3. Qualità

un concetto molte sfaccettature

Andrea Polini

Ingegneria del Software
Corso di Laurea in Informatica

Sommario

Sommario

Una definizione

Qualità: qualsiasi caratteristica, proprietà o condizione di una persona o di una cosa che serva a **determinarne la natura e a distinguerla dalle altre**

Nel nostro caso la “cosa” di cui si parla è il software (prodotto) oppure il processo seguito per produrlo...dunque:

Qualità nell’ambito dell’ingegneria del software: qualsiasi caratteristica, proprietà o condizione di un prodotto o processo di sviluppo che serva a determinarne la natura e a distinguerlo da altri prodotti software.

In particolare siamo interessati a quelle proprietà che ci permettono di distinguere software che forniscono le stesse funzionalità.

Qualità

È necessario definire delle **metriche** per poter associare valori ad una specifica qualità dunque per poter misurare una qualità

In generale non tutte le Qualità possono essere misurate in modo semplice e assoluto. Possono invece dipendere da fattori esterni quali il contesto d'uso o da valutazioni soggettive che fa l'attore che esegue la misurazione.

"*Formalmente una metrica definisce una corrispondenza da un insieme di entità e attributi del mondo reale su di una rappresentazione o modello nel mondo matematico con l'obiettivo di ottenere maggiori informazioni e comprensione del mondo reale*" (*Shari Pfleeger*)

Classificazione della qualità del software

Le qualità del software si possono classificare a seconda della percezione che ne hanno gli utenti (qualità esterne) o gli sviluppatori (qualità interne).
Le varie qualità possono comunque essere interrelate e mostrare contemporaneamente aspetti “esterni” ed “interni”

La caratterizzazione di qualità può poi riferirsi al prodotto o al processo.

Correttezza

esterna

Un software si dice corretto se si comporta in accordo a quanto definito nella specifica del sistema.
Come è possibile verificare la correttezza? Come viene misurata la correttezza?

Esempio: definire una libreria di supporto a manipolazioni matematiche sui naturali:

```
public class math1{
    public double sum(double x, double y)
    { return x+y; }

    public double subtract(double x, double y)
    { return x-y;}

    public double abs(double x) {
        if (x>0) {return x;}
        else {return x;}
    }
....
}
```

```
public class math2{
    public int sum(int x, int y)
    { return x+y; }

    public int subtract(int x, int y)
    { return x-y; }

    public int abs(int x) {
        if (x>0) {return x;}
        else {return -x;}
    }
....
}
```

Affidabilità

esterna

Un sistema software è affidabile se un utente può confidare nel suo comportamento. In generale l'affidabilità à definita statisticamente in base al numero di errori che si manifestano dato un certo numero di prove. Attenzione: il software spesso contiene bachi quando rilasciato. Come è possibile verificare l'affidabilità? Che tipo di misura può essere usata?

Esempio: definire una libreria di supporto a manipolazioni matematiche su numeri reali maggiori di -1:

```
public class math1{
    public double sum(double x, double y)
    { return x+y; }

    public double subtract(double x, double y)
    { return x-y;}

    public double abs(double x) {
        if (x>0) {return x;}
        else {return x;}
    }
....
```

```
public class math2{
    public int sum(int x, int y)
    { return x+y; }

    public int subtract(int x, int y)
    { return x-y; }

    public int abs(int x) {
        if (x>0) {return x;}
        else {return -x;}
    }
....
```

Robustezza

esterna

Misura di quanto il software si comporta in maniera ragionevole in circostanze non previste nella specifica.

Come possiamo verificare la robustezza? Come può essere misurata?

Esempio: definire una libreria di supporto a manipolazioni matematiche sui naturali:

```
public class math1{
    public double sum(double x, double y)
    { return x+y; }

    public double subtract(double x, double y)
    { return x-y;}

    public double abs(double x) {
        if (x>0) {return x;}
        else {return x;}
    }
....
}
```

```
public class math2{
    public int sum(int x, int y)
    { return x+y; }

    public int subtract(int x, int y)
    { return x-y; }

    public int abs(int x) {
        if (x>0) {return x;}
        else {return -x;}
    }
....
}
```

Affidabilità e Robustezza tipiche qualità di prodotto applicabili anche al processo.

Performance

esterna

Questa qualità in genere si riferisce alla velocità con cui il sistema risponde agli stimoli (latency), od anche al numero di stimoli che riesce a gestire nell'unità di tempo (throughput).
Come possiamo verificare la performance? Come possiamo misurarla?

- misurazioni a run time
- analisi
- simulazione

Lo studio si può riferire a diversi casi: ottimo, medio, pessimo

Efficienza

interna/esterna

L'efficienza si riferisce alla capacità del software di utilizzare le risorse (cpu, disco in particolare)

Le dimensioni del software si riferiscono allo spazio di memoria occupato ai diversi livelli della gerarchia di memoria nelle diverse fasi del ciclo di vita. La sua importanza è di nuovo crescente vista la crescente importanza dei sistemi embedded.
Che tipo di misura può essere utilizzato?

Usabilità

esterna

Questa qualità si riferisce alla semplicità d'uso che viene sperimentata dall'utente "obiettivo" del software. Ovviamente presenta forti fattori soggettivi. Uso di interfacce grafiche certamente aumentano usabilità di un sistema. Allo stesso tempo sistema dovrà ovviamente essere affidabile e fornire buone performance.
Che tipo di misura possiamo utilizzare? Come possiamo ricavarla?

Ruolo della standardizzazione?

Verificabilità

interna

Questa qualità fornisce una misura di quanto sia complesso verificare la correttezza del sistema. Include nozione di testabilità.
Come possiamo misurarla? Quali techniche possono essere messe in atto per aumentare la verificabilità o la testabilità?
Uso di metodi “getter” o “setter” aumentano testabilità?
Uso di pre-, post-condizione e meccanismi di eccezione.

Manutenibilità

interna ... ma

In generale si riferisce alla possibilità di modificare il prodotto una volta rilascito per “ripararlo” o “adattarlo” o “migliorarlo” (riparabilità ed evolvabilità). Come possiamo misurare la manutenibilità nei vari casi? Come può essere aumentata?
Il caso dei sistemi a plug-in.

Riusabilità

interna

Principalmente riferita a componenti software e non a interi sistemi, misura la capacità e la semplicità con cui il componente può essere utilizzato in differenti contesti.
Come possiamo migliorare riusabilità? Come può essere misurata?

Portabilità

interna ed esterna

Un software è portabile se può essere “facilmente” istallato ed utilizzato in diversi contesti e piattaforme.
Come può essere misurata la portabilità? Come può essere migliorata?
Il caso del linguaggio Java?
Non si riferisce solo al linguaggio usato.

Comprensibilità

interna

Questa qualità specifica quanto sia semplice capire a quali compiti le varie componenti del software assolvono.
Influenza fortemente altre qualità quali quelle collegate alla manutenibilità.
Come possiamo misurarla? Come può essere migliorata?

Interoperabilità

esterna ma anche interna

Si riferisce alla capacità di poter far interagire il software prodotto con altri software presenti. Un esempio “dalla notte dei tempi” è quello delle pipe di Unix.
La standardizzazione gioca un ruolo fondamentale in questo contesto!!

Misure?

Produttività

Si riferisce all'efficienza ed alla velocità con cui permette di rilasciare il prodotto. Efficienza vuol dire in particolare ridurre le risorse impegnate! Ovviamente non esiste un processo migliore ma la produttività dipenderà dal contesto. Lo stesso processo può fornire differenti valori di produttività in differenti contesti di produzione.

Timeliness

La capacità di un processo di permettere di rilasciare un prodotto in accordo alle scadenze. In generale richiede processo attento hai rischi e spesso processi incrementali offrono maggiori garanzie.

Visibilità

Si riferisce alla possibilità che hanno i vari attori partecipanti allo sviluppo di capire a che punto dello sviluppo ci si trova. In generale produzione periodica di documentazione aumenta la visibilità del processo, così come la precisa definizione di eventi di transizione. Particolarmente importante quando i team di sviluppo sono “volatili”. In questi casi altrettanto importante diventa comprensibilità riferita al prodotto.

Qualità ed ambiti specifici

Qualità definite fin qui sono di natura generale
In ambiti specifici si potranno considerare altre caratteristiche:

- Sicurezza
- Transaction performance
- Mobilità
- Safety
- ...

Riferimenti

Capitolo 2
Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli
Fondamenti di Ingegneria del Software, 2ª Ed. Italiana
Prentice Hall, 2004.