Il software: natura e qualità

Leggere Cap. 2 Ghezzi et al.

Sommario

- Natura e peculiarità del software
- Classificazione delle qualità del software
  - Qualità del prodotto e del processo
  - Qualità interne ed esterne
- Qualità desiderabili dei prodotti e dei processi software
- Qualità per tipologie specifiche di prodotti software

Il prodotto software

- Caratteristiche peculiari rispetto ad altre tipologie di prodotto
  - intangibile
  - malleabile
    - estremamente facile da modificare “fisicamente”
  - human-intensive
    - il processo di fabbricazione è banale e rappresenta una frazione piccolissima del costo
    - la produzione coincide sostanzialmente con progettazione e implementazione

Classificazione: qualità del prodotto e del processo

- **Prodotto**: il sistema consegnato al cliente
  - Ma per l'ingegnere sono significativi anche i prodotti intermedi, o artefatti
- **Processo**: la composizione delle attività che portano al prodotto
- Le qualità del processo influenzano le qualità del prodotto (es.: un processo con test accuratamente pianificati genera un prodotto più affidabile)

Classificazione: qualità esterne ed interne del prodotto

- Qualità esterne: visibili agli utenti (es.: affidabilità, prestazioni, etc.)
- Qualità interne: accessibili agli ingegneri (es.: verificabilità, rispetto delle specifiche di progetto, etc.)
- Le qualità interne influenzano le esterne (es.: la verificabilità migliora l'affidabilità)

Correttezza, affidabilità, robustezza

- Qualità esterne del prodotto (solitamente)
- Strettamente correlate
- Intuitivamente: un software possiede queste qualità se fa ciò che l'utente si aspetta, realizza le funzionalità attese

Correttezza

- Un programma è corretto se soddisfa i suoi requisiti funzionali (cioè se svolge i suoi compiti come previsto)
- Se i requisisti funzionali sono specificati formalmente, è possibile dimostrare (matematicamente) o confutare (matematicamente o empiricamente) la correttezza di un software

Limiti del concetto di correttezza

- I requisiti funzionali possono essere
  - ambigui: impossibile stabilire la correttezza del software
  - sbagliati: il software può essere corretto ma non soddisfare l'utente
- E' una proprietà assoluta
  - difficile definire un grado di correttezza
  - non tutte le "scorrettezze" sono ugualmente importanti

Metodi per verificare la correttezza

- Sperimentali:
  - testing
- Analitici:
  - ispezioni
  - prove di correttezza
  - esecuzione simbolica

Affidabilità

- Definita in termini statistici: probabilità che il software operi come atteso in un determinato intervallo di tempo
- Un software si può considerare affidabile se la sua affidabilità supera una determinata soglia(ad esempio: uptime di un server web > 99.9%)

Affidabilità

- I prodotti ingegneristici in genere hanno una garanzia
- Il software invece viene venduto con una clausola di non responsabilità

Affidabilità e correttezza

- Se i requisiti funzionali sono esatti (cioè rappresentano fedelmente le aspettative dell'utente)
  - la correttezza implica l'affidabilità
  - non vale il viceversa
- Ma i requisiti funzionali possono essere sbagliati: un software corretto può non essere affidabile

Robustezza

- Capacità del software di operare in modo accettabile in condizioni non previste (input inatteso, malfunzionamento hardware)
- Mentre la correttezza è il rispetto di requisiti specificati, la robustezza è il rispetto di requisiti non specificati
- Affidabilità e robustezza anche per i processi

Prestazioni

- Qualità esterna del prodotto (solitamente)
- Quantità di compiti che un software è in grado di svolgere in un tempo determinato
  - Esempio: servire N richieste al secondo
- Influenzate dall'hardware disponibile
- Influenzate dall'*efficienza* (uso delle risorse senza sprechi di tempo e spazio)
- Influiscono sull'usabilità

Prestazioni

- Influenzate dalla complessità degli algoritmi usati
- Per molti problemi sono stati dimostrati limiti inferiori della complessità degli algoritmi
- Influenzano la *scalabilità*, cioè la capacità di gestire quantità maggiori di input

Prestazioni

- Come valutarle?
  - studiando la complessità degli algoritmi usati
    - i risultati teorici forniscono ordini di grandezza per tempo e spazio: $O(n)$, $O(n^2)$, …
    - le prestazioni di un'implementazione dipendono anche dalle costanti: $an+b$, an^2+bn+c, …
  - studiando l'implementazione
    - misura: sistemi hardware e software di monitoraggio
    - analisi: studio analitico di un modello del prodotto (spesso basato sulla teoria delle code)
    - simulazione: misura di un modello del prodotto (più costosa e accurata)

Prestazioni

- Come migliorarle?
  - migliorando l'efficienza dei singoli moduli
  - può essere necessario riprogettare l'architettura del sistema
  - delegando alcune operazioni critiche ad hardware special-purpose
  - Parallelizzando (per esempio, *map-reduce* approach)

Usabilità

- Qualità esterna del prodotto
- Misura quanto un utente reputa il software facile da usare
- Qualità soggettiva
  - un utente principiante può gradire messaggi frequenti e prolissi
  - un utente esperto può essere infastidito
  - un utente principiante può preferire un'interfaccia grafica
  - un utente esperto può preferire una riga di comando

Usabilità

- Per sistemi interattivi dipende dalla coerenza e dalla prevedibilità dell'interfaccia utente
  - Standardizzazione delle interfacce
- Per sistemi embedded dalla facilità di installazione e configurazione
- Dipende anche da correttezza e prestazioni

Verificabilità

- Misura quanto è facile verificare le proprietà del software come correttezza e prestazioni
- Generalmente proprietà interna
- Come migliorarla?
  - progettazione modulare
  - linguaggi di programmazione adatti
  - norme di codifica
  - software monitor (codice inserito nel software per monitorare prestazioni o correttezza)

Manutenibilità

- Qualità interna del prodotto
- Manutenzione: attività di modifica del software dopo il suo rilascio
- Manutenibilità: facilità di manutenzione
- Il 60% dei costi del software è dovuto alla manutenzione

Manutenibilità

- Tre tipi di manutenzione:
  - correttiva: corregge gli errori presenti
  - adattativa: modifica il software per permetterne il funzionamento quando cambiano le condizioni operative
  - perfettiva: modifica il software per migliorarne alcune qualità
    - aggiunta di funzioni (nuovi requisiti)
    - miglioramento delle prestazioni
    - miglioramento dell'usabilità

Legacy software

- Software ereditato
- Software che esiste in un'organizzazione da lungo tempo
- Alto valore strategico, alti investimenti
- Obsoleto per tecnologie software e hardware utilizzate
- Difficile da modificare e mantenere
- Reverse engineering e reengineering

Manutenibilità

- Si può considerare l'insieme di due diverse qualità:
  - riparabilità: facilità di eliminare difetti del software (manutenzione correttiva)
  - evolvibilità: facilità di modificare il software per adattarlo a un nuovo ambiente (manutenzione adattativa) o migliorarne le qualità (manutenzione perfettiva)

Manutenibilità

- Riparabilità: migliorabile con
  - progettazione modulare
  - debole accoppiamento fra moduli
  - linguaggi di programmazione di alto livello
  - ambienti di programmazione che rendano più facile individuare gli errori

Manutenibilità

- Evolvibilità:
  - favorita da progettazione modulare
  - le modifiche dovrebbero iniziare dalla specifica e non dall'implementazione
  - l'evolvibilità tende a decrescere con l'introduzione di nuove versioni

Evolvibilità

- Molto importante: alcuni software degli anni ‘70 sono tutt’ ora in uso, ad es:
  - OS360, OS400, usato dalle banche
  - SABRE, sistema di prenotazione dei voli aerei
    - inizialmente sviluppato da IBM per American Airlines
    - attualmente utilizzato da più di 400 compagnie aeree, 30.000 agenzie di viaggio e da alberghi, ferrovie e noleggi auto

Riusabilità

- Qualità interna
- Possibilità di riutilizzare un prodotto software (o parte di esso) nella creazione di un altro, eventualmente con modifiche marginali.
- *Librerie* di componenti riusabili. Ad es:
  - librerie matematiche (inizialmente in Fortran, ora in C e C++)
  - librerie per interfacce grafiche
  - librerie per la simulazione

Riusabilità

- Facilitata da progettazione modulare.
- Uno degli scopi della programmazione orientata agli oggetti
- Anche:
  - riuso degli artefatti nei processi (specifiche, manuali)
  - riuso dei processi

Portabilità

- Qualità interna del prodotto
- Capacità del software di funzionare su piattaforme hardware e software diverse
- Hardware: differenti architetture o dispositivi (pc, palmari, cellulari)
- Software: differenti sistemi operativi o software di base come DBMS o interfaccia grafica

Portabilità

- Economicamente importante
- Facilitata da progettazione modulare (in modo da confinare le dipendenze dalla piattaforma in pochi moduli)
- Esempi: UNIX e Linux
- Importante nei sistemi distribuiti, fondamentale in applicazioni Web (Java)

Comprensibilità

- Qualità interna del prodotto
- Dipende dalla progettazione e dalla documentazione del sistema e dagli strumenti usati
- Dipende anche dalla difficoltà del problema che il software risolve
- Necessaria per la manutenibilità

Interoperabilità

- Capacità di un software di cooperare con altri
- Facilitata dalla definizione e dall'uso di interfacce standard (ad es. plug and play nei sistemi operativi)
- Permette la creazione di
  - soluzioni modulari (ad es. plugin nei browser web)
  - sistemi aperti (ad es. CORBA)

Produttività

- Proprietà del processo, inverso del tempo necessario a produrre un software
- Varia da individuo a individuo
- La produttività di una squadra dipende dalla produttività degli individui (ma è molto meno della somma!)
- Influenza i processi, e viceversa
- Riuso per aumentarla (tradeoff)
- Difficile da misurare (**metriche**)

Tempestività

- Qualità di un processo: capacità di rendere disponibile il software al momento giusto.
- Commercialmente, può essere più importante di altre qualità
- Richiede una precisa pianificazione temporale (*milestone*)
- Possibilità: consegna incrementale
- Richiede processo incrementale e progetto scomponibile

Visibilità

- Qualità del processo: misura quanto lo stato del processo è documentato in ogni momento.
- Occorre documentare
  - i passi
  - i prodotti intermedi
- Favorisce la coordinazione (evita i conflitti)
- Serve a rispondere a richieste del management o del committente sullo stato di avanzamento
- Rende più facile la sostituzione di membri della squadra (turnover).

Qualità in tipologie specifiche di prodotti software

- Alcune qualità sono particolarmente significative per determinate tipologie di prodotti
- Qualità in
  - Sistemi informativi
  - Sistemi real-time
  - Sistemi distribuiti
  - Sistemi embedded

Sistemi informativi

- Sistemi atti a gestire informazioni
- Spesso l'informazione è la risorsa più importante di un'organizzazione
- Qualità:
  - Integrità dei dati
  - Sicurezza
  - Disponibilità dei dati
  - Prestazioni delle transazioni
  - Usabilità
  - Personalizzazione (end-user computing)

Sistemi real-time

- Sistemi in cui la correttezza dipende anche dal tempo in cui avviene l'elaborazione (automazione di fabbrica, sorveglianza, controllo aereo, gestione del mouse,…)
- Spesso utilizzati in applicazioni in cui si richiede
  - alta affidabilità (sistemi mission critical)
  - safety (impossibilità del verificarsi di situazioni pericolose)

Sistemi distribuiti

- Sistemi software che funzionano su macchine diverse connesse in rete
- Consentono di migliorare affidabilità e prestazioni per mezzo di replicazione e distribuzione
- Qualità peculiari:
  - portabilità
  - livello di distribuzione (dati e/o elaborazione distribuiti)
  - capacità di tollerare il frazionamento
  - capacità di tollerare malfunzionamenti di una parte dei nodi

Sistemi embedded

- Sistemi in cui il software non ha interfacce verso l'utente finale
- Usati in aerei, robot, elettrodomestici, automobili, cellulari, …
- Scelta tra implementazione di funzionalità in hardware o software
- Spesso parti di sistemi informativi, in tempo reale, distribuiti
  - Es. sistema di monitoraggio pazienti in un ospedale

Misura delle qualità

- Individuate le qualità, è necessario verificare se un prodotto o processo le possiede (***quality assurance***)
- Nell'ingegneria del software, oggi solo alcune qualità (es. prestazioni) si possono misurare con precisione.
- La misura di altre qualità è un'area di ricerca molto attiva (metriche software).