Glossario

Di seguito sono riportati alcuni termini utilizzati nel documento con il significato più comunemente dato dall'ingegneria del software.

A

Accettazione: Ultima fase del ciclo di sviluppo, dopo il test e collaudo, in cui il prodotto software è validato dal committente. E' detto anche collaudo di accettazione.

Accuratezza: Caratteristica di qualità del software definita dalle norme ISO9126 come il livello di aderenza ai requisiti, prescrizioni, ecc.

Adattabilità: Caratteristica di qualità del software definita dalle norme ISO9126 come la facilità con la quale è possibile modificare il software per adattarlo a nuove esigente.

Affidabilità: Caratteristica di qualità del software definita dalle norme ISO9126 come la capacità di funzionare correttamente nel tempo.

Algoritmo: Complesso ben definito di regole (generalmente matematiche) che permettono di risolvere un problema attraverso una serie di passi successivi.

Analisi: Primissima fase del ciclo di sviluppo del software in cui sono analizzati il problema da risolvere e le esigenze della committenza da indirizzare.

Analogico: Informazione che esprime quantità fisiche variabili in forma continua (per esempio tensioni elettriche, frequenze sonore ecc.). Si contrappone al dato "digitale", cioè discontinuo, discreto, esprimibile con una cifra (*digit*).

Anomalia: vedi difetto.

Apprendibilità: Caratteristica di qualità del software definita dalle norme ISO9126 come la facilità con cui l'utente può apprendere le funzioni rese disponibili dal software in oggetto.

Appropriatezza: Caratteristica di qualità del software definita dalle norme ISO9126 come il grado di attinenza del prodotto ai requisiti specificati.

Architettura: L'organizzazione fondamentale di un sistema considerato nella sua interezza. L'architettura di un sistema è composta da un numero di aspetti, tra cui elementi statici, elementi dinamici, l'interazione tra tali elementi e lo stile architetturale generale, che dà la sua impronta all'organizzazione del sistema stesso. Il concetto di architettura fa anche riferimento ad aspetti quali prestazioni, scalabilità, riuso, oltre agli immancabili vincoli economici e tecnologici.

Assicurazione qualità (AQ): vedi qualità.

Attività: Qualsiasi passo fatto o funzione eseguita, sia mentale che fisica, per il conseguimento di qualche obiettivo. Le attività includono tutto il lavoro dello staff tecnico e manageriale fatto per realizzare i task del progetto e dell'organizzazione.

Attore: Un ruolo interpretato da un utente del sistema o da un'entità esterna, come un altro sistema o un data-base, che risiede fisicamente al di fuori del sistema considerato.

Attributo: In UML una proprietà che descrive un *oggetto* o una *classe*.

Automazione: Processo attraverso il quale viene affidato a macchine automatiche (nel nostro caso i computer) l'intero svolgimento di una procedura di lavoro (aprire e chiudere un conto corrente, stampare una fattura, aggiornare i dati del magazzino ecc.).

Azione: In UML un'istruzione eseguibile che provoca il cambiamento del valore di uno o più *attributi* di un *oggetto*, oppure la restituzione di uno o più valori all'oggetto che ha inviato il *messaggio*.

B

Banca dati: In generale, insieme di informazioni strutturate organicamente e caratterizzate da un sistema di classificazione e ricerca. Secondo la L. 675/96, per banca dati si intende qualsiasi complesso di dati personali, ripartito in una o più unità dislocate in uno o più siti, organizzato secondo una pluralità di criteri determinati tali da facilitarne il ritrovamento.

Black-box: Tipo di test effettuato sul software in cui i moduli (programmi) sono visti esteriormente considerando solo le funzionalità esterne. In contrapposizione a *White-box*.

Bottom-up: vedi Top-down.

Bug: vedi difetto.

C

Cammino (*Path*): Sequenza di istruzioni di un modulo o programma che descrive uno dei possibili percorsi della sua logica interna. E' riferita al grafo di flusso. E' utilizzato per calcolare la complessità ciclomatica di un modulo o il percorso valicato tramite l'esecuzione di un test.

Capability Maturity Model (CMM): Una descrizione degli scenari, attraverso i quali le organizzazioni si evolvono, di come le aziende definiscono, misurano, controllano ed incrementano i loro processi software. Questo modello fornisce una guida per selezionare strategie di miglioramento dei processi agevolando la determinazione della capacità del processo corrente e l'identificazione degli elementi più critici per la qualità del software e per il miglioramento dei processi.

Capability Maturity Model Integration (CMMI): nuova versione del modello CMM estesa a tutti i settori d'industria, oltre a quello del software da cui è partito originariamente (vedi CMM).

Capo progetto: Il ruolo che ha l'intera responsabilità di un progetto. La persona che dirige, controlla, amministra e regola un progetto per la realizzazione di un sistema software o hardware/software.

Caso di test: Unità di test che descrive le operazioni da eseguire, i dati richiesti dalle prove ed i risultati attesi.

Caso d'uso (*Use Case*): Una sequenza di azioni eseguite da un *attore* all'interno del sistema per raggiungere un determinato scopo.

Ciclo di vita: Processo che descrive lo sviluppo (la produzione) del software. E' descritto generalmente in termini di fasi e sottofasi, input e output, ruoli e responsabilità, validazioni e controlli, ecc. Dai primi cicli di vita a cascata (*waterfall*), si è passati a cicli iterative, incrementali, rad, ecc.

Classe: In UML una collezione di *oggetti* che hanno le stesse caratteristiche.

Cliente/Committente: La persona o l'organizzazione che ha commissionato il lavoro di sviluppo software e che è responsabile di accettare il prodotto realizzato ed autorizzare il pagamento del corrispettivo pattuito all'organizzatore di sviluppo.

Codice: E' il risultato dell'attività di programmazione e costituisce la materia prima del software.

Codifica: Attività volta alla produzione del software. Detta anche programmazione.

Collaudo: Test formale condotto per determinare se un sistema soddisfa i criteri di accettazione e permettere quindi al cliente di accettare il sistema.

Complessità: Caratteristica di qualità del software che indica il livello di strutturazione del codice sorgente, del disegno del sistema. Le metriche relative sono state sviluppate da McCabe ed altri.

Componente: Una parte fisica e sostituibile di un sistema che si conforma alle specifiche stabilite e realizza un insieme di funzioni attraverso *interfacce*.

Controllo: vedi *Verifica*.

Copertura: Misura il livello di efficacia del test ed indica quanto il test eseguito copra il software validato.

Correttezza: Caratteristica di qualità del software che indica il livello di aderenza alle specifiche funzionali e tecniche definite.

Criteri di accettazione: I criteri che un sistema o componente deve soddisfare per essere accettato da un utente, un cliente, o un'altra entità autorizzata.

D

Database: vedi *Banca dati*.

Debugging: Attività di ricerca ed eliminazione degli errori nel software.

Design: vedi *Progettazione*.

Design pattern: Un *pattern* che entra in gioco durante la fase di progettazione (design) del sistema software.

Diagramma: Rappresentazione schematica della sequenza delle operazioni di una elaborazione.

Diagramma di flusso (*Flow chart*): Schema basato su simboli grafici e linee di interconnessione per descrivere il flusso dei dati e delle informazioni in una procedura.

Diagramma di Gantt: Una forma di rappresentazione grafica di uno o più compiti coordinati, definiti all'interno di un progetto e posti in relazione temporale. Consente la pianificazione dei progetti.

Diagramma di stato: Un diagramma che rappresenta la *macchina a stati* di un *oggetto*.

Dialogo: (1) In generale, una serie di interazioni ordinate tra due entità, attraverso un sistema di comunicazione. (2) Nel software, Una serie di procedure di domande e risposte tra un utente e il proprio sistema di elaborazione.

Difetto: Un vizio in un sistema o componente del sistema che determina il fallimento del sistema / componente nell'eseguire la funzionalità richiesta.

Driver*:* In un ambiente di test o nella fase di integrazione di un sistema software, componente usato per controllare il comportamento di uno o più moduli. Un driver invoca le funzionalità esportate dai moduli letteralmente "pilotando" la loro esecuzione".

E

Efficienza: Caratteristica di qualità del software definita dalle norme ISO9126 come la capacità di svolgere le funzionalità richieste con il minimo consumo di risorse (tempo, memoria, spazio disco, utilizzo linee, ecc.). Nota anche come *performance*.

Ergonomia: Studio delle condizioni di lavoro umano a contatto con macchine per migliorarlo. Interviene quando un particolare modo di lavorare si diffonde tanto da diventare un fenomeno socialmente rilevante. Nel settore dell'informatica la questione riguarda soprattutto l'introduzione dei computer nella vita quotidiana: banche, uffici pubblici, fabbriche, magazzini, supermercati, intrattenimento, ecc.

Errore: E' la causa di un difetto. E' commesso a livello concettuale (ad esempio, l'errata comprensione dei requisiti del cliente) oppure operativo (ad esempio, errata produzione del codice sorgente).

Evento: Qualcosa che accade e che ha rilevanza per un *oggetto* (per esempio, ne modifica lo stato).

F

Fase: Il lasso di tempo tra due punti temporali, uno di partenza ed uno di arrivo. Nel ciclo di sviluppo si identificano più fasi: Studio di fattibilità, Analisi e disegno, Realizzazione, Collaudo e Rilascio.

Function Point (FP): In italiano Punto Funzione, misura le dimensioni di un software (modulo, componente, prodotto, ecc.) in termini di funzionalità offerte.

Flusso: In UML, il *Flusso principale* rappresenta il cammino "ideale" all'interno di un *caso d'uso* in cui l'*attore* e il *sistema* seguono il percorso principale dall'inizio alla fine e le circostanze

incontrate sono sempre quelle "normali". Il *Flusso eccezionale* rappresenta il cammino attraverso un *caso d'uso* che descrive una situazione d'errore o semplicemente un cammino che l'*attore* e il *sistema* seguono meno frequentemente, per esempio in condizioni particolari.

Funzionalità: Caratteristica di qualità del software definita dalle norme ISO9126 come la sua capacità di soddisfare i requisiti funzionali.

G

Grafo di flusso: Rappresentazione schematica di un programma: ad ogni nodo è associato un comando, mentre gli archi, orientati, rappresentano il legame di sequenzialità fra i comandi.

I

Installabilità: Caratteristica di qualità che indica la facilità con cui un prodotto software può essere installato nel suo definitivo ambiente operativo.

Integrazione: Attività volta a verificare che i vari moduli sviluppati interagiscano correttamente tra di loro dal punto di vista funzionale e non.

Interfaccia: Tutto ciò che realizza la interconnessione di due componenti di un sistema finalizzata alla loro interoperazione. In UML una collezione di *operazioni* che rappresentano i servizi offerti da una *classe* o da un *componente*.

Interfaccia utente: Insieme di strumenti (componenti, maschere, bottoni, ecc.) che il sistema mette a disposizione dell'utente per le sue interazioni con il sistema stesso.

ISO: *International Organization for Standardization*: organizzazione internazionale per la standardizzazione nei vari settori produttivi.

Ispezione: tecnica di verifica della completezza e correttezza di un prodotto intermedio del software (codice, documento, ecc.).

L

Line of code (*Loc*): In italiano linea di codice, misura le dimensioni del software in termini di numero di istruzioni scritte nel linguaggio adoperato. L'uso dei generatori di codice, e la necessità di una misura più orientata al valore del software, hanno portato a sostituire questa misurazione con la tecnica dei punti funzione (*function points*).

M

Malfunzionamento: comportamento di un prodotto software o di una sua componente in maniera non conforme alle specifiche. Il prodotto, cioè, non si comporta come ci si aspetta che faccia.

Manutenibilità: Caratteristica di qualità del software definita dalle norme ISO9126 come la facilità a modificare il software prodotto ed in esercizio per motivi diversi (correzioni di errori, aggiunta di nuove funzioni, ecc.). Distingue un buon codice da un ottimo codice.

Manutenzione: Fase finale del ciclo di vita del software in cui il prodotto in esercizio è modificato per correggerne eventuali errori rilevati, aggiungere nuove funzionalità, migliorarne le caratteristiche di qualità, ecc.

Metodo: In UML una funzione che utilizza o cambia il valore di uno o più *attributi* di un *oggetto*.

Metrica: Criterio di misurazione di una caratteristica del software secondo una determinata scala ed unità di misura. Esempi: dimensioni del software misurato in termini di numero di linee di codice o di *function point*, efficienza misurata in termini di tempo di risposta, ecc.

Modello: Una semplificazione della realtà che aiuta le persone a comprendere la complessità intrinseca del software.

Modello di analisi: In UML un *modello* che aiuta gli sviluppatori a raffinare e strutturare i requisiti funzionali acquisiti (direttamente o indirettamente) e venuti alla luce durante l'analisi dei casi d'uso.

Modificabilità: Caratteristica di qualità del software definita dalle norme ISO9126 come la facilità nel poter eseguire modifiche all'architettura, al disegno, al codice, ecc.

Modulo: Una parte del sistema software, componente o sottocomponente, riferito all' architettura o al codice.

N

Nodo: In UML una parte di hardware che rappresenta una risorsa computazionale, quindi in grado di memorizzare ed elaborare dati e informazioni.

O

Oggetto: In UML una cosa o un concetto esistente nel mondo reale.

Operabilità: Caratteristica di qualità del software definita dalle norme ISO9126 come la facilità con cui un sistema software è utilizzato dai suoi utenti nell'operatività quotidiana.

P

Package: Un raggruppamento coerente di componenti (elementi) che

completa un insieme di funzioni. In UML, un insieme concettuale di elementi facenti parte di un *modello*.

Pattern: La soluzione di un problema applicabile in una varietà di contesti.

Polimorfismo: Il principio dell'object-orientation in base al quale un *oggetto* di una *classe figlia* può sempre ridefinire ognuna delle *operazioni* che *eredita* dalla *superclasse* (o superclassi).

Portabilità: Caratteristica di qualità, definita da ISO9126 e che indica la capacità di un prodotto software di essere trasportato da un ambiente ad un altro.

Performance: vedi efficienza.

Processo: Una sequenza di passi formali eseguiti per uno dato scopo, per esempio, il processo del sviluppo software.

Progettazione: In inglese *design*, è la fase del ciclo di sviluppo del software con l'obiettivo di studiare la soluzione da offrire come soluzione ai problemi analizzati nella precedente fase di analisi.

Progetto: In inglese *project*, è la struttura organizzativa che ha la responsabilità di sviluppare un sistema software secondo il processo definito per il ciclo di sviluppo.

Programma: E' l'insieme delle istruzioni, scritte nel linguaggio formale adoperato, per trasformare una funzionalità, un algoritmo, ecc. in modo comprensibile al calcolatore. E' anche utilizzato per rappresentare un insieme di progetti informatici.

Q

Qualità: Riferito al software, è l'insieme di tutte le sue caratteristiche che gli consentono di soddisfare le diverse richieste e necessità direttamente espresse o implicitamente necessarie. Le caratteristiche di qualità del software sono definite dalle norme ISO9126: funzionalità, affidabilità, usabilità, efficienza, manutenibilità, portabilità. Il livello di qualità atteso per un software specifico è definito nel documento di progetto Piano della qualità.

Assicurazione della qualità (in inglese Quality Assurance) è l'insieme delle attività svolte per assicurare che il livello di qualità previsto sia raggiungibile e quindi raggiunto.

R

Recuperabilità: Caratteristica di qualità del software definito da ISO9126 come la capacità di ripristinare la normale funzionalità

operativa dopo il verificarsi di un malfunzionamento.
Regressione: Attività di test volta a verificare che tutte le caratteristiche di qualità del software, o di una sua parte, siano mantenute a seguito di modifiche fatte ad uno o più componenti, programmi, moduli.

Requisito: Definizione dettagliata che descrive una funzionalità (requisito funzionale) o caratteristica di qualità (requisito di qualità o non-funzionale) del sistema software da sviluppare. Può essere esplicito o implicito.

Rimpiazzabilità: Caratteristica di qualità del software definita da ISO9126 come la facilità a sostituire un software analogo.

Robustezza: Rappresenta la tolleranza ai guasti di un sistema software, ovvero la capacità di funzionamento continuativo senza evidenziare malfunzionamenti.

S

Scaffolding: In un ambiente di test o nella fase d'integrazione di un sistema software, componente usato per simulare il comportamento di uno o più moduli. E' dotato della stessa interfaccia (in termini di input e output) dei moduli che deve simulare.

Sicurezza: Caratteristica di qualità del software definita da ISO9126 come la capacità di proteggere da eventuali intrusioni o manomissioni, intenzionali o involontarie.

Sistema: un insieme di componenti organizzate per eseguire una specifica funzione od un insieme di funzioni. Secondo il linguaggio UML, un sistema è un *package* che contiene tutti i *modelli* prodotti dal gruppo durante lo sviluppo.
Software: Il complesso dei programmi, delle istruzioni e dei documenti necessari per risolvere attraverso certe unità fisiche (hardware) i problemi di elaborazione dei dati. Si distingue tra software di base, destinato a gestire il funzionamento delle diverse unità di un sistema in modo automatico, e software applicativo, destinato a gestire le specifiche operazioni che danno vita a un particolare impiego concreto dell'elaboratore.

Stabilità: Caratteristica di qualità del software definita da ISO9126 come il livello di maturità raggiunto tale da non richiedere ulteriori modifiche di correzione e/o miglioramento.
Standard: Requisiti obbligatori impiegati e vigenti per descrivere un disciplinato approccio uniforme allo sviluppo del software.

Studio di fattibilità: Attività della fase di analisi in cui il sistema software è valutato in termini di costi e benefici prima di impegnare risorse per l'eventuale realizzazione.

T

Task: Una sequenza di istruzioni considerate come un'unità base del lavoro. Una ben definita unità di lavoro nel processo di sviluppo del software che consente la verifica dello stato di avanzamento del progetto.

Template: Letteralmente "schema", rappresenta un modello predefinito per la realizzazione di oggetti (documento, programma, ecc.).

Test (o Testing): Tecnica principe per la validazione del software e la valutazione del livello di qualità raggiunto. "Il processo di valutazione di un sistema attraverso strumenti manuali o automatici col fine di determinare se il sistema soddisfa i requisiti specificati oppure se il suo comportamento attuale differisce da quello atteso" (IEEE Std 829-1983, Standard for Software Test Documentation).

Tolleranza ai guasti: Caratteristica di qualità del software definita da ISO9126 come la capacità a proseguire nel suo funzionamento corretto anche a seguito di malfunzionamenti.

Top-down: Tecnica di test che prevede la validazione del software partendo dalle componenti più alte fino a raggiungere i dettagli delle sotto-componenti e moduli. In contrapposizione con la tecnica inversa (**Bottom-up**).

U

Usabilità: Caratteristica di qualità di un prodotto software relativa alla sua possibilità di essere impiegato produttivamente da una particolare utenza. ISO 9126 identifica nell'usabilità una caratteristica di qualità primaria, composta da comprensibilità, apprendibilità e operabilità.

Use case: vedi *Caso d'uso*.

V

Validazione: Attività atta a verificare che il software sviluppato soddisfi i requisiti attesi, espliciti ed impliciti. La tecnica di validazione più usata ed efficace è il Test.

Verifica: Attività atta a verificare che il software sviluppato (o le sue componenti come, ad esempio, la documentazione) sia corretto, cioè privo di errori. Le tecniche più efficaci

sono sempre i test e le revisioni tecniche (dette *ispezioni*).

W

White-box: Tipo di test effettuato sul software in cui i moduli (programmi) sono visti internamente considerando la loro struttura. In contrapposizione a *Black-box*.

Workflow: Un insieme di attività eseguite da uno o diversi membri del gruppo di sviluppo all'interno del processo.