Unità di apprendimento 5
Lezione 7

La valutazione
della qualità del software

Generalità

L'obiettivo di ogni produttore è quello di migliorare la qualità, cioè di ottenere un prodotto che rispetti le specifiche e soddisfi le aspettative dell'utente.

Questo diventa problematico quando riguarda il software in quanto i soggetti coinvolti nel progetto (utilizzatori e sviluppatori) hanno punti di vista diversi ed esigenze di qualità diverse.

Generalità

Gli utilizzatori ed i committenti si aspettano **efficienza**, **usabilità**, **affidabilità**; mentre gli sviluppatori si **riusabilità**, **comprensibilità**, **testabilità**, **manutenibilità**.

Inoltre, la qualità di un prodotto è sempre influenzata dalla qualità del **processo di produzione.**

Generalità

- Uno dei modelli maggiormente diffuso per la valutazione della qualità del software è quello di **McCall**, successivamente modificato da **Boehm**.
- Da esso deriva la norma ISO/IEC 9126 ***Software engineering – Product quality***, divenuta nel 2005 norma **ISO25000**.

Struttura del modello di McCall-Boehm

PRODOTTO SOFTWARE

Il prodotto software è definito come "*l'insieme di programmi, procedure, regole, documenti, pertinenti all'utilizzo di un sistema informatico*".

QUALITÀ DEL SOFTWARE

La qualità del software è definita come "*l'insieme delle caratteristiche che incidono sulla capacità del prodotto di soddisfare requisiti espliciti od impliciti*".
(Definizione molto vicina a quella riportata nella ISO 8402, il "vocabolario" della qualità ISO9000)

Struttura del modello di McCall-Boehm

Il modello ha un'architettura a 3 livelli:

- fattori, che descrivono il software da un punto di vista esterno, quello degli utenti; i fattori corrispondono a dei requisiti che sono specificati dal cliente e McCall ne individua 11;
- criteri, che descrivono gli elementi su cui agiscono gli sviluppatori per corrispondere ai requisiti del cliente: sono descritti 23 criteri;
- metriche, che servono a controllare che i criteri sviluppati corrispondano ai fattori specificati.

Un modello gerarchico con 11 fattori e 23 criteri

La qualità per l'utente coincide con l'utilità generale che l'utente associa a tre domande, anch'esse derivate dall'esperienza:

- *How well (easily, reliably, efficiently) can I use it as-is?* => usabilità
- *How easy I sit to maintain (understand, modify and test)* => manutenibilità
- *Can I use it if I change my environment?* => portabilità

Un modello gerarchico con 11 fattori e 23 criteri

USABILITÀ

Il grado con il quale è conveniente e praticabile l'uso del sistema (Boehm, 1978).
La misura dello sforzo che l'utente spende per imparare a usare il sistema (Arthur, 1985).

L' **usabilità** (***product operation***) è l'insieme delle caratteristiche del SW evidenti nella fase in cui esso è in servizio ed è definito da cinque attributi:

- correttezza
- affidabilità
- efficienza
- integrità
- Soddisfazione

Un modello gerarchico con 11 fattori e 23 criteri

MANUTENIBILITÀ

Un prodotto software possiede la caratteristica di manutenibilità nella misura in cui è facile migliorarlo al fine di soddisfare nuovi requisiti (Boehm, 1978).

La **manutenibilità** (***product revision***), è l'insieme delle caratteristiche evidenti quando si vanno ad attuare delle modifiche sul software:

- manutenibilità
- flessibilità
- testabilità

Un modello gerarchico con 11 fattori e 23 criteri

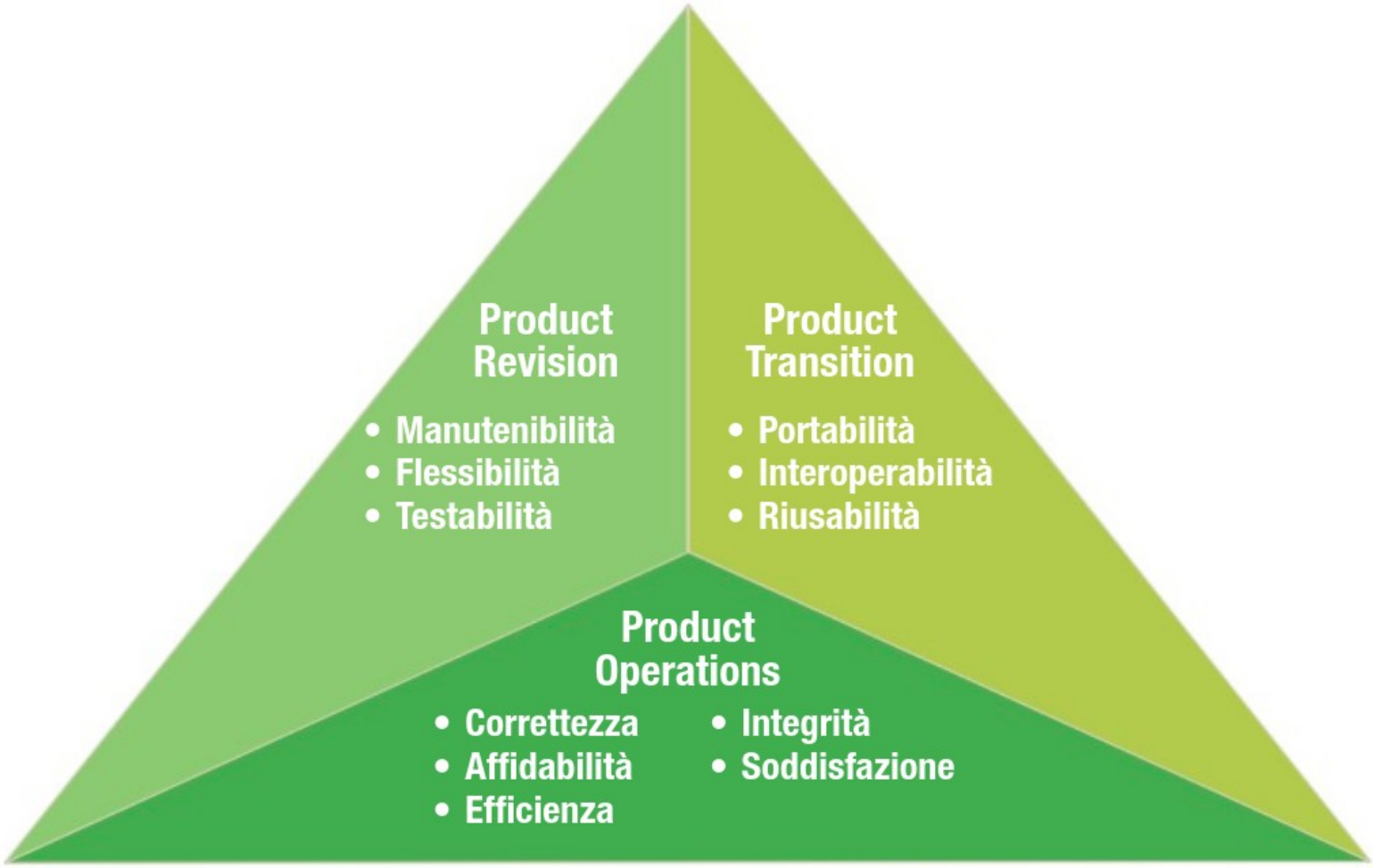
PORTATILITÀ

Un prodotto software possiede l'attributo di portatilità se esso può essere usato semplicemente su altre installazioni oltre a quella di uso (Boehm, 1978).

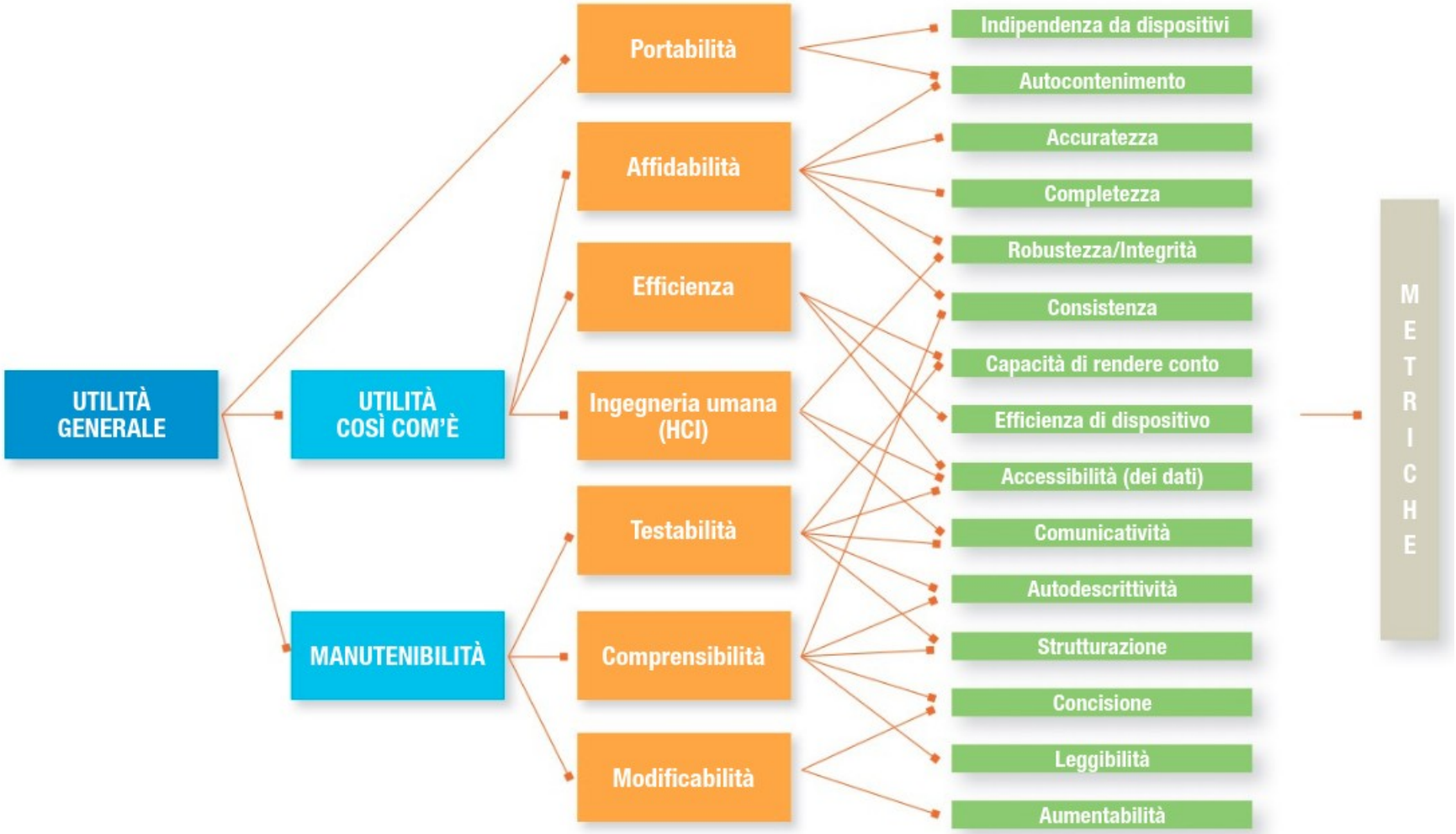
La **portabilità** (***product transition***), è l'insieme delle caratteristiche evidenti quando il software viene fatto operare su un nuovo dominio tecnologico:

- portabilità
- riusabilità
- interoperabilità

Un modello gerarchico con 11 fattori e 23 criteri



Modello di Boehm: 15 sottocategorie misurabili



Errori, difetti e malfunzionamenti

Un indicatore fondamentale di qualità del software è la presenza o meno degli errori.

Gli utenti spesso utilizzano questo elemento per valutare la bontà di un prodotto e la presenza di errori e/o malfunzionamenti possono pregiudicare il loro utilizzo o addirittura portare al fallimento di tutto il progetto.

È difficile produrre codice esente da errori, soprattutto se il progetto è complesso ed il tempo a disposizione per lo sviluppo è limitato.

Errori, difetti e malfunzionamenti

La rimozione degli errori non è sempre agevole, soprattutto quando il software è alla fine dello sviluppo o quando è in esercizio.

La soluzione ottimale “sarebbe” di ridurre al minimo gli errori in fase di sviluppo in modo da produrre **codice di qualità**.

Errori, difetti e malfunzionamenti

Le cause che determinano la qualità del codice possono essere raggruppate in tre categorie:

- **errore** (error): commesso da un essere umano (errore nel codice, nella documentazione, nei dati, nelle procedure);
- **difetto** (faults or defect): può manifestarsi a causa di un errore ed è una caratteristica fisica di una porzione di codice, di una sezione di documentazione, di una porzione dei dati, ecc…;
- **malfunzionamento** (failures): è la conseguenza di un difetto che può manifestarsi durante l'utilizzo del prodotto software.

Errori, difetti e malfunzionamenti

Non è detto che il problema si verifichi subito, anzi, spesso il fallimento di un prodotto può verificarsi dopo anni di utilizzo.

	Requisiti	Analisi e progettazione	Codifica e test unitario	Test d'integrazione	Test di sistema	Esercizio
Errori	3%	7%	53%	21%	11%	5%

Gli errori sono presenti in ogni fase del ciclo di sviluppo, anche se il grosso degli errori emerge durante la fase di codifica e test delle singole funzioni e continua a manifestarsi anche dopo la messa in servizio, almeno nei primi 18 mesi di attività.

Errori, difetti e malfunzionamenti

Le norme non garantiscono che il software sia esente da errori, ma il rispetto della normativa in ogni fase del processo di sviluppo aiuta a migliorare il prodotto e permette di ridurre i tempi di manutenzione.

Conclusioni

I tre elementi principali su cui agire per migliorare la qualità del software sono:

1. Competenza delle persone
2. Processi maturi
3. Metriche, metodi, tecniche e strumenti a supporto.

È inoltre importante ottenere una certificazione serie ISO che porta numerosi vantaggi tra cui:

- Assicurare che i dati siano affidabili e di alta qualità;
- Assicurare che il SW rispetti i requisiti funzionali;
- Migliorare la documentazione ed il manuale d'uso;
- ...