Unità di apprendimento 5

Gestione di progetti informatici

Unità di apprendimento 5

Tecniche di testing

In questa lezione impareremo...

- strategie di collaudo
- pianificazione del testing
- tipologie di testing

Tecniche di testing (collaudo) del software

Le tecniche di testing sono un insieme di metodologie adottate per collaudare il software e verificarne:

- la **correttezza**: il software fa ciò che ci si aspetta come insieme di funzionalità?
- l'**affidabilità**: il software esegue le funzioni richieste senza errori?

In generale un malfunzionamento è un comportamento del software diverso da quello atteso e, viene riscontrato durante l'esecuzione del software.

Un difetto è invece una porzione di codice scritta in modo da generare un malfunzionamento.

Il debugging (collaudo) è l'attività con cui possono essere scoperti i difetti che originano i malfunzionamenti riscontrati.

Fasi di distribuzione

Collaudo Alpha

Appena un software è stato costruito, prima di distribuirlo fuori dall'azienda, viene sottoposto ad un collaudo interno all'azienda, il collaudo Alpha.

Questo collaudo viene eseguito dagli stessi programmatori.

Collaudo Beta

- Viene eseguito in condizioni reali.
- Viene eseguito da un gruppo selezionato di utenti.
- Gli utenti segnalano gli errori → viene sviluppata una successiva versione Beta.
- Quando la frequenza degli errori risulta sufficientemente "bassa", si pubblica la versione "ufficiale".
- Nell'azienda può comunque continuare il collaudo Alpha mentre è in corso quello Beta.

Strategia di collaudo

Il collaudo avviene a diversi livelli:

- Unit testing: eseguito a livello di singolo modulo (es: una classe)
- Test di integrazione: eseguito su più moduli integrati (riuniti)
- Test di sistema: eseguito sull'intero sistema (software e hardware)

Per rilevare il maggior numero possibile di difetti, occorre eseguire il programma con diverse combinazioni di input, che permettano di provare “tutti i casi”, cioè di fare eseguire la maggior quantità possibile di codice.

Strategia di collaudo

Purtroppo le combinazioni di input validi possono essere tantissime, e diventa impraticabile provarle tutte, cioè fare un test esaustivo. L'obiettivo diventa quindi quello di avere una “bassa” probabilità di malfunzionamenti.

Il valore di questa probabilità dipende dal tipo di applicazione sviluppata. Le due tipologie estreme di applicazioni sono:

- software life-critical che è un software funzionante su apparecchiature biomedicali o aeronautiche);
- software di produttività personale, come office automation o videogiochi.

Quando termina il testing?

"*Le operazioni di testing possono individuare la presenza di errori nel software ma non possono dimostrarne la correttezza*" (Dijkstra 1972)

Per determinare il momento di conclusione del testing, possono essere adottati alcuni criteri:

- temporale: il tempo da dedicare al test è predefinito;
- di costo: è stato raggiunto il limite di budget per tale attività;
- di copertura: tutte le righe del programma sono state provate almeno una volta;
- statistico: negli ultimi k test, non si sono riscontrati malfunzionamenti;

Pianificazione del testing

Poiché il collaudo del software è una fase particolarmente importante, richiede un'attenta pianificazione, cioè la **produzione di un piano di collaudo**, costituito dai seguenti elementi:

- una **check list**: elenco delle prove (test case) manuali o automatiche, ripetibili più volte;
- degli **scenari di collaudo**, cioè situazioni realistiche di uso del software (tipologie di utente, casistiche in cui si può venire a trovare l'utilizzatore, ecc.).

Check list

La check list è un insieme di test case che generano un output tabellabile con la seguente struttura:

TC (id)	Precondizione per poterlo eseguire	Input	Output attesi	Post condizioni attese (ex. Modifiche archivi)	Output riscontrato	Post condizione riscontrata	Esito	Priorità

Tipologie di testing

Test funzionali

Lo scopo di questi testi è quello di scoprire i difetti che originano malfunzionamenti.

Ve ne sono di due tipi complementari:

- white box testing: si eseguono sul codice (es. debugging);
- black box testing: si fanno accedendo al software tramite la sua interfaccia utente o tramite l'interfaccia di comunicazione fra processi (nel caso di test automatico). Spesso sono condotti da persone esterne al gruppo di sviluppo.

Test prestazionali

Lo scopo di questi test è di verificare alcune qualità del software in termini di prestazioni.

Per esempio possono essere svolti:

- test di performance: verificano i tempi di esecuzione;
- test di carico: verificano le reazioni del sistema al crescere del carico.

Tipologie di testing

Test funzionali – White box testing

Vengono collaudati i "**cammini di base**".

Un cammino di base (o indipendente) è costituito da un insieme di istruzioni in cui ve ne siano di nuove, rispetto ad altri cammini di base, o vi sia una nuova condizione.

Tutti i cammini indipendenti devono essere eseguiti almeno una volta e, quindi, sarà necessario preparare un test case per ogni cammino.

Il grafo di flusso di tutti i cammini è derivabile dal flow chart.

Tipologie di testing

Test funzionali – Black box testing

Questo test collauda il comportamento del software, cioè se sono stati soddisfatti i requisiti software.

E' usato soprattutto nelle ultime fasi del testing e risulta complementare al white box testing.

Nella preparazione dei casi prova, si opera una suddivisione degli input in classi di equivalenza, cioè per gruppi di input che il programma deve elaborare nello stesso modo.

Esempi di errori rilevati:

- funzioni mancanti o incomplete;
- errori nelle strutture dati;
- errori di interfaccia.