UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

DIPARTIMENTO DI INFORMATICA

CORSO DI LAUREA IN
INFORMATICA E TECNOLOGIE PER LA PRODUZIONE DEL SOFTWARE

Una definizione tassonomica dei Test Software

Relatori:
Chiar.mo prof. **Michele SCALERA**
Dott. **Giovanni BRUNO**

Laureando: **Angelo RICATTI**

SERLAB
SOFTWARE ENGINEERING RESEARCH LABORATORY
c/o DIPARTIMENTO DI INFORMATICA
via Orabona, 4 - 70126 - Bari
Tel.: + 39.080.5443279 Fax: + 39.080.5442536

INDICE

- Software Testing
- Obiettivo
- Ricerca
- Esperimento
- Sperimentazione
- Risultati
- Conclusioni

Software Testing

È lo strumento che permette di accertarsi, senza grossi sforzi, che il software sia di qualità e che funzioni rispettando le specifiche.

L'uso dei test prima della vendita del prodotto consente di:

- ridurre la probabilità che si verifichino errori durante il suo utilizzo;
- assicurarsi che il software implementato rispecchi i requisiti funzionali del sistema.

Il test ha due approcci:

- Test funzionale o Black Box:
  - Verifica la conformità del sistema con le specifiche, basandosi solamente sull'osservazione di input e output senza analizzare la struttura interna del software.
- Test strutturale o White Box:
  - Analizza i meccanismi interni della struttura di un software e rileva eventuali errori di implementazione: esige una buona conoscenza di come il sistema è stato implementato.

Esiste anche una combinazione di questi due approcci chiamato Grey Box.

Obiettivo

Analisi di una copiosa bibliografia, al fine di definire una tassonomia chiara riguardo al software testing.

⇨ Questa tassonomia:

- Contiene una chiara classificazione delle tipologie di test del software.
- Raggruppa gli stessi test a cui sono stati attribuiti, nei diversi documenti, nomi differenti.

Ricerca

Dall'analisi della letteratura sono stati ritrovati circa cinquanta test classificati in quattro macro-livelli descritti in quasi tutti i documenti:

- Unit test
- Integration test
- System test
- User test

Unit test (o Module test, Small-scale test, Isolation test, test delle componenti)

Il test di unità è il processo con cui vengono provati i componenti di un programma, come i metodi o le classi di oggetti.

⇨ Test di mutazione

- Utilizzato per valutare la qualità delle test case di unità di un software attraverso dei programmi mutanti.

Integration test

Il test di integrazione è utilizzato per testare l'integrazione tra i moduli e ritrovare i bug non rilevati nel test di unità. Esso comprende:

- Intra-system test
  - Costituisce test di integrazione di basso livello con l'obiettivo di combinare insieme i moduli per costruire un sistema coeso.
- Inter-system test
  - È una fase di test di alto livello che richiede l'interfacciamento di sistemi testati in modo indipendente. In questa fase, tutti i sistemi sono connessi tra di loro ed è condotto un test end to end.

- End-to-end test
  - È una metodologia utilizzata per verificare se il flusso di un'applicazione si sta comportando come progettato dall'inizio alla fine.
- Pairwaise test
  - Garantisce che due sistemi possano funzionare insieme, presumendo che gli altri sistemi all'interno dell'ambiente si comportino come previsto.

System test (o System-level testing)

Il test di sistema ha lo scopo di provare il corretto comportamento del sistema software nella sua interezza rispetto ai vincoli e ai requisiti espressi dal committente.

- ⇨ Basic test
  - Vengono eseguiti per garantire che le funzioni di uso comune funzionino in modo soddisfacente. (Es. avvio, upgrade/downgrade, ecc.)
- ⇨ Scalability test
  - Ha lo scopo di verificare che il sistema possa scalare fino ai propri limiti di progettazione testando i limiti del sistema.

- Documentation test
  - Consiste nel verificare l'accuratezza tecnica e la leggibilità dei manuali dell'utente, inclusi i tutorial e la guida on-line.
- Interoperability test
  - È progettato per verificare l’abilità del sistema di interoperare con prodotti di terze parti. Contengono i test di configurazione e test di compatibilità.
- Performance test
  - È un controllo mirato alla valutazione dell’efficienza di un sistema rispetto ai tempi di elaborazione e di risposta.

- Functionality test (o Facility test)
  - Verifica il sistema sull'intera gamma di requisiti definiti nel documento delle specifiche (compresi quelli di sicurezza, GUI, comunicazione, ecc.), utilizzando valori moderati.
- Stress test (o Workload test)
  - Il sistema è sottoposto a carichi di lavoro superiori a quelli previsti dai requisiti o è portato in condizioni operative eccezionali sottraendo risorse di calcolo e di memoria.
- Reliability test
  - È progettato per misurare la capacità del sistema di rimanere operativo per lunghi periodi di tempo.

- Robustness test
  - È progettato per verificare che comportamento assume il sistema in situazioni di errore e in un ambiente operativo modificato. (Es. presenza di glitch, rimozione moduli online, presenza «nodi degradati», ecc.)

User test

Il test degli utenti è un processo di test in cui gli utenti o clienti danno i loro input e suggerimenti sui test del sistema.

- Alpha test
  - Utenti selezionati operano a fianco del team di sviluppo per testare le prime release del software.
- Beta test
  - È condotto dai potenziali acquirenti prima del rilascio ufficiale del prodotto, avendo come scopo quello di ottenere un feedback sull'usabilità del prodotto.
- Acceptance test
  - Ha lo scopo di verificare che il software soddisfi il suo scopo funzionale secondo il punto di vista dell'utente.

Regression test (o Test di non regressione)

Il test di non regressione controlla che eventuali modifiche al codice non abbiano introdotto nuovi bug nel sistema, accertandosi che non sia regredito. Esso è presente in tutti e quattro i livelli.

Esperimento

- Studio e implementazione di un modello di decisione per la scelta di framework o tool per il software testing.
- Cinquantuno framework e tool sono stati classificati in base alle seguenti domande:
  - Che tipo di sistema è?
    - Web, Java, Mobile, API, Database
  - Che test occorre effettuare?
    - Unit, Integration, Functional, Load, Performance, Regression, Acceptance
  - Qual è il linguaggio di scripting?
    - JS, Python, Java, Ruby, PHP
  - Qual è il tipo di licenza del tool?
    - Open Source Software (OSS), Proprietary Software (PS)

Sperimentazione

- Test di unità di due sistemi diversi con tool e framework scelti attraverso il modello definito.
- Scenari
  - Sistema Web scritto in PHP
  - Sistema Java scritto nell'omonimo linguaggio
- Tool/Framework utilizzati
  - Primo sistema: Codeception
  - Secondo sistema: JUnit, testNG, Mockito

Lo scopo della sperimentazione è stato quello di confrontare diversi framework per valutare qual è da considerarsi il migliore.

⇨ Criteri di valutazione

- Tempi di produzione;
- Curva di apprendimento;
- Tempi di esecuzione con diversi runner;
- Integrazione con altri tool, framework e IDE;
- Funzionalità di ciascun tool/framework.

Risultati

- Primo sistema
  - Non avendo alternative nel modello decisionale con licenza OSS è stato analizzato solo Codeception.
  - Codeception ha avuto riduzione dei tempi di produzione pari quasi al 50% a fine esperimento.
- Secondo sistema
  - Tempi di produzione simili tra loro, con testNG leggermente inferiore ai suoi concorrenti.
  - Con runner testNG, testNG ha tempi di esecuzione più rapidi di JUnit.

- Con runner JUnit i tempi di esecuzione di JUnit sono più rapidi di testNG eseguito con il suo omonimo runner.
- Mockito applicato a JUnit, in fase di esecuzione, è più rapido dello stesso applicato a testNG.
- TestNG e JUnit sono integrabili con una quantità di tool, framework e IDE pressoché simile tra loro e ne condividono molti. Mockito è integrabile principalmente con JUnit, testNG, PowerMock.
- Dal punto di vista funzionale testNG, rispetto a JUnit, permette operazioni particolari prima o dopo l'esecuzione di determinati test, suite e/o gruppi di test. Mockito invece aggiunge la possibilità di eseguire test con oggetti simulati.

Conclusioni

- Si è cercato di svolgere un lavoro proficuo e completo nell'ambito del software testing, provando a definire uno standard generale per la materia.
- Sperimentalmente, non è stato ritrovato un framework da considerarsi in ogni caso il migliore, ma la scelta dipende dalle necessità.

  Concludendo, è preferito:
  - JUnit se non si richiedono operazioni particolari;
  - TestNG se si necessitano operazioni prima o dopo l'esecuzione di determinati test, suite e/o gruppi di test;
  - Mockito se occorre effettuare test con mock object.

Grazie per l’attenzione.