Natura e qualità del software

a cura di **Angelo Furfaro**
da "Ingegneria del Software, Fondamenti e Principi"
Ghezzi, Jazayeri, Mandrioli

Dipartimento di
Ingegneria Informatica, Elettronica, Modellistica e Sistemistica
Università della Calabria, 87036 Rende(CS) - Italy
Email: a.furfaro@unical.it
Web: http://angelo.furfaro.dimes.unical.it

La natura del software

Il software come prodotto di ingegneria

- Tutte le attività ingegneristiche hanno come scopo la costruzione di un artefatto o di un prodotto
- Il prodotto dell'ingegneria del software è un sistema software
- Il software si distingue dagli altri prodotti dell'ingegneria per la sua *duttilità*: è possibile modificare direttamente il prodotto anzichè modificare il progetto
- La duttilità del software è spesso utilizzata male: introdurre modifiche sostanziali senza riconsiderare il progetto può essere molto rischioso
- Un'altra caratteristica distintiva del software è che il suo costo finale non è determinato dalla *fabbricazione* ma dalla progettazione e dall'*implementazione*
- Mentre per gli altri prodotti è possibile definire le loro qualità in modo distinto dal progetto, per il software tale distinzione non è così netta

Classificazione delle qualità del software

Qualità interne ed esterne

- Le qualità esterne sono quelle visibili agli utenti del sistema software
- Le qualità interne riguardano gli sviluppatori
- Le qualità interne interessano la struttura del software e sono importanti per il conseguimento di quelle esterne

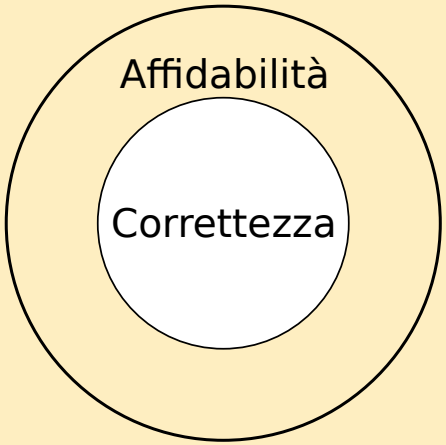
Qualità del processo e qualità del prodotto

- Per realizzare un prodotto software si utilizza un processo
- Le caratteristiche e le qualità del processo influenzano quelle del prodotto
- Il prodotto di un processo di sviluppo software non si riduce solo a quanto viene consegnato al committente (codice eseguibile e documentazione)
- Nel corso del processo di sviluppo vengono realizzati altri artefatti quali i documenti di specifica dei requisiti, i dati dei test etc.
- Anche questi altri artefatti sono soggetti agli stessi requisiti qualitativi del prodotto finale

La correttezza

- Un prodotto software deve soddisfare i suoi requisiti funzionali, ma anche quelli che riguardano le sue prestazioni, il livello di scalabilità, etc. (detti requisiti *Non funzionali*)
- La correttezza impone il soddisfacimento sia delle specifiche (funzionali) che dei requisiti non funzionali
- La definizione di correttezza assume che le specifiche del sistema siano note e che sia possibile stabilire in modo non ambiguo se esse siano soddisfatte
- La correttezza è una proprietà matematica che stabilisce l'equivalenza tra il software e la sua specifica
- Raramente le specifiche di un sistema software sono disponibili
- Quando presenti, le specifiche spesso non sono espresse in un linguaggio formale e pertanto è probabile che contengano ambiguità
- Nonostante tali difficoltà, la definizione di correttezza è importante in quanto esprime un obiettivo desiderabile dei sistemi software
- La correttezza si può *valutare* con vari metodi:
  - Testing
  - Approcci analitici: ispezione di codice, verifica formale

Affidabilità



- Un software è considerato affidabile se opera come ci si attende che esso faccia
- L'affidabilità è definita come la probabilità che il software si comporti come atteso per un certo intervallo di tempo
- La correttezza è una proprietà assoluta mentre l'affidabilità è un concetto relativo (un software non corretto può essere considerato affidabile)
- Posto che i requisiti funzionali individuino esattamente le proprietà desiderabili, i software corretti sono un sottoinsieme di quelli affidabili
- Chi assicura la correttezza dei requisiti? Un software corretto i cui requisiti non siano corretti può non comportarsi come desiderato

Robustezza

- Un sistema software è robusto se si comporta in modo accettabile anche in circostanze non previste dalle specifiche (ad esempio se vengono forniti in input dati non corretti, o in presenza di malfunzionamenti hw)
- La robustezza è qualità difficile da definire: se fosse possibile farlo equivarrebbe alla correttezza
- Il soddisfacimento di un requisito che fa parte della specifica è un problema di correttezza
- Un requisito desiderabile che non fa parte della specifica può diventare un problema di robustezza

Prestazioni

- Le prestazioni di un sistema sono influenzate dalla sua efficienza ma non si riducono ad essa
- Le prestazioni influiscono sull'usabilità del sistema
- Un sistema che utilizza troppe risorse (memoria, spazio su disco) non ha un buon livello di qualità prestazionali
- Le prestazioni influenzano la *scalabilità* del sistema ovvero la capacità di funzionare con input di dimensioni più grandi
- Le prestazioni possono essere valutate mediante:
  - Misura
  - Analisi (ad es. della complessità)
  - Simulazione
- Oltre che il prodotto software, le prestazioni possono essere viste come una qualità del processo di sviluppo ed in tal caso ci si riferisce ad esse con il termine *Produttività*

Usabilità

- Un software è considerato usabile (o user friendly) se è facile da utilizzare dagli utenti
- È una qualità soggettiva
- L'interfaccia utente influisce molto sull'usabilità
- Per i sistemi privi di interfaccia utente (ad es. embedded) l'usabilità riguarda il grado di configurabilità ed adattabilità all'ambiente hardware
- L'usabilità dipende dalla coerenza e dalla prevedibilità della sua interfaccia nei confronti del *mondo esterno* (utenti, operatori, altro software/hardware con cui deve interagire)

Verificabilità

- È molto importante poter verificare la correttezza delle prestazioni di un sistema
- La verifica può essere svolta con metodi di analisi (formale) e testing
- Spesso si utilizzano dei moduli software di monitoraggio
- L'utilizzo di metodi di progettazione modulare e di linguaggi di programmazione appropriati possono aumentare la verificabilità
- La verificabilità è una qualità interna. Per sistemi critici essa può divenire anche una qualità esterna

Manutenibilità

- La manutenzione del software riguarda l'eliminazione di difetti presenti nel prodotto software, gli interventi atti a migliorare il suo funzionamento ed introdurre nuove funzionalità (evoluzione del software) [oltre il 60% dei costi totali]
- Ci sono tre categorie di manutenzione:
  - Correttiva: eliminazione di errori (20%)
  - Adattativa: modifiche per adeguare il software a cambiamenti dell'ambiente in cui esso opera (20%)
  - Perfettiva: interventi migliorativi, introduzione di nuove funzioni/caratteristiche (60%)
- La Manutenibilità può essere vista come l'insieme di due qualità: la riparabilità e l'evolvibilità

Riparabilità

- Un sistema è riparabile se i suoi difetti possono essere corretti con una quantità ragionevole di lavoro
- In altri campi ingegneristici l'uso di componenti standard nelle parti maggiormente soggette a deterioramento aumenta la riparabilità del sistema. Ciò non vale per il software in quanto esso non si deteriora.
- Un prodotto software costituito da moduli ben progettati è più facile da analizzare e riparare
- L'incapsulamento delle informazioni, i tipi di dati astratti ed altre tecniche di modularizzazione favoriscono la riparabilità

Evolvibilità

- La *duttilità* intrinseca del software consente di effettuare modifiche direttamente sull'implementazione
- Se il software è progettato tenendo in conto la sua evoluzione e se ogni modifica viene progettata attentamente allora esso può evolvere in modo ordinato e controllato
- Tipicamente, nelle fasi iniziali i prodotti software sono in grado di evolvere facilmente se progettati con adeguati livelli di modularità
- Cambiamenti progressivi tendono a diminuire la modularità ed introdurre dipendenze
- Empiricamente risulta che la capacità di evolvere tende a diminuire con i rilasci successivi del prodotto

Riusabilità

- È una qualità simile all'evolvibilità
- Un software può essere riusato per dare origine ad un altro prodotto
- La riusabilità può coinvolgere l'intera applicazione o specifiche parti di essa (moduli, routine, etc.) Le librerie software (es la libreria standard di Java) sono esempi di software riusabile
- La riusabilità è uno degli obiettivi principali della programmazione orientata agli oggetti
- La riusabilità si applica anche al processo ed ai modelli di ciclo di vita

Portabilità

- Il software è portabile se può essere eseguito in ambienti diversi (piattaforme hardware, sistemi operativi)
- Molte applicazioni software sono portabili rispetto all'hardware poiché si basano su software di base (es. SO, DBMS) che offrono interfacce standardizzate
- La portabilità può essere favorita dall'uso di linguaggi di programmazione interpretati (es Java)

Comprensibilità

- La comprensibilità di un sistema software dipende principalmente da come è stato progettato
- Sistemi software complessi anche se ben progettati sono più difficili da capire
- Le tecniche di astrazione e modularità favoriscono la comprensibilità
- La comprensibilità gioca un ruolo fondamentale durante la fase di manutenzione
- È una qualità interna che aiuta a perseguire altre qualità quali l'evolvibilità e la verificabilità

Interoperabilità

- L'interoperabilità si riferisce alla capacità di un sistema di coesistere e cooperare con altri sistemi
- L'interoperabilità può essere raggiunta mediante l'uso di interfacce standardizzate
- Ad esempio i browser web supportano diversi tipi di *plug-in*
- È una qualità importante per i sistemi software *aperti* ovvero sistemi cui possono essere aggiunte nuove funzionalità, anche da altri produttori, dopo che è stato rilasciato
- Nell'ambito dei sistemi distribuiti un ruolo importantissimo è svolto dalle piattaforme middleware (CORBA, RMI, Web-Service, etc.)

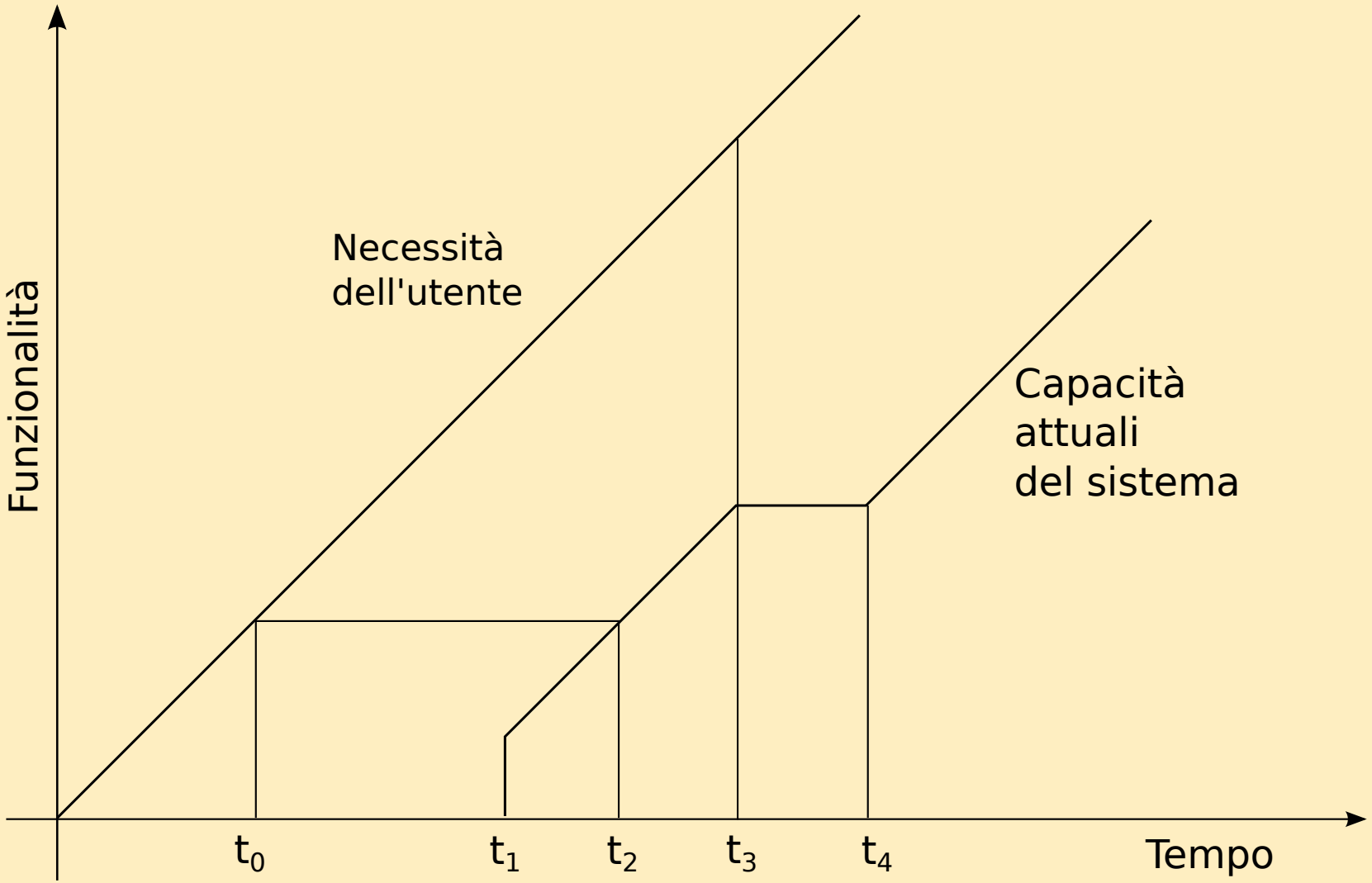
Produttività

- La produttività è una qualità del processo di produzione del software
- Indica l'efficienza e le prestazioni del processo
- Nella gestione di un processo di sviluppo si cerca di organizzare e coordinare i gruppi di lavoro in modo da sfruttare al massimo la produttività dei singoli individui
- La produttività dipende anche dalla adeguatezza del processo adottato alla categoria di prodotto sviluppato (un processo può essere ottimale per un tipo di software e non adeguato per altri)
- La riusabilità è una qualità del software che influenza la produttività del processo

Tempestività

- La tempestività indica la capacità di rendere disponibile un prodotto al momento giusto
- La tempestività non sempre è utile: non ha molto senso consegnare un prodotto entro la data prevista ma privo di qualità quali affidabilità e prestazioni
- Per perseguire adeguatamente la tempestività occorre pianificare in modo accurato la tempistica delle attività del processo e l'individuazione chiara di obiettivi intermedi
- La continua evoluzione dei requisiti dell'utente è spesso un ostacolo alla tempestività
- La consegna incrementale del prodotto può favorire la tempestività ma non è sempre applicabile

Tempestività



Visibilità

- Un processo di sviluppo è visibile se lo stato corrente e tutti i passi del processo sono documentati in modo chiaro
- La visibilità consente di valutare l'impatto delle azioni sul progetto complessivo
- La visibilità gioca un ruolo fondamentale quando c'è un avvicendamento di personale nel corso dello sviluppo