione e codifica può essere anche praticamente impossibile da identificare. Per esempio, alcuni strumenti CASE sono in grado di generare codice a partire da diagrammi UML che descrivano graficamente la struttura di un sistema software.

Storia

L'esigenza di comprendere le difficoltà del realizzare software nasce alla fine degli anni sessanta con la consapevolezza che una base di buona programmazione non voleva dire avere a disposizione dei buoni sistemi.

Lo sviluppo si ebbe negli anni cinquanta e anni sessanta quando il problema era l'attività di programmazione tramite altri studi riferiti a algoritmi, strutture Dati, programmazione Strutturata, linguaggi di Programmazione. La programmazione veniva vista come "mettere assieme" una sequenza d'istruzioni. Negli anni '50 fece la sua comparsa Assembly, linguaggio a basso livello, e alla fine Anni '50 comparvero i primi linguaggi di "alto" livello (Cobol, Fortran). In questo periodo la programmazione si basava sull'abilità dei programmatori, sullo sviluppo di applicazioni di dimensioni piccole o medie (calcolo centralizzato) e sulla metodologia "prova e correggi" ("code and fix"). Ma con la crescita della complessità, la comparsa delle reti e applicazioni client-server si avverte la necessità di una metodologia per guidare il processo di sviluppo.

A questo punto si assistette all'introduzione dei nuovi linguaggi che rese più semplice la programmazione e la diffusione dei PC permise di far nascere la figura professionale di programmatore, non come utente finale dell'applicazione, ma come realizzatore dell'applicazione.

Verso la fine degli anni '60 nacquero i primi progetti software a fini commerciali, ma le buone tecniche di programmazione acquisite fino a quei tempi non bastavano per l'evoluzione di un software potente. I progetti erano più costosi del previsto e sempre in ritardo con i tempi di sviluppo. In una conferenza del 1968 si iniziò a parlare della Software crisis, dove si sottolinea che il problema non era più "scrivere sequenze d'istruzioni" ma era comunicare con le persone coinvolte nello sviluppo. Le cause della crisi sono da rapportare al cambio dei requisiti durante lo sviluppo, ai cambiamenti che coinvolgono tutto il sistema, al turn over del personale.

Dalla conferenza emerse un nuovo scenario, che prevedeva un approccio strutturato allo sviluppo, con la conseguente introduzione di Management, Organizzazione, Teorie e Tecniche, Strumenti e Metodologie appropriate. Si iniziò a parlare di ciclo di vita del software, inteso come descrizione delle attività che portano alla realizzazione e alla gestione di un sistema.

Nel 1984 James Martin, esperto di software engineering pubblica "An Information Systems MANIFESTO" (sic), un vero e proprio provocatorio manifesto destinato a scuotere la letargia e scarsa propensione del management di allora a vedere rischi e opportunità nella tecnologia informatica. Il leitmotiv dell'opera, si può riassumere ne *"l'informatica non è un dettaglio tecnico demandato agli specialisti EDP, ma una risorsa strategica che merita l'attenzione continua dell'alta direzione"*. Nel contempo James Martin pone le basi di nuove tecniche di sviluppo quali il CASE e il RAD.

Nel 1994 la "Standish Group International", nell'ambito di un progetto di ricerca, ha cercato di dare delle risposte nell'ambito dei fallimenti dei progetti software.

I progetti sono stati classificati in tre categorie distinte:

1. Progetti di successo: completati nei tempi stabiliti e senza sforare il budget stabilito e con tutte le caratteristiche e funzioni stabilite nelle specifiche iniziali.
2. Progetti in ritardo: il progetto è completato ed operativo ma ha richiesto più tempo e

denaro rispetto a quanto stimato, ed offre un minor numero di funzioni

3. Progetti cancellati: il progetto è stato cancellato durante il ciclo di sviluppo, prima di essere terminato.


Sebbene differenti progetti falliscano in modi differenti, la causa principale per la maggior parte dei fallimenti nello sviluppo del software risulta essere l'inadeguatezza della metodologia.

Altre cause risultano essere:

- una eccessiva burocratizzazione dello sviluppo
- difficoltà nel tenere conto dei cambiamenti nei requisiti
- software difficile da mantenere o da fare evolvere
- scoperta tardiva dei gravi errori di progetto
- test insufficienti
- scarsa qualità del software
- prestazioni inaccettabili del software.

Per questo motivo vennero introdotte varie tipologie di sviluppo del software che servirono a migliorare i limiti riscontrati.

Altri progetti

-  Wikimedia Commons (<https://commons.wikimedia.org/wiki/?uselang=it>) contiene immagini o altri file su **progettazione del software** (https://commons.wikimedia.org/wiki/Category:Software_design?uselang=it)



Portale Design



Portale Informatica



Portale Ingegneria

Estratto da "[https://it.wikipedia.org/w/index.php?title=Progettazione_\(ingegneria_del_software\)&oldid=143018100](https://it.wikipedia.org/w/index.php?title=Progettazione_(ingegneria_del_software)&oldid=143018100)"