Qualità del Software

In generale, con il termine **Qualità** si intende l'insieme di attributi che caratterizzano un bene o un servizio, e che ne determinano la capacità di soddisfare i bisogni impliciti e espliciti dell'utenza.

Ma quando questa definizione generale viene applicata ad uno specifico bene/servizio software, ci si accorge che essa lascia adito ad un numero molto alto di possibili interpretazioni.

Valutare la qualità del software risulta difficile, con risultati che appaiono per lo più discutibili e spesso scarsamente verificabili.
Il problema di fondo è che il concetto di qualità assume connotazioni diverse in funzione dell'oggetto esaminato, del contesto di analisi e del punto di vista dell'osservatore.

Qualità del Software

Parlando di qualità del software, distinguiamo fra:

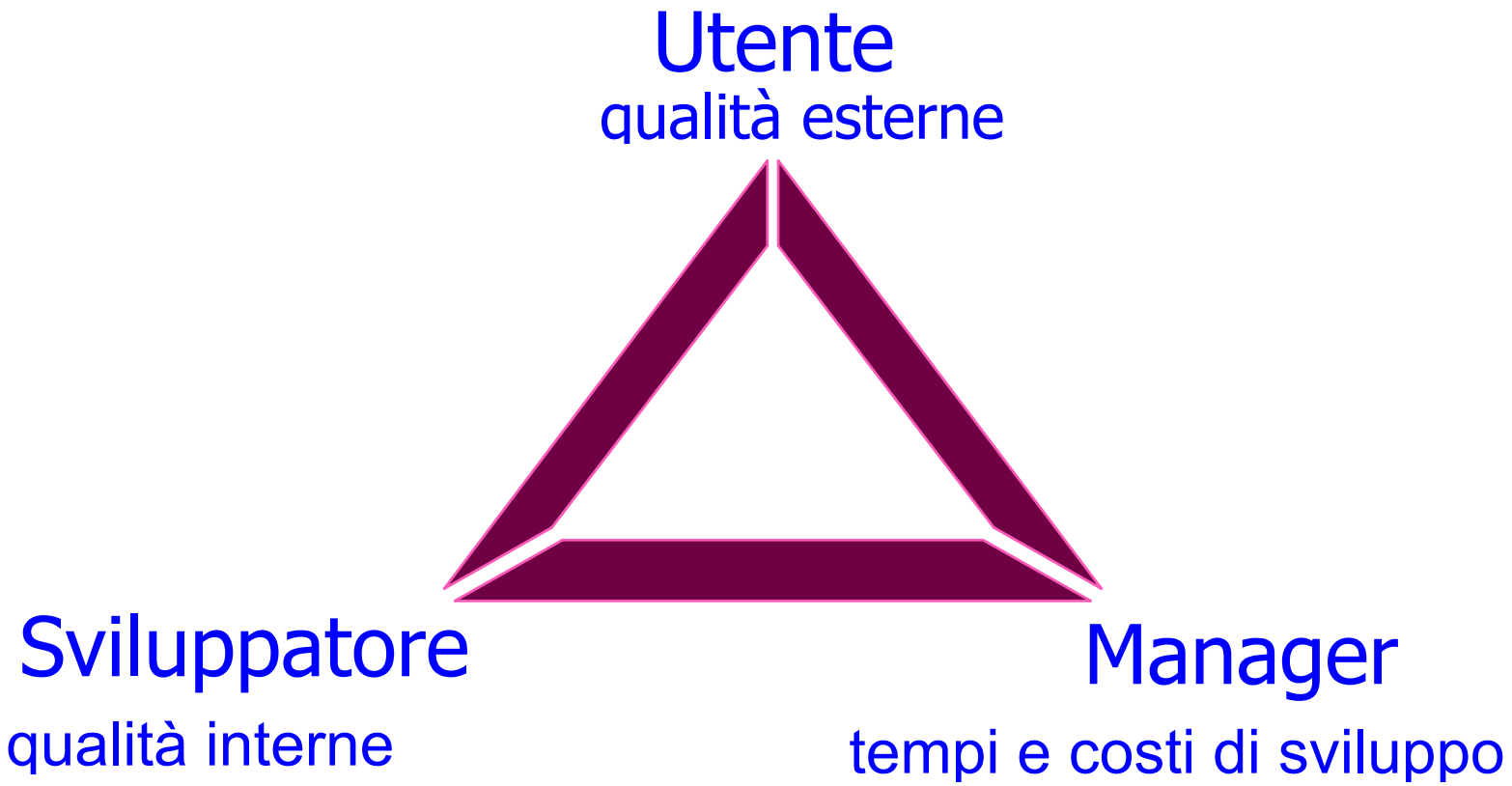
- Qualità del **processo**

- Qualità del **prodotto**
  - Qualità **esterne** → percepite dagli utenti (tempi di risposta, facilità d'uso, manuale utente)
  - Qualità **interne** → percepite dagli sviluppatori (architettura di sistema, documentazione interna)

L'obiettivo è produrre un prodotto di qualità. Ma le qualità del prodotto sono influenzate dalla qualità del processo.
Analogamente, ciò che realmente interessa sono le qualità esterne, ma queste sono diretta conseguenza delle qualità interne.

Qualità del Software

La "qualità" dipende dall'ottica!



Qualità del Software

Fra le cause che concorrono a rendere difficoltoso il processo di valutazione della qualità del software, possiamo elencare:

- Il mondo delle tecnologie dell'informazione è caratterizzato da una velocità e intensità di cambiamento che non ha precedenti nella storia dello sviluppo industriale.

- I prodotti IT, e in particolare il software, sono contraddistinti da una complessità estremamente elevata. Nel caso di sistemi così complessi, risulta difficile enucleare un insieme ragionevolmente limitato di parametri e caratteristiche che possano rappresentare in maniera significativa il comportamento e le prestazioni del sistema osservato.

- Le tecnologie dell'informazione costituiscono il "sistema nervoso" di moltissimi prodotti e servizi. Spesso, le tecnologie dell'informazioni sono immerse all'interno di sistemi più complessi, siano essi strutture organizzative come un call center o apparati elettromeccanici come un aeroplano. Ciò che si può misurare, spesso, sono le caratteristiche prestazionali del sistema

complessivo, mentre risulta critico identificare il contributo specifico delle tecnologie dell'informazione.

- La valutazione di beni e servizi spesso è condizionata dalla cultura, formazione, ruolo e attitudine dell'utente. Per esempio, ciò che è "facile da usare" per un utente, può non esserlo per altri.

- Ancora troppo spesso, la valutazione e lo studio dei prodotti IT, e in particolare del software, si basa su competenze insufficienti e su una sostanziale superficialità di chi gestisce tali processi di valutazione.

Qualità del Software – Norme internazionali

Norme ISO

ISO (International Organisation for Standardization) è una organizzazione internazionale che si occupa di definire e diffondere standard in tutti i processi produttivi.

Sicuramente le norme più conosciute e più applicate nel settore della qualità sono le norme ISO 9000, interpretate in modo più specifico per le aziende di software dalla guida ISO 9000-3, ed ora rinnovate con l'uscita della versione 2000 (Vision 2000).
Sono norme per la certificazione dei sistemi di qualità e si applicano in generale ad ogni organizzazione di progettazione, produzione e distribuzione (in precedenza con le tre varianti ISO 9001, 9002 e 9003, ora con la stessa versione base).

Le norme ISO 9126 definiscono il modello dei requisiti qualitativi del prodotto software. In queste norme sono definite le voci e sottovoci con le quali vengono classificati i requisiti di qualità del software.

Qualità del Software – Norme internazionali

ISO 12207 è un esempio di definizione rigorosa dei processi di sviluppo del software, dei quali indica uno schema di riferimento riconosciuto e condiviso dagli operatori del settore.

ISO 15504 copre il passo successivo: si basa infatti sul modello di processi della ISO 12207 per regolamentare in modo molto dettagliato le attività di valutazione dei diversi processi e di attribuzione di un "voto" al livello di maturità dimostrato.

Ricordiamo anche la norma ISO 8402, che costituisce il glossario dei termini della qualità, e la norma ISO 10011 che definisce le attività di verifica ispettiva.

Norme SEI-CMM e Mil-Std-498

SEI-CMM è uno dei primi modelli di valutazione dei processi software, sviluppato nel 1986 in ambito IBM e poi adottato nel 1993 da SEI (Software Engineering Institute), che lo ha diffuso a livello mondiale.

Qualità del Software – Norme internazionali

La metodologia CMM si basa essenzialmente sul processo di sviluppo descritto nello standard Mil-Std-498, sviluppato negli Stati Uniti in ambito militare.

Qualità del Software – Norme internazionali

La **Norma ISO 9126** ("Information Tecnology - Software product evaluation - Quality characteristics and guidelines for their use"), pubblicata nella sua prima versione nel 1991, definisce il modello dei requisiti qualitativi del Prodotto Software.

La presenza dei requisiti richiesti può essere verificata tramite riesami e test. Ogni caratteristica deve essere valutata tramite questionari e tabelle, da compilare assieme al cliente.

Secondo il modello proposto dalla norma, i requisiti sono raggruppabili in 6 "caratteristiche" e in 21 "sottocaratteristiche", distinte fra caratteristiche esterne (orientate all'utente) e caratteristiche interne (orientate allo sviluppo e manutenzione).

Qualità del Software – Norme internazionali

Requisiti individuati da norme ISO 9126:

1. Functionality:
   - Suitability: coerenza delle funzioni offerte dal prodotto rispetto alle esigenze dell'utenza.
   - Accuracy: correttezza e precisione dei risultati prodotti.
   - Interoperability: capacità del prodotto di interagire con altri sistemi software.
   - Compliance: conformità a standard, convenzioni o regolamenti rilevanti per il settore applicativo del prodotto.
   - Security: capacità del prodotto di prevenire accessi non autorizzati alle informazioni e funzioni gestite dal prodotto.

2. Reliability:
   - Maturity: distribuzione dei malfunzionamenti in funzione degli errori presenti nel software.
   - Fault tolerance: capacità di mantenere livelli predeterminati di prestazioni anche in presenza di malfunzionamenti o utilizzi scorretti del prodotto.

Qualità del Software – Norme internazionali

- Recoverability: capacità del prodotto di ripristinare il livello appropriato di prestazioni del prodotto e di recuperare tutte le informazioni rilevanti a valle di un malfunzionamento del prodotto.

3. Usability:
   - Understandability: facilità di comprensione per l'utenza dei concetti del prodotto.
   - Learnability: facilità di apprendimento del prodotto.
   - Operability: facilità di utilizzo del prodotto da parte dell'utenza.

4. Efficiency:
   - Time behaviour: tempi di risposta.
   - Resource behaviour: utilizzo delle risorse di sistema.

Qualità del Software – Norme internazionali

5. Maintainability:
   - Analyzeability: facilità con la quale è possibile localizzare un errore nel codice del prodotto.
   - Changeability: facilità con la quale è possibile cambiare il codice del programma.
   - Stability: livello del rischio di effetti indesiderati attraverso la modifica del codice.
   - Testability: sforzo necessario per verificare il programma.

6. Portability:
   - Adaptability: capacità del prodotto di essere adattato a diversi ambienti operativi.
   - Installability: facilità di installazione del prodotto.
   - Conformance: conformità a standard relativi alla portabilità.
   - Replaceability: facilità con cui si può utilizzare il prodotto al posto di un altro componente.

Qualità del Software – Norme internazionali

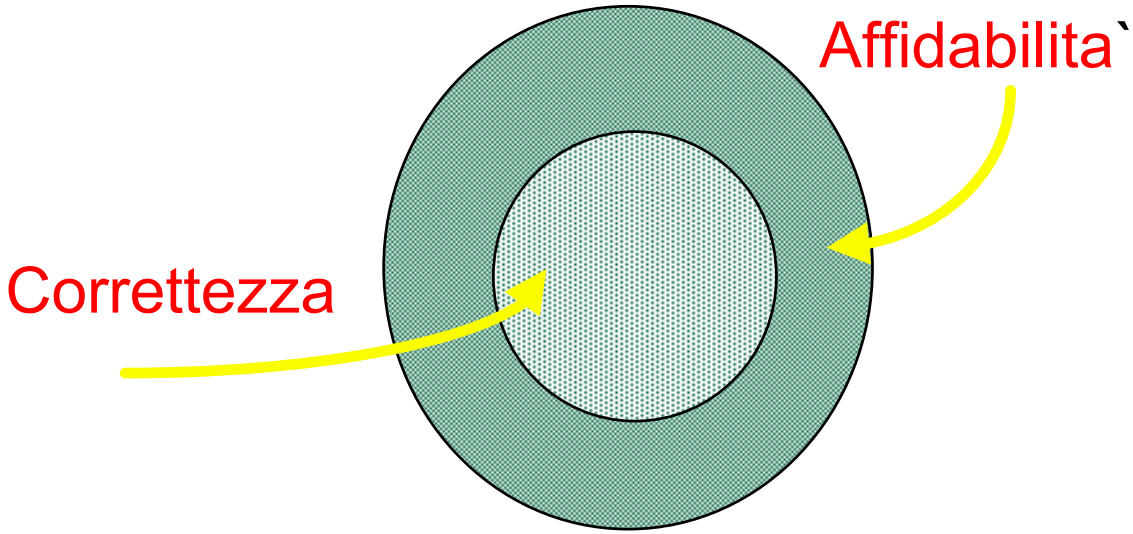
Possiamo quindi evidenziare, fra i requisiti ISO 9126:

Correttezza
Un software è corretto quando si comporta secondo le specifiche funzionali. Si tratta probabilmente della caratteristica più importante, specie per sistemi critici. La correttezza può essere verificata con metodi analitici e sperimentali di test.

Affidabilità
E' definibile in termini statistici come probabilità di assenza di malfunzionamenti in un intervallo temporale specificato. Il concetto di affidabilità rilassa in maniera pragmatica il concetto di correttezza.
La correttezza è una proprietà assoluta, mentre l'affidabilità è una proprietà relativa.
Nota: mediamente il livello di affidabilità è basso, si immette sul mercato software che si sa contenere errori (bugs).

Qualità del Software – Norme internazionali

Se le specifiche colgono esattamente i requisiti, tutti i programmi corretti sono affidabili.



Questo, tuttavia, è un quadro ideale: non è sufficiente cogliere esattamente i requisiti perchè un sistema sia affidabile.
I requisiti di partenza contenuti nelle specifiche del sistema da realizzare sono spesso, molto spesso, sbagliati, incompleti, o imprecisi.

Qualità del Software – Norme internazionali

Qualità del Software – Norme internazionali

Robustezza
Un software è robusto se si comporta ragionevolmente bene anche in circostanze non previste nelle specifiche dei requisiti (ad esempio un input errato o un malfunzionamento hardware).

Efficienza
Un software è efficiente se fa un uso ottimale delle risorse. L'efficienza si riflette nelle prestazioni del software, e di conseguenza sulla sua usabilità.
Si tratta del requisito di qualità che agli albori del'industria del software costituiva la preoccupazione prevalente, a causa degli alti costi dell'hw.
Progressivamente le dimensioni della memoria e la velocità di esecuzione sono cresciuti con una diminuzione dei costi. Ciò ha reso meno rilevante l'importanza delle prestazioni come fattore primario di qualità.
Al tempo stesso, ai fattori tradizionali che influenzano le prestazioni se ne sono aggiunti altri. Tipicamente, legati alla velocità e alla qualità dei mezzi trasmissivi.

Qualità del Software – Norme internazionali

Attenzione però, le ambizioni delle applicazioni informatiche sono pure andate crescendo di pari passo con la crescita di potenza dei dispositivi. Pertanto le prestazioni, in molti casi, continuano a costituire un aspetto chiave.

Manutenibilità
Rappresenta la facilità di un software ad essere verificabile e riparabile.
Un software è verificabile se è possibile sviluppare con facilità delle procedure di test; è riparabile se non presenta vincoli o limiti architetturali che impediscono la correzione di difetti.

Nota: L'ultima versione della norma definisce anche le possibili metriche con cui misurare ognuna delle sottocaratteristiche, con indicazione precisa della formula di misura delle modalità di calcolo e dei criteri di interpretazione dei risultati delle prove.

Qualità del Software ad Oggetti

Betrand Meyer, in aggiunta ai precedenti, elenca i criteri fondamentali della qualità del software object-oriented:

- **Estensibilità**
- **Riusabilità**

(B. Meyer, *"Object Oriented Software Construction"*, Prentice-Hall 1988)

Estensibilità
L'estensibilità è la facilità con cui si possono adattare i software ai cambiamenti di specifiche (che sono la regola e non l'eccezione: da qui nasce il bisogno di linguaggi OO). Il problema della mancanza di estensibilità è particolarmente sentito nel "Programming in the large".
Meyer indica come principali soluzioni la **Semplicità dell'Architettura** e il **Decentramento**. Infatti un'architettura semplice e non contorta è più facile da gestire, e avere tanti moduli decentrati e autonomi minimizza il rischio di reazioni a catena da un modulo all'altro: a questo proposito diventa importante definire il concetto di *modulo*.

Qualità del Software ad Oggetti

Un **Modulo** è una entità in cui è possibile suddividere un software (sia esso una *classe*, un *package* o altro) per renderlo più gestibile.
Per ottenere questo però un modulo deve essere autonomo e coerente. Meyer definisce più esattamente la modularità attraverso un insieme di 5 criteri che un metodo deve rispettare per essere "modulare" e alcuni principi che sono la logica conseguenza dei criteri:

I Criteri della modularità
scomponibilità: "divide et impera" → scomporre il problema in sottoproblemi di più facile soluzione
componibilità: posso combinare nuovi "pezzi" di software partendo da "pezzi" già esistenti
comprensibilità: il metodo usato favorisce la comprensione di un singolo "pezzo" senza conoscere gli altri "pezzi" nel dettaglio
protezione: una anomalia in un "pezzo" si propaga al più a quelli confinanti, o, meglio ancora, rimane confinata al modulo stesso.
continuità: analogamente alla definizione matematica di continuità, ne viene

definita una informatica. Ponendo sulle ascisse le specifiche e sulle ordinate l'architettura, un piccolo cambiamento di x deve produrre un piccolo cambiamento di y. Naturalmente nell'Analisi Matematica la continuità è molto più precisa, in quanto nel nostro caso non è molto facile definire cosa sia un piccolo cambiamento delle specifiche, cosa uno grande o come si misuri con precisione il cambiamento nel'architettura (numero di classi, interconnessione fra esse, numero di righe di codice nei metodi, ecc.)

I Principi della modularità

I principi di Meyer si possono invece "tradurre" brevemente come la necessità di avere un linguaggio che supporti direttamente il concetto di modulo appena espresso, che cioè non abbia bisogno di artifici ma abbia un supporto del compilatore (ad esempio le classi C++ e Java), la necessità di mantenere basso l'Accoppiamento e alta la Coesione e la necessità di poter "occultare" l'informazione (*Information Hiding*).

A livello intuitivo vediamo cosa comportano questi principi e criteri con un piccolo esempio: un "piccolo" cambiamento nelle specifiche come la necessità di

gestire date con anno a quattro cifre può portare a cambiamenti devastanti nel codice di programmi non modulari. Questo perchè la conoscenza della struttura della data (che ci deve essere e non può essere "nascosta" per sempre) è sparsa un po' ovunque nel codice. Praticamente in ogni buon vecchio programma Cobol ogni riga di codice che fa riferimento ad una data è "a conoscenza" che la data è di 2 cifre. Se si potesse spostare questa "conoscenza" in un modulo separato, pubblicare un'interfaccia valida per tutti e mantenere privata al modulo "Data" la conoscenza dei dettagli implementativi (Information Hiding), si potrebbe fare in modo che tutto il resto del codice faccia riferimento al modulo "Data" e il problema del Millennium Bug non sussisterebbe.

Qualità del Software ad Oggetti

Riusabilità La Riusabilità è la capacità dei prodotti software di essere riutilizzata, anche parzialmente, per produrre nuove applicazioni.
La riusabilità influenza tutte le altre qualità, in quanto scrivendo meno codice si possono usare più risorse per concentrarsi sulla Correttezza, la Robustezza,ecc.
Questa qualità può sembrare una chimera… ed è spesso citata dai detrattori delle tecnoòogie ad oggetti. In effetti spesso ben poco codice viene sfruttato, ed è più facile che si riutilizzino *architetture*, soluzioni tipiche (*Patterns*) che vero e proprio codice di programmazione.
Ovviamente ci sono casi in cui è facile "pensare" un framework generico e poi "riusarlo" in senso stretto per n progetti simili ed altri in cui è un po' meno facile, ma comunque il concetto generale è ugualmente valido.