Certificazione di Tester

Syllabus Livello 'Advanced'

Test Analyst

Versione 2012

International Software Testing Qualifications Board

ITAlian - Software Testing Qualifications Board

Sottogruppo di lavoro per il Advanced Level Test Analyst: Judy McKay (presidente), Mike Smith, Erik Van Veenendaal

Versione	Data	Note
ISEB v1.1	04SEP01	ISEB Practitioner Syllabus
ISTQB 1.2E	SEP03	ISTQB Advanced Level Syllabus from EOQ-SG
V2007	12OCT07	Certified Tester Advanced Level syllabus version 2007
D100626	26JUN10	Incorporation of changes as accepted in 2009, separation of chapters for the separate modules
D101227	27DEC10	Acceptance of changes to format and corrections that have no impact on the meaning of the sentences.
D2011	23OCT11	Change to split syllabus, re-worked LOs and text changes to match LOs. Addition of BOs.
Alpha 2012	09MAR12	Incorporation of all comments from NBs received from October release.
Beta 2012	07APR12	Incorporation of all comments from NBs received from the Alpha release.
Beta 2012	07APR12	Beta Version submitted to GA
Beta 2012	08JUN12	Copy edited version released to NBs
Beta 2012	27JUN12	EWG and Glossary comments incorporated
RC 2012	15AUG12	Release candidate version - final NB edits included
GA 2012	19OCT12	Final edits and cleanup for GA release

Storico delle Revisioni di traduzione per ITA-STQB

REV.	AUTORI	DESCRIZIONE DELLE MODIFICHE	DATA DI APPROVAZIONE DEL COMITATO SCIENTIFICO
01	M.Sogliani / R.Rosci	Traduzione del syllabus TA Advanced Level	23 Novembre 2012
02	S. Reale	Revisione di errata corrige	12 Agosto 2013

Ringraziamenti

Questo documento (versione 2012) è stato prodotto da un team dedicato appartenente al gruppo di lavoro dell’International Software Testing Qualifications Board Advanced Level Test Analyst:

Judy McKay (chair), Mike Smith, Erik van Veenendaal.

Il team desidera ringraziare il gruppo dei revisori e tutte le organizzazioni nazionali per i suggerimenti e gli input.

Al momento in cui l’Advanced Level Syllabus è stato completato, il gruppo di lavoro Advanced Level era costituto dai seguenti membri (in ordine alfabetico):

Graham Bath, Rex Black, Robert Bender, Chris Carter, Maria Clara Choucair, Dorothy Graham, Bernard Homès (Vice-Chair), Jayapradeep Jiothis, Vipul Kocher, Anastasios Kyriakopoulos, Judy McKay, Thomas Mueller, Klaus Olsen, Avinoam Porat, Meile Posthuma, Erkki Pöyhönen, Jürgen Richter, Eric Riou du Cosquer, Jan Sabak, Hans Schaefer, Maud Schlich, Rajesh Sivaraman, Mike Smith (chair), Michael Stahl, Geoff Thompson, Erik van Veenendaal

Hanno partecipato a revisione, commento e votazione di questo syllabus:

Graham Bath, Arne Becher, Rex Black, Piet de Roo, Frans Dijkman, Mats Grindal, Kobi Halperin, Bernard Homès, Maria Jönsson, Junfei Ma, Eli Margolin, Rik Marselis, Don Mills, Gary Mogyorodi, Stefan Mohacsi, Reto Mueller, Thomas Mueller, Ingvar Nordstrom, Tal Pe'er, Raluca Madalina Popescu, Stuart Reid, Jan Sabak, Hans Schaefer, Marco Sogliani, Yaron Tsubery, Hans Weiberg, Paul Weymouth, Chris van Bael, Jurian van der Laar, Stephanie van Dijk, Erik van Veenendaal, Wenqiang Zheng, Debi Zylbermann.

Questo documento è stato formalmente rilasciato dall’Assemblea Generale dell’ISTQB® il 19 Ottobre 2012.

0. Introduzione a questo syllabus

0.1 Scopo di questo Documento

Questo syllabus costituisce la base per l’International Software Testing Qualification di Livello Avanzato (Advanced Level) per Test Analyst.

L’ISTQB® mette a disposizione questo syllabus con le seguenti modalità:

1. Ai National Board membri, per consentirne la traduzione nella loro lingua madre e di accreditare i fornitori della formazione. I Board nazionali possono adattare il syllabus alle loro particolari esigenze linguistiche e modificare i riferimenti bibliografici per adattarli alle loro pubblicazioni locali.
2. Agli Enti Esaminatori, per consentire la traduzione delle domande d’esame nella loro lingua adattandole agli obiettivi di apprendimento di ogni modulo.
3. Ai Fornitori della formazione, per consentire la produzione di materiale didattico e mettere a punto metodologie d’insegnamento appropriate.
4. Ai Candidati alla certificazione, per consentire la preparazione all’esame (a séguito della partecipazione a un corso o in modo indipendente).
5. Alla comunità internazionale di ingegnerizzazione dei sistemi e del software, per consentire di migliorare la professione di tester del software e dei sistemi e per fornire una base di riferimento per libri ed articoli.

L’ISTQB® permette inoltre ad altre entità di usare questo syllabus per altre finalità, purché ne chiedano ed ottengano anticipatamente un permesso scritto.

0.2 Panoramica

Il livello avanzato è composto da tre Syllabi distinti:

- Responsabile del Test
- Analista del Test
- Analista Tecnico del Test

Il documento Advanced Level Overview [ISTQB_AL_OVIEW] include le seguenti informazioni per ognuno dei syllabi:

- Risultati di business
- Sommario
- Relazioni fra i syllabi
- Descrizione dei Livelli di Conoscenza (Livelli K)
- Appendici

Obiettivi di Apprendimento Esaminabili

Gli obiettivi di apprendimento supportano i risultati di business e sono utilizzati per progettare gli esami volti ad ottenere la certificazione Advanced Test Analyst. In generale tutte le sezioni di questo syllabus sono esaminabili ad un livello K1, nel senso che il candidato dovrà riconoscere, ricordare e richiamare un termine o un concetto. Gli obiettivi di apprendimento per i livelli K2, K3 e K4 sono mostrati all'inizio di ogni pertinente capitolo.

1. Processo di Test – 300 minuti

Termini:

test case concreti, criteri di uscita, test case di alto livello, test case logici, test case di basso livello, controllo del test, progettazione del test, esecuzione del test, implementazione del test, pianificazione del test.

Obiettivi di Apprendimento per il Processo di Test

1.2 Il Testing nel Ciclo di Vita Sviluppo Software

TA-1.2.1 ((K2) Spiegare come e perché i tempi e il livello di coinvolgimento per l'Analista del Test variano quando si lavora con modelli del ciclo di vita diversi (ad esempio, V-model, iterativi incluso Agile)

1.3 Pianificazione, Monitoraggio e Controllo

TA-1.3.1 (K2) Riassumere le attività svolte dall’Analista del Test a sostegno della pianificazione e controllo del processo di test

1.4 Analisi

TA-1.4.1 (K4) Analizzare un determinato scenario, compresi una descrizione del progetto ed il modello del ciclo di vita, per determinare il miglior momento di coinvolgimento e le attività appropriate per l'Analista del Test durante le fasi di analisi e di progettazione

1.5 Progettazione

TA-1.5.1 (K2) Spiegare perché le condizioni di test devono essere ben comprese da parte degli stakeholders.
TA-1.5.2 (K4) Analizzare uno scenario di progetto per determinare l'uso più appropriato di test case di basso livello (concreti) e di alto livello (logici)
TA-1.5.3 (K2) Descrivere i vantaggi dell'utilizzo dell‘analisi, progettazione ed implementazione del test per scoprire i difetti

1.6 Implementazione

TA-1.6.1 ((K2) Descrivere i tipici criteri di uscita dell'analisi e della progettazione del test e spiegare come il rispetto di tali criteri incida sullo sforzo di implementazione del test.

1.7 Esecuzione

TA-1.7.1 (K3) In un determinato scenario, determinare i passi e le considerazioni di cui tenere conto quando si registra un difetto

1.8 Valutazione criteri di uscita e Reporting

TA-1.8.1 (K2) Spiegare l'importanza di raccogliere delle metriche accurate

1.9 Attività di chiusura

TA-1.9.1 (K2) Fornire modelli ed esempi che possono essere adottati dall’Analista del Test durante le attività di chiusura del test.

1.1 Introduzione

Il syllabus ISTQB ® Foundation Level descrive un processo fondamentale di test che comprende le seguenti attività:

- Pianificazione, monitoraggio e controllo
- Analisi e progettazione
- Implementazione ed esecuzione
- Valutazione dei criteri di uscita e Reporting
- Attività di chiusura

Per il syllabus di livello avanzato alcune di queste attività sono state considerate separatamente, al fine di arrivare a un ulteriore affinamento e ottimizzazione dei processi perché meglio si adattino al ciclo di vita di sviluppo del software, e facilitare un efficace monitoraggio e controllo dei test.

Le attività di test sono perciò raggruppate come segue:

- Pianificazione, monitoraggio e controllo
- Analisi
- Progettazione
- Implementazione
- Esecuzione
- Valutazione dei criteri di uscita e Reporting
- Attività di chiusura

Queste attività possono essere attuate tutte in sequenza o qualcuna può essere svolta in parallelo; ad esempio, la progettazione può essere eseguita in parallelo con la realizzazione (ad esempio, test sperimentali). Le principali aree di attenzione per l'Analista del Test sono proprio la definizione dei test e test case appropriati, la loro progettazione e la loro esecuzione. Benché sia importante capire tutti i passi del processo di test, la maggior parte del lavoro dell'Analista del Test viene di solito svolto durante le attività di analisi, progettazione, implementazione ed esecuzione del progetto di test.

I tester specializzati devono affrontare una serie di sfide quando promuovono i diversi aspetti del testing descritti in questo syllabus nel contesto delle proprie organizzazioni, dei propri team e nelle relative attività. È importante considerare come i diversi cicli di vita di sviluppo software, così come il tipo di sistema da testare possano influenzare l'approccio al test.

1.2 Il Testing nel Ciclo di Vita Sviluppo Software

L'approccio a lungo termine del testing nel ciclo di vita del software deve essere considerato e definito come parte della strategia di testing. Il momento del coinvolgimento per l'Analista del Test differisce nei vari cicli di vita, ma anche la quantità di coinvolgimento, l’impegno richiesto, le informazioni disponibili e le aspettative possono variare di molto. Poiché le attività di test non si svolgono in isolamento, l'Analista del Test deve essere consapevole di quando le informazioni possono essere fornite agli altri settori organizzativi, quali:

- Ingegneria e gestione dei requisiti – durante la revisioni dei requisiti
- Gestione del progetto – durante la definizione della schedulazione
- Gestione della configurazione e delle modifiche – durante il test di verifica delle build e il controllo delle versioni
- Sviluppo del software - quando si prevedono anticipatamente i prodotti da testare
- Manutenzione del software - nella gestione dei difetti: tempi di risoluzione (cioè dalla scoperta del difetto alla risoluzione)
- Supporto tecnico - una documentazione accurata per le soluzioni temporanee
- Produzione di documentazione tecnica (ad es. specifiche di progettazione dei database) – quando si forniscono informazioni per la redazione di questi documenti e durante la loro revisione tecnica

Le attività di testing devono essere allineate al modello ciclo di vita di sviluppo software prescelto, la cui natura può essere sequenziale, iterativa o incrementale. Ad esempio, nel modello V-sequenziale, il fondamentale processo di test ISTQB ® applicato al livello di test di sistema potrebbe allinearsi come segue:

- La pianificazione del System Test si svolge insieme alla pianificazione di progetto e il controllo del testing prosegue fino all’esecuzione dei test ed alla sua completa chiusura.
- L’analisi e la progettazione del System Test si svolgono insieme alla specifica dei requisiti, alle specifiche di progettazione di sistema e di architettura (ad alto livello) ed alle specifiche di progettazione di componente (basso livello).
- La realizzazione dell’ambiente per il System Test (ad esempio, test bed, dati di test) potrebbe iniziare in fase di progettazione del sistema, anche se la maggior parte delle fasi correlate a tale attività in genere vengono portate a termine in concomitanza con la codifica ed il test dei componenti e le attività di implementazione dell’ambiente si estendono spesso fino a pochi giorni prima dell'inizio del System test.
- L’esecuzione del System Test inizia quando i criteri di ingresso al System Test sono tutti soddisfatti (o abbandonati espressamente), il che in genere significa che almeno il test di componenti e spesso anche il test di integrazione dei componenti sono completati. L’esecuzione del System Test continua fino a che i criteri di uscita dal test siano stati soddisfatti.
- La valutazione dei criteri di uscita ed il report dei risultati si svolgono durante tutta l'esecuzione del System Test, in genere con maggiore frequenza ed urgenza man mano che la scadenza del progetto si avvicina.
- Le attività di chiusura del System Test si effettuano dopo che i criteri di uscita sono stati soddisfatti e che l'esecuzione del System Test è stata dichiarata conclusa, anche se a volte possono essere ritardate sino a che il Test di Accettazione e tutte le attività del progetto siano stati completati.

I modelli iterativi e incrementali possono non seguire lo stesso ordine di attività e possono escludere alcune di esse. Ad esempio, un modello iterativo può utilizzare un insieme ridotto dei processi standard di test previsti per ogni iterazione.

L’analisi e la progettazione, l’implementazione e l’esecuzione, la valutazione ed il reporting possono essere effettuati per ogni iterazione, mentre la pianificazione viene fatta all'inizio del progetto ed il reporting di chiusura viene effettuato alla fine.

In un progetto Agile, è prassi comune utilizzare un processo meno formalizzato e relazioni di lavoro molto più strette, che consentano di apportare modifiche in modo semplificato. Poiché quello Agile è un processo "leggero", c’è minor attenzione alla completezza della documentazione di test, in favore di un metodo che consenta comunicazioni più veloci come le riunioni giornaliere di tipo “stand up”, chiamate così proprio perché durano 10-15 minuti, non c’è necessità di sedersi e tutti i partecipanti vi rimangono coinvolti.

I progetti di tipo Agile, contrariamente ad altri modelli del ciclo di vita, richiedono un intervento anticipato da parte degli Analisti di Test. L'Analista del Test deve aspettarsi di essere coinvolto fin dall’avvio del progetto, collaborando con gli sviluppatori nel lavoro di architettura iniziale e di progettazione. Le revisioni possono non essere formalizzate, e svolgersi continuamente durante l’evoluzione del software. Il coinvolgimento dovrebbe essere a tempo pieno durante tutto l’arco del progetto e l’Analista del Test dovrebbe essere una risorsa che il gruppo di lavoro dovrebbe avere nelle sue disponibilità effettive. A causa di ciò, i membri dei team Agile sono di solito dedicati al singolo progetto e sono pienamente coinvolti in tutti gli aspetti del progetto stesso.

I modelli Iterativi / incrementali vanno da un approccio Agile, in cui c’è un’aspettativa di modifica continua in relazione alle modifiche del software, a quelli basati sul modello a V (a volte chiamati “iterativi incorporati”). In quest’ultimo caso, l'Analista del Test può essere coinvolto nelle consuete attività di pianificazione e di progettazione standard, ma deve aspettarsi un coinvolgimento ancora

maggiore e una più marcata interazione non appena il software viene sviluppato, testato, modificato e distribuito.

Qualunque sia il ciclo di sviluppo software utilizzato, è importante che l'Analista del Test capisca le aspettative nonché la durata del suo coinvolgimento. Poiché ci sono molti modelli ibridi in uso, come il modello iterativo inserito nel modello V precedentemente menzionato, l'Analista del Test deve interpretare il suo ruolo più efficace al loro interno, piuttosto che seguire quanto previsto da un modello pre-impostato.

1.3 Pianificazione, monitoraggio e controllo del Test

Questa sezione si concentra sui processi di pianificazione, monitoraggio e controllo del testing.

1.3.1 Pianificazione del test

La pianificazione del test si svolge, nella maggior parte dei casi, all'inizio delle attività di test e comporta l'identificazione e la schedulazione di tutte le attività e le risorse necessarie per soddisfare la missione e gli obiettivi identificati nella strategia di test. È importante che l'Analista del Test collabori, durante la pianificazione, con il Responsabile del Test, nel considerare e pianificare quanto segue:

- Assicurarsi che i piani di test non si limitino al test funzionale. Nel piano di test dovrebbero essere considerate e programmate di conseguenza tutte le tipologie di test. Ad esempio, oltre che del test funzionale, l’Analista del Test può essere responsabile del test di usabilità: questa tipologia di test deve essere prevista nel Piano di test.
- Rivedere le stime con il Responsabile del Test e assicurarsi che sia inserita nel budget una tempistica adeguata per l’acquisizione e la validazione dell'ambiente di test.
- Pianificare il test di configurazione. Se sono possibili differenti combinazioni tra tipi di processori, sistemi operativi, macchine virtuali, browser diversi, periferiche etc., si deve prevedere di utilizzare tecniche di test che forniscano una copertura adeguata di tali diverse combinazioni.
- Pianificare il test della documentazione. Poiché agli utenti viene fornito sia il software sia la documentazione, questa deve essere corretta ed accurata per essere efficace. L'Analista del Test deve allocare sufficiente tempo per verificare la documentazione e potrebbe essere necessario che lavori con i “technical writers” per preparare i contenuti delle schermate e dei video clip di assistenza.
- Pianificare il test delle procedure di installazione; queste, al pari di quelle di backup e di ripristino, devono essere sufficientemente testate in quanto possono essere più critiche del software stesso, se si pensa che se il software non può essere installato, tanto meno può essere utilizzato. Questo può essere difficile da programmare, in quanto l'Analista del Test effettua spesso i test iniziali in un sistema che è stato pre-configurato, sul quale non ha neanche avuto luogo il processo di installazione finale.
- Pianificare il testing per allinearsi con il ciclo di vita del software. L’esecuzione sequenziale delle attività non si adatta alla maggior parte delle schedulazioni; molte attività hanno spesso bisogno di essere eseguite in parallelo, magari solo in parte. L'Analista del Test deve essere a conoscenza del ciclo di vita adottato e delle aspettative sul suo coinvolgimento nel corso della progettazione, sviluppo e implementazione del software. Ciò richiede anche l’allocazione del tempo necessario per il test di conferma e di regressione.
- Allocare il tempo sufficiente per l'identificazione e l'analisi dei rischi con il team interfunzionale. Anche se di solito non è sua responsabilità organizzare le sessioni di gestione del rischio, l’Analista del test deve aspettarsi di essere coinvolto attivamente in queste attività.

Ci possono essere interazioni complesse tra la base di test, le condizioni di test ed i test case, tali da configurare relazioni “molti-a-molti” tra questi prodotti di lavoro. Tali relazioni devono essere ben comprese per consentire una pianificazione ed un controllo del testing veramente efficaci. L'Analista del test è di solito la persona più adatta a identificare queste relazioni e a semplificarne quanto più possibile tali dipendenze.

1.3.2 Monitoraggio e controllo del test

Il controllo del test è di norma appannaggio del Responsabile del Test, ma l’Analista del Test contribuisce a individuare le metriche e a rendere possibile tale controllo.

Le misurazioni (anche prettamente numeriche) andrebbero raccolte durante tutto il ciclo di vita del software (ad esempio, la percentuale delle attività pianificate già completate, la percentuale di copertura raggiunta, il numero di test case superati o falliti). In ogni caso deve essere definita una baseline (cioè uno standard di riferimento), e in relazione a essa devono essere monitorati i progressi. Mentre il Responsabile del Test si occupa della redazione e del reporting delle metriche, è l'Analista del Test che raccoglie le informazioni per ogni metrica. Ciascun caso di test completato, ogni difetto segnalato, ogni milestone raggiunta si riflette nelle metriche generali del progetto. È importante che le informazioni inserite nei vari strumenti di monitoraggio siano precise il più possibile, così che le metriche riflettano la realtà.

Metriche accurate permettono ai manager di gestire un progetto (monitoraggio) e di avviare le modifiche necessarie (controllo). Ad esempio, un elevato numero di difetti segnalato in una certa area del software può indicare la necessità di uno sforzo di test in quell’area. I requisiti e informazioni sulla copertura del rischio (tracciabilità) possono essere utilizzati per assegnare priorità al lavoro rimanente e per allocare le risorse. Le informazioni sull’analisi causale possono essere utilizzate per individuare le aree di miglioramento dei processi. Se i dati registrati sono accurati, il progetto può essere adeguatamente controllato e possono essere segnalate agli stakeholder informazioni corrette sullo stato di avanzamento. I progetti futuri possono essere pianificati in modo più efficace quando si dispone di dati raccolti dai passati progetti. C’è una miriade di usi possibile per i dati, se essi sono accurati: compito dell’Analista del Test è proprio quello di assicurarsi che i dati siano accurati, tempestivi e obiettivi.

1.4 Analisi del Testing

Durante la pianificazione dei test viene definita la finalità del progetto di test, che l'Analista del Test utilizza per:

- Analizzare la base di test
- Identificare le condizioni di test

Affinché l’Analista del Test possa procedere efficacemente con analisi del testing, devono essere soddisfatti questi criteri di ingresso:

- Esistenza di un documento che descriva l'oggetto del testing, che possa servire come “base di test”
- Questo documento deve aver superato un riesame con risultati ragionevoli e deve essere stato aggiornato, se necessario, dopo il riesame
- Disponibilità di un budget e di una schedulazione ragionevoli per realizzare il lavoro rimanente di questo oggetto del testing.

Le condizioni di test sono normalmente identificate tramite l’analisi della base di test e degli obiettivi del test. In alcune situazioni, in cui la documentazione potrebbe essere obsoleta o inesistente, le condizioni di test possono essere identificate parlando con gli stakeholder (ad esempio, in occasione di workshops o durante la pianificazione di sprint). Queste condizioni sono poi utilizzate per determinare che cosa testare, utilizzando tecniche di progettazione dei test individuate all'interno della strategia di test e / o del piano di test.

Mentre le condizioni di test sono specifiche per il singolo elemento di test, ci sono alcune considerazioni standard per l’Analista del Test.

- E‘ di solito opportuno definire le condizioni di test a diversi livelli di dettaglio. Inizialmente si individuano le condizioni di alto livello, per definire obiettivi generali per il testing, come "la funzionalità della schermata x". Successivamente, vengono identificate condizioni più dettagliate, costituenti la base di test case specifici, come ad esempio "verificare che la schermata x respinga un numero di conto con un numero di cifre inferiore di uno rispetto a quello corretto". L’utilizzo di questo tipo di approccio gerarchico per definire le condizioni di test, può contribuire a garantire una copertura sufficiente per gli oggetti di test di alto livello.
- Una volta definiti i rischi del prodotto, devono essere identificate le condizioni di test necessarie per affrontarli e tali condizioni devono successivamente essere tracciate all’indietro verso ogni elemento di rischio.

Alla conclusione dell’attività di analisi del testing, l'Analista del Test dovrebbe sapere quali test specifici debbano essere progettati per soddisfare le esigenze delle aree assegnate al progetto di test.

1.5 Progettazione del Testing

Sempre aderendo alle finalità determinate durante la pianificazione dei test, il processo di test continua con l'Analista del Test che progetta i test da implementare ed eseguire. Il processo di progettazione del testing include le seguenti attività:

- Determinare quale sia l’area più appropriata sia per i test di basso livello (concreti) sia per quelli di alto livello (logici)
- Determinare le tecniche di progettazione dei test case che forniscano la necessaria copertura del testing
- Creare i test case che esercitino le condizioni di testing individuate

In tutto il processo, dal momento dell’analisi e progettazione a quello dell’implementazione ed esecuzione, devono essere applicati i criteri di priorità individuati durante l'analisi dei rischi e la pianificazione dei test.

A seconda delle tipologie di test, uno dei criteri di ingresso della progettazione può essere la disponibilità di strumenti da utilizzare durante la progettazione stessa.

Durante la progettazione del testing è importante ricordare quanto segue:

- Alcuni elementi di test si affrontano meglio definendo solo le condizioni di test, piuttosto che addentrarsi nella definizione completa dei test script. In questo caso, dovrebbero essere definite le condizioni di test da utilizzare come guida per un testing “unscripted”.
- I criteri di pass / fail devono essere chiaramente identificati.
- I test devono essere progettati per essere comprensibili ad altri tester, non solo all'autore. Se l'autore non è la persona che esegue il test, altri tester avranno bisogno di leggere e comprendere i test specificati in precedenza, allo scopo di comprendere gli obiettivi del test e l'importanza relativa del test.
- I test devono essere comprensibili anche agli altri stakeholder, quali gli sviluppatori, che ne riesamineranno i risultati, e degli auditor, che possono dover approvare i test.
- I test dovrebbero essere progettati per coprire tutte le interazioni del software con tutti gli attori (ad esempio, gli utenti finali, altri sistemi), quindi non solo le interazioni che avvengono attraverso l'interfaccia visibile all'utente. Comunicazioni inter-processuali, esecuzioni batch e altri interrupt possono interagire con il software e possono creare dei difetti, per cui l’Analista del Test deve progettare i test per mitigare questo rischio.
- I test dovrebbero essere progettati per testare le interfacce tra i vari oggetti di test.

1.5.1 Test case concreti e logici

Uno dei compiti dell’Analista del Test è di determinare la migliore tipologia di test case per una data situazione. I test case “concreti” forniscono tutte le informazioni e le procedure necessarie al tester per eseguire il caso di test (compresi i requisiti riguardanti i dati) e per verificarne i risultati. I test case “concreti” sono utili quando i requisiti sono ben definiti, quando il personale di test è meno esperto e quando è necessaria una verifica esterna del testing, come ad esempio un processo di auditing. I test case “concreti” forniscono un’eccellente riproducibilità (cioè, un altro tester otterrà gli stessi risultati), ma possono richiedere un notevole sforzo di manutenzione e tendono a limitare l'ingegnosità del tester durante l'esecuzione.

I test case “logici” forniscono linee guida per ciò che dovrebbe essere testato, ma permettono all'Analista del Test di variare i dati effettivi o anche la procedura da seguire durante l'esecuzione del test. I test case logici possono fornire una migliore copertura dei test case concreti, perché essi variano un po’ ogni volta che sono eseguiti, il che comporta anche una perdita di riproducibilità. I test case logici sono indicati quando i requisiti non sono ben definiti, quando l'Analista del Test ha una buona dimestichezza sia con il testing sia con il prodotto, e quando non è richiesta una documentazione formale né sono previste sessioni di auditing. I test case logici possono essere specificati nelle fasi iniziali del processo di raccolta dei requisiti, anche quando tali requisiti non sono ancora ben definiti; possono poi essere utilizzati per sviluppare test case concreti, non appena i requisiti diventano più definiti e stabili. In questo caso, la creazione del caso di test è svolta in modo sequenziale, procedendo dal logico al concreto, con l’utilizzo in esecuzione dei soli test case concreti.

1.5.2 Creazione di test case

I test case sono progettati attraverso l'elaborazione graduale e l’affinamento delle condizioni di test identificate, utilizzando le tecniche di progettazione del testing (vedi capitolo 3) individuate nella strategia di test e / o nel piano di test. I test case devono essere ripetibili, verificabili e tracciabili fino alle basi del test (ad esempio, i requisiti), come dettato dalla strategia di test che viene utilizzata.

La progettazione dei test case comprende l'identificazione di:

- Obiettivo
- Pre-condizioni, quali i requisiti dell’ambiente di test (sia esso di progetto o locale), i piani per la loro consegna etc.
- Requisiti dei dati di test (sia i dati di input per il caso test, sia i dati che devono preesistere nel sistema per eseguire il caso di test)
- Risultati attesi
- Post-condizioni, quali i dati impattati, lo stato del sistema, collegamenti con processi successivi, ecc

Il livello di dettaglio dei test case, che incide sia sul loro costo di sviluppo che sul livello di ripetibilità esecutiva, dovrebbe essere definito prima della loro effettiva creazione. Un minor dettaglio nel caso di test permette una maggior flessibilità all’Analista del Test durante la sua esecuzione e gli offre l'opportunità di investigare aree potenzialmente interessanti. Un minor dettaglio, tuttavia, tende anche a creare una minor riproducibilità.

Una sfida particolare è spesso la definizione del risultato atteso da un test. Tale attività, se eseguita manualmente, è spesso noiosa e soggetta ad errori, per cui è preferibile, se possibile, utilizzare o creare un sistema automatico di “oracolo del test”. I tester, per individuare il risultato atteso, devono considerare non solo gli output a video, ma anche le post-condizioni relative a dati impattati e all’ambiente. Se la base di test è chiaramente definita, l’identificazione del risultato corretto dovrebbe, in teoria, essere semplice. Tuttavia, nella realtà, le basi di test sono spesso vaghe, contraddittorie, prive della copertura di aree chiave quando non completamente mancanti; in questi casi l’Analista del Test deve possedere, o avere accesso a, un know-how specifico nella materia da testare. Inoltre, anche dove la base di test è ben specificata, l’eventualità che ci siano interazioni complesse tra sistemi o applicazioni, o magari risposte complesse a fronte di complesse stimolazioni del sistema può rendere l’elaborazione dei risultati attesi particolarmente difficile, per cui è essenziale avere un oracolo di test. Un caso di test che venga eseguito senza alcun modo di verificare la

correttezza del risultato ha un valore aggiunto davvero basso, e un beneficio pressoché nullo; anzi, a volte fornisce un risultato spurio (come la generazione di immotivati report di errori, nella realtà non presenti) e può dar vita ad una fallace confidenza nel sistema.

Le attività sopra-descritte possono essere applicate a tutti i livelli di test, anche se la base di test potrà variare. Ad esempio, i test di accettazione utente possono essere basati principalmente sulle specifiche dei requisiti, sui casi d'uso e sui processi di business definiti, mentre i test di componenti possono basarsi principalmente su specifiche di progetto di basso livello, sulle storie degli utenti e sul codice stesso. È importante ricordare che tali attività occorrono in tutti i livelli di test, anche se l'obiettivo del test può variare. Ad esempio, il test funzionale a livello di unità è progettato per assicurare che un particolare componente fornisca la funzionalità specificata nella progettazione di dettaglio per quel componente; il collaudo funzionale a livello di integrazione deve verificare che i componenti interagiscano tra loro e forniscano le funzionalità richieste attraverso la loro interazione; a livello di sistema un obiettivo dei test dovrebbe essere la funzionalità end-to-end. Durante l'analisi e la progettazione dei test, è importante tener presente sia l'oggetto sia l’obiettivo del test, perché ciò contribuisce a determinare il livello di dettaglio richiesto, nonché gli eventuali strumenti che possono essere necessari (ad esempio, i driver e gli stub a livello di test di componente).

Durante la creazione delle condizioni e dei case test, viene in genere prodotta una certa quantità di documentazione, considerata un prodotto di lavoro del test. Nella pratica, il livello con cui sono documentati i prodotti di lavoro del test varia notevolmente in base a:

- I rischi di progetto (cosa deve o non deve essere documentato)
- Il "valore aggiunto" che la documentazione apporta al progetto
- Le norme da seguire e/o i regolamenti che devono essere soddisfatti
- Il modello del ciclo di vita utilizzato (ad esempio, un approccio Agile tende a produrre la documentazione "quanto basta")
- Il requisito di tracciabilità, dalla base di test all'analisi e progettazione dei test

A seconda dell’ambito del test, l’analisi e la progettazione dei test prendono in considerazione le caratteristiche di qualità dell'oggetto del test. Lo standard ISO25000 (che sta sostituendo l’ISO9126) fornisce un utile riferimento al riguardo. Durante il test di sistemi hardware / software, potrebbero essere considerate anche delle caratteristiche aggiuntive.

I processi di analisi e progettazione del test possono essere migliorati attraverso l’utilizzo in contemporanea di revisioni e analisi statiche. Infatti l'analisi e la progettazione del test sono spesso una forma di test statici, poiché i problemi possono essere trovati, durante lo svolgimento del processo, nei documenti di base. L'analisi e la progettazione del test basati sulle specifiche dei requisiti sono un ottimo modo per prepararsi a una riunione di revisione dei requisiti. La lettura dei requisiti, da utilizzarsi nella creazione dei test case, richiede di comprendere a fondo il requisito e di essere in grado di valutare il soddisfacimento o meno del requisito stesso. Quest’attività spesso è utile per scoprire requisiti che non sono chiari, non verificabili o per i quali non sono stati definiti i criteri di accettazione. Allo stesso modo possono essere sottoposti a revisione i prodotti di lavoro del testing, quali i test case, l’analisi dei rischi e i piani di test.

Alcuni progetti, come quelli che seguono un ciclo di vita Agile, possono avere i requisiti solo minimamente documentati. Questi sono a volte documentati in forma di "user stories", che descrivono frammenti più o meno completi di funzionalità. Una storia utente deve includere la definizione dei criteri di accettazione, per cui se si è in grado di dimostrare che il software ha soddisfatto i criteri di accettazione, questo è di solito considerato pronto per l'integrazione con le altre funzionalità già completate.

Durante la progettazione dei test, possono essere definiti i requisiti dettagliati dell’infrastruttura di test richiesta, anche se in pratica essa non può essere realizzata fino all'implementazione del testing. Occorre ricordare che l'infrastruttura di test può comprendere non solo oggetti di test e testware, ma anche ambienti, attrezzature, personale, strumenti, software, periferiche, apparecchiature per le comunicazioni, autorizzazioni utenti, e tutti gli altri elementi necessari per eseguire i test.

I criteri di uscita per l'analisi di test e la progettazione dei test variano secondo i parametri di progetto, ma tutti gli elementi descritti in queste due sezioni dovrebbero essere presi in considerazione nella loro definizione. È importante che i criteri siano misurabili e garantiscano che tutte le informazioni e le attività preparatorie necessarie per le successive fasi del testing siano state fornite.

1.6 Implementazione dei Test

L’implementazione dei test è la realizzazione della progettazione dei test. Essa perciò include la creazione dei test automatizzati, l’organizzazione (sia manuale sia automatica) dell’ordine di esecuzione, l’approntamento dei dati e degli ambienti di test, e la schedulazione esecutiva dei test, tra cui l'assegnazione delle risorse. Deve includere anche la verifica dei criteri d’ingresso (espliciti e impliciti) per il livello di testing in questione e la garanzia che i criteri di uscita dalle fasi precedenti del processo siano stati rispettati. Se invece i criteri di uscita fossero stati ignorati, per il livello di testing o anche soltanto per una sua fase, è probabile che lo sforzo d’implementazione sia soggetto a ritardi, qualità insufficiente e costi extra inattesi. È importante perciò garantire che tutti i criteri di uscita siano stati soddisfatti prima di iniziare l’implementazione.

Nel determinare l'ordine di esecuzione, ci possono essere molte considerazioni da fare. In alcuni casi, può essere utile organizzare i test in una suite (ad esempio, gruppi di test case), il che può aiutare ad organizzare il testing in modo che i test case correlati vengano eseguiti insieme. Se viene utilizzata una strategia di test basata sul rischio, l’ordine di priorità dei rischi identificati può dettare l'ordine di esecuzione dei test case.

Peraltro, ci possono essere anche altri fattori che determinano tale ordine, quali la disponibilità delle persone giuste, delle apparecchiature, dei dati e delle funzionalità da testare. Inoltre quando il codice viene rilasciato a blocchi, il testing deve essere coordinato con l'ordine con cui il software è reso disponibile per il test. In particolare, nei modelli del ciclo di vita incrementale, è importante per l'Analista del Test coordinarsi con il team di sviluppo per assicurare che il software sia rilasciato in un ordine che ne renda possibile il test. Durante l'implementazione del testing, gli Analisti di Test dovrebbero definire e confermare l'ordine con cui i test manuali e automatici debbano essere eseguiti, verificando attentamente i vincoli che potrebbero richiedere la loro esecuzione in un ordine particolare. Le dipendenze devono inoltre essere documentate e verificate.

Il livello di dettaglio e la complessità del lavoro svolto durante l'esecuzione del testing, possono essere influenzati dal contenuto dei casi e delle condizioni di test. In alcuni casi occorre seguire specifiche norme regolamentari, ed il testing deve fornire l’evidenza della conformità alle norme applicabili, quali la 12B DO-178B/ED della Federal Aviation Administration [RTCA DO-178B/ED-12B].

Come precedentemente evidenziato, i dati di test sono necessari per il testing e, in alcuni casi, l’insieme di questi dati può essere molto grande. Durante l’implementazione, gli Analisti di Test creano i dati di input ed ambientali da caricare nelle banche dati e altri archivi, nonché i dati da utilizzare con gli eventuali test automatizzati o manuali “data-driven”.

L’implementazione dei test si occupa anche degli ambienti di test; infatti durante questa fase gli ambienti dovrebbero essere completamente configurati e verificati prima di iniziare l'esecuzione. Un ambiente di test "adatto allo scopo" è essenziale: esso, cioè, dovrebbe consentire l’evidenziazione dei difetti presenti durante il testing, funzionare normalmente quando non si verificano errori, e replicare in modo adeguato, se necessario, l’ambiente di produzione, almeno per i livelli di testing più elevati. Modifiche all'ambiente di test possono essere necessarie durante l'esecuzione dei test, in funzione di eventi imprevisti, dei risultati dei test o di altre considerazioni. Se avvengono modifiche all'ambiente durante l'esecuzione, è importante valutarne l'impatto sui test che sono già stati eseguiti.

Durante l’implementazione dei test, i tester devono assicurarsi che i responsabili della creazione e della manutenzione dell'ambiente di test siano noti e disponibili, e che tutto il testware e gli strumenti di supporto al testing ed ai processi correlati siano pronti per l'uso. Ciò include la gestione della configurazione, la gestione dei difetti, e la gestione (con il “logging”) dei test. Inoltre gli Analisti di Test devono verificare le procedure che raccolgono dati per la valutazione dei criteri uscita e per il “reporting” dei risultati.

È consigliabile utilizzare un approccio equilibrato per l’implementazione, come specificato durante la pianificazione dei test. Ad esempio, strategie di testing analitiche basate sul rischio sono spesso miscelate con strategie di testing dinamiche. In questo caso, una certa percentuale di sforzo d’implementazione è assegnata ai test che non seguono script predeterminati (unscripted).

I test unscripted non devono però essere improvvisati o senza obiettivi, poiché possono diventare imprevedibili in termini di durata e di copertura (a meno che il tempo sia predefinito, cd “time-boxed”). Nel corso degli anni i tester hanno sviluppato una varietà di tecniche basate sull'esperienza, come il testing esplorativo, “error-guessing” [Myers79] e basato-sugli-attacchi. L’analisi, la progettazione e l'implementazione dei test si svolgono comunque, ma soprattutto durante l'esecuzione dei test, poiché, con queste strategie di testing dinamiche, i risultati di ogni test influenzano l'analisi, la progettazione e l'implementazione dei test successivi.
Sebbene tali strategie siano “leggere” e spesso efficaci nel trovare i difetti, esse presentano alcuni svantaggi:

- richiedono elevate competenze da parte dell'Analista del Test,
- la durata può essere difficile da prevedere,
- la copertura può essere difficile da tracciare,
- la ripetibilità, senza una buona documentazione e adeguati strumenti di supporto, può essere compromessa.

1.7 Esecuzione dei Test

L’esecuzione dei test inizia una volta che l'oggetto da testare sia consegnato, ed i criteri d’ingresso per l'esecuzione siano soddisfatti (o espressamente abbandonati). I test devono essere eseguiti secondo il piano stabilito durante l’implementazione dei test, ma l'Analista del Test deve avere il tempo sufficiente per garantire la copertura di ulteriori ed interessanti scenari e comportamenti che abbia osservato durante i test (ogni esito negativo, rilevato durante tali deviazioni dal piano, dovrebbe essere registrato, comprendendo anche le variazioni allo script di test che sono necessarie per riprodurlo). Questa integrazione di tecniche di test “scripted” e “unscripted” (per esempio, esplorative) aiuta a evitare affrettate uscite dal test dovute a lacune nella copertura dei test scripted e ad aggirare il “paradosso pesticida”.

Al centro dell'attività di esecuzione del test, c’è il confronto tra i risultati effettivi ed i risultati attesi. Gli Analisti di Test devono dedicare interesse e attenzione a questo compito; in caso contrario tutto il lavoro di progettazione e d’implementazione dei test può essere vanificato, quando errori non siano rilevati (falsi negativi) o magari il comportamento corretto sia erroneamente classificato come errato (falso positivo). La regola è che se i risultati attesi ed effettivi non corrispondono, si è verificato un incidente. Gli incidenti devono essere attentamente scrutinati per determinarne la causa (che potrebbe essere o meno un difetto nell’oggetto da testare) e per raccogliere dati utili a supportare la risoluzione dell’incidente (si veda il Capitolo 6 per ulteriori dettagli di gestione dei difetti).

Quando viene identificato un errore, la documentazione di test (specifica di test, caso di test, ecc.) deve essere attentamente valutata al fine di garantirne la correttezza. Un documento di test può essere non corretto per un certo numero di ragioni. Se è non corretto, bisogna correggerlo e, alla luce della nuova versione, rieseguire senz’altro il test. Dato che modifiche nella base e nell'oggetto del test possono rendere non corretto un caso di test anche dopo che sia stato eseguito con successo molte volte, i tester devono essere consapevoli della possibilità che i risultati osservati possano essere causati da un test non corretto.

Durante l'esecuzione, i risultati dei test devono essere registrati in modo appropriato. I test che siano stati eseguiti, ma per i quali i risultati non siano stati registrati, potrebbero dover essere ripetuti per identificare il risultato corretto, con conseguenti inefficienze e ritardi. Si noti che una registrazione corretta può risolvere i problemi di copertura e di ripetibilità associati a particolari tecniche come il testing sperimentale.
Poiché l'oggetto di test, il testware e gli ambienti di test possono essere tutti in continua evoluzione, la raccolta dei risultati dovrebbe identificare con precisione le specifiche versioni testate e le specifiche configurazioni ambientali. La registrazione dei test fornisce quindi un registro cronologico delle informazioni dettagliate relative all'esecuzione dei test.

La registrazione dei risultati si applica sia ai test individuali sia alle attività ed agli eventi intercorsi. Durante l'esecuzione, ogni test deve essere identificato in modo univoco e il suo stato va registrato, così come devono esserlo tutti gli eventi che ne influenzano l'esecuzione. Inoltre dovrebbero essere registrate anche informazioni sufficienti per misurare la copertura del testing e per documentare le ragioni dei ritardi e delle interruzioni nei test. Infine devono essere loggate tutte le ulteriori informazioni che si ritengano utili per supportare il controllo e l'avanzamento dei test, la misura dei criteri di uscita, ed il miglioramento dei processi di testing.

Le modalità di registrazione variano a seconda del livello e della strategia di testing. Per esempio, se si esegue un test automatizzato di componente, sarà lo stesso strumento di test automatico a raccogliere i dati sul test eseguito; se invece il test è eseguito in modalità manuale, sarà l’Analista del Test a raccogliere le informazioni riguardanti la loro esecuzione, spesso con l’ausilio di uno strumento di gestione che tiene traccia delle informazioni di esecuzione dei test. In alcuni casi, come per l’implementazione, la quantità d’informazioni di esecuzione dei test da registrare è influenzata da requisiti normativi o di audit.

In alcuni casi, gli utenti od i clienti possono partecipare all’esecuzione dei test. Questo può servire a costruire la loro fiducia nel sistema, anche se questo presuppone che i test trovino pochi difetti. Tale assunzione spesso non è valida nei livelli di test iniziali, ma potrebbe essere valida durante il test di accettazione.

Di seguito sono riportate alcune aree specifiche che dovrebbero essere prese in considerazione durante l'esecuzione dei test:

- Notare ed esplorare stranezze "irrilevanti" Osservazioni o risultati che possono sembrare irrilevanti, sono spesso indicatori di difetti che (come un iceberg) sono in agguato sotto la superficie visibile.
- Verificare che il prodotto non stia facendo ciò che non dovrebbe fare. La verifica che il prodotto faccia quello che dovrebbe fare è una focalizzazione normale del testing, ma l'Analista del Test deve essere anche sicuro che il prodotto non si comporti scorrettamente, facendo qualcosa che non dovrebbe fare (per esempio, altre funzioni indesiderate).
- Costruire la suite di test ed aspettarsi che cresca e si modifichi nel tempo. Il codice si evolve e dovranno essere eseguiti test aggiuntivi per coprire le nuove funzionalità, e per controllare le regressioni in altre aree del software. Inoltre durante l'esecuzione vengono spesso alla luce lacune nel processo di test. Costruire la suite di test è perciò un processo in continua evoluzione.
- Prendere appunti per le attività di test successive. Il compito del testing non termina quando il software viene fornito all'utente o distribuito sul mercato. È molto probabile che sia prodotta una nuova versione o release del software, per cui le conoscenze devono essere memorizzate e trasferite ai tester responsabili dei test successivi.
- Non aspettarsi di rieseguire tutti i test manualmente: è una prospettiva non realistica. Se si sospetta l’esistenza di un problema, l'Analista del Test dovrebbe indagarlo e segnalarlo immediatamente. Non può presumere che sarà scoperto e segnalato in una successiva esecuzione dei test case.
- Estrarre i dati dallo strumento monitoraggio dei difetti per la creazione di test case aggiuntivi. Bisogna prendere in considerazione la creazione di nuovi test case basati sui difetti scoperti durante le sessioni di test unscripted o esplorativo, e aggiungerli alle suite di regressione.

- Trovare i difetti prima del test di regressione, perché il tempo per questo genere di test è spesso limitato e di conseguenza trovare un difetto durante i test di regressione può portare a ritardi di pianificazione. I test di regressione di solito non trovano un gran quantitativo di errori, soprattutto perché sono test che sono già stati eseguiti (ad esempio, per una versione precedente del medesimo software), ed i difetti dovrebbero essere stati già rilevati nelle esecuzioni precedenti. Ciò non significa che i test di regressione debbano essere eliminati del tutto, ma solo che la loro efficacia, in termini di capacità di individuare nuovi difetti, è inferiore agli altri test.

1.8 Valutazione dei Criteri di Uscita e Reporting

Secondo il processo di test, il monitoraggio dell’avanzamento consiste nella raccolta di informazioni necessarie per soddisfare gli obblighi di “reporting”. Questo include la misurazione dei progressi compiuti verso il completamento delle attività. Nella definizione dei criteri di uscita, ci può essere una distinzione fra criteri "deve" e "dovrebbe". Ad esempio, i criteri potrebbero affermare che "non *deve* rimanere aperto nessun difetto di Priorità 1 o 2 " e che ci "*dovrebbe* essere un tasso di almeno il 95% di superamento in tutti i test case". In questo caso, un mancato rispetto del criterio "deve", impedirebbe il soddisfacimento dei criteri di uscita, mentre un tasso di superamento del 93% potrebbe consentire al progetto di procedere al livello successivo. Ovviamente i criteri di uscita devono essere chiaramente definiti, in modo che possano essere valutati obiettivamente.

L'Analista del Test è responsabile di rendere disponibili le informazioni utilizzate poi dal Responsabile del Test per valutare i progressi verso il rispetto dei criteri di uscita e di garantire che i dati siano accurati. Se, per esempio, il sistema di gestione di test fornisce i seguenti codici sullo stato di completamento del caso di test:

- Superato
- Fallito
- Superato con eccezione

allora l'Analista del Test deve avere molto chiaro il significato di ciascuno di questi stati e li deve applicare in modo coerente. "Superato con eccezione" significa che è stato trovato un difetto, ma questo non sta pregiudicando la funzionalità del sistema? Come classificare un difetto di usabilità che crea confusione all'utente? Se il tasso di superamento è un criterio di uscita di tipo "deve", il conteggio di un caso di test "fallito" piuttosto che "superato con eccezione" diventa un fattore critico. Occorre anche considerare i test case che sono contrassegnati come "falliti", ma la causa dell’esito negativo non è imputabile a un difetto (ad esempio, l'ambiente di test non è stato correttamente configurato). Se c'è qualche ambiguità sulle metriche tracciate o sui codici di stato da utilizzare, l'Analista del Test la deve chiarire con il Responsabile del Test, in modo che le informazioni possano essere monitorate in modo accurato e coerente in tutto il progetto.

Non è insolito che all'Analista del Test possa essere chiesto un rapporto sullo stato di avanzamento durante i cicli di test, e anche di contribuire al reporting redatto al termine del test. Questo può richiedere la raccolta di metriche dai sistemi di gestione del testing e dei difetti, nonché la valutazione della copertura complessiva. L'Analista del Test dovrebbe essere in grado di utilizzare gli strumenti di reporting e di fornire al Responsabile del Test le indicazioni per estrarre le informazioni necessarie.

1.9 Attività di Chiusura dei Test

Una volta che l'esecuzione dei test è considerata completata, i risultati “chiave” dell’attività di test dovrebbero essere raccolti per archiviarli o trasferirli agli stakeholders. Nel loro complesso, queste sono le attività di chiusura del testing. L'Analista del Test dovrebbe occuparsi di consegnare i prodotti del lavoro di testing a chi ne ha bisogno. Ad esempio, i difetti rilevati che si trovino in stato di differimento o accettazione devono essere comunicati a coloro che useranno il sistema o ne daranno assistenza all'uso. I casi e gli ambienti di test dovrebbero essere consegnati ai responsabili dei test di

manutenzione. Un altro prodotto di lavoro può essere l’insieme dei test di regressione (sia automatici sia manuali).

Il pacchetto d’informazioni circa l’intero lavoro eseguito in fase di test deve essere ben documentato e includere collegamenti, diritti di accesso e quanto altro necessario al suo corretto utilizzo.

L'Analista del Test deve anche aspettarsi di partecipare alle riunioni retrospettive ("lessons learned ") in cui possono venire alla luce informazioni provenienti sia dal processo di test, sia da tutto il processo di produzione del software; tali informazioni possono essere fondamentali per individuare ciò che funziona e ciò che non funziona nel processo, rafforzando così le sue parti vincenti ed eliminando (o limitando l’impatto di) quanto invece non ha funzionato. L'Analista del Test è una fonte di informazioni preziosa e ben informata per questi incontri e vi deve partecipare, se si desidera che tali incontri siano validi per il miglioramento del processo. Se solo il Responsabile del Test sarà presente alla riunione, l'Analista del Test dovrà comunicagli le informazioni pertinenti per poter così presentare un quadro preciso del progetto.

Altra attività da eseguire è quella di archiviazione dei risultati, dei log di esecuzione, dei report e degli altri documenti e prodotti; si tratta di un task che spesso tocca all’Analista del Test ed è un’importante attività di chiusura, soprattutto se si prevede che le informazioni saranno utilizzate per un progetto futuro.
Mentre il Responsabile del Test di solito determina quali informazioni debbano essere archiviate, l'Analista del Test dovrebbe pensare a quali informazioni sarebbero necessarie, se il progetto dovesse essere riavviato in un secondo momento. Il pensare a queste informazioni alla fine di un progetto può far risparmiare mesi di lavoro quando il progetto sarà riavviato in un secondo momento, magari con un differente team.

2. Test Management: Responsabilità dell’Analista del Test – 90 minuti

Termini:

rischio di prodotto, analisi dei rischi, identificazione dei rischi, livello di rischio, gestione del rischio, mitigazione del rischio, testing basato sul rischio, monitoraggio del testing, strategia di test

Obiettivi di Apprendimento per il Test Management: Responsabilità dell’Analista del Test

2.2 Monitoraggio e controllo dell’avanzamento del testing

TA-2.2.1 (K2) Spiegare i tipi di informazioni che devono essere monitorate durante il testing per consentire un adeguato monitoraggio e controllo del progetto

2.3 Test distribuito, in outsourcing e in insourcing

TA -2.3.1 (k2) Esempi di modelli di comunicazione in caso di lavoro in un ambiente di test H24.

2.4 Compiti dell’Analista del Test nel Testing basato sul rischio

TA-2.4.1 (K3) In una determinata situazione di progetto, partecipare all’identificazione dei rischi, eseguire la valutazione dei rischi e proporre appropriati controlli del rischio

2.1 Introduzione

A differenza di altre aree in cui l’Analista del Test interagisce con (e fornisce dati a) il Responsabile del Test, questo capitolo si concentra sui settori specifici del processo di testing in cui l'Analista del Test fornisce un contributo molto importante. Sarà il Responsabile del Test a rivolgersi a lui per le informazioni che gli necessitano.

2.2 Monitoraggio e Controllo Avanzamento dei Test

Sono cinque le dimensioni principali con cui viene monitorato l’avanzamento del testing:

- Rischi del prodotto (qualità)
- Difetti
- Test
- Copertura
- Fiducia

I rischi di prodotto, i difetti, i test e la copertura possono essere (e spesso lo sono) misurati e oggetto di reporting con modalità diverse durante il progetto o l’esercizio da parte dell'Analista del Test. La confidenza nel sistema, anche se misurabile attraverso indagini, è invece un dato molto soggettivo. La raccolta delle informazioni necessarie per supportare tali metriche è parte del lavoro quotidiano dell'Analista del Test. È importante ricordare che l’accuratezza di questi dati è fondamentale, in quanto i dati inesatti forniranno informazioni imprecise sulle tendenze e potranno portare a conclusioni errate. Nel peggiore dei casi, dati inesatti indurranno decisioni gestionali sbagliate e danni alla credibilità del team di test.

Quando si utilizza un approccio di testing basato sul rischio, l'Analista del Test dovrebbe monitorare:

- Quali rischi sono mitigati mediante il testing

- Quali rischi sono considerati non mitigabili

Il monitoraggio della mitigazione del rischio è spesso svolto con uno strumento che tiene traccia anche del completamento del testing (ad esempio, uno strumento di gestione dei test). Ciò richiede che i rischi individuati siano mappati verso le condizioni di test, che a loro volta sono mappate sui test cases che si ritiene contribuiscano all’attenuazione dei rischi nel caso siano stati eseguiti con successo. In questo modo, le informazioni sulla mitigazione del rischio sono aggiornate automaticamente, quando i test case vengono aggiornati, sia che si tratti di test manuale che di test automatico.

Il controllo dei difetti è fatto di solito per mezzo di uno strumento apposito. Insieme alla registrazione del difetto, sono registrate anche le informazioni circa la sua classificazione, che vengono utilizzate per produrre grafici di tendenza che indicano il progresso del testing e la qualità del software. La classificazione delle informazioni è trattata in maggiore dettaglio nel capitolo 6 “Gestione dei difetti”. Il ciclo di vita può influenzare la quantità di documentazione dei difetti ed i metodi utilizzati per la registrazione delle informazioni.

Durante l’esecuzione del testing, è opportuno registrare le informazioni circa lo status dei test case eseguiti; il che avviene solitamente tramite uno strumento di gestione dei test, ma può essere fatto, se necessario, con mezzi manuali. Le informazioni sui test case possono includere:

- Stato di creazione del caso di test (progettato, recensito)
- Stato di esecuzione del caso di test (superato, fallito, bloccato, saltato)
- Informazioni di esecuzione del caso di test (data e ora, nome del tester, dati utilizzati)
- Elementi di esecuzione del caso di test (schermate, logs ecc.)

Come per gli elementi di rischio individuati, i test case devono essere mappati verso i requisiti oggetto del testing. È importante per l’Analista del Test ricordare che se il caso di test A è mappato al requisito A (ed è il solo caso di test associato a tale requisito), quando il caso di test A viene eseguito e superato, il requisito A deve essere considerato soddisfatto. Questo può essere considerato corretto o meno: spesso più test case sono necessari per testare a fondo un requisito, ma a causa del tempo limitato, solo un sottoinsieme di questi test è in pratica creato. Per esempio, se sono necessari 20 test case per collaudare compiutamente l’attuazione di un requisito, ma ne sono stati creati ed eseguiti soltanto 10, l’informazione di copertura dei requisiti indicherà un 100%, quando in realtà è stato raggiunto solo il 50%. Un monitoraggio accurato della copertura ed un monitoraggio dello stato di revisione degli stessi requisiti possono essere utilizzati per misurare la confidenza in un prodotto.

La quantità (e il livello di dettaglio) delle informazioni da registrare dipende da diversi fattori, tra cui il modello del ciclo di vita di sviluppo del software. Ad esempio, in progetti Agile, saranno archiviate un minor numero di informazioni, per via della stretta interazione all’interno del team e del fatto che la comunicazione tra i vari membri avviene spesso in modo diretto.

2.3 Testing Distribuito, Outsourced e Insourced

In molti casi, non tutto lo sforzo di test viene profuso da un singolo team di test, composto solo da dipendenti dell’organizzazione, in una singola località e nello stesso sito del resto del team di progetto. Se l’attività si svolge in più luoghi, il testing può essere chiamato distribuito. Se si svolge in un singolo luogo si dice centralizzato. Se l’attività di testing è effettuata in uno o più luoghi differenti da persone che non sono colleghi del resto del team di progetto e che non ne condividono il luogo di lavoro, il testing viene detto “outsourced”. Se invece il testing è effettuato da persone che sono co-localizzate con il team di progetto, ma che non sono loro colleghi, si dice “insourced”.

Quando si lavora in un progetto nel quale il team di test è distribuito su più sedi o anche in più aziende, l'Analista del Test deve prestare particolare attenzione ad una comunicazione efficace e al trasferimento di informazioni. Alcune organizzazioni lavorano con un modello detto “testing h24”, in cui un team in un certo fuso orario deve consegnare il lavoro ad un team in un altro fuso orario per permettere al testing di continuare per tutto il giorno. Ciò richiede una speciale ed accurata

pianificazione da parte dell’Analista del Test che deve consegnare o ricevere il lavoro, il che è importante per attribuire le responsabilità e fondamentale per garantire la disponibilità di informazioni corrette.

Quando non sia fattibile la comunicazione verbale, deve essere instaurata quella scritta, svolta tramite e-mail, rapporti sullo stato del testing e utilizzo di strumenti di gestione dei test e di monitoraggio dei difetti. Se lo strumento di gestione del test permette l’assegnazione dei test case ai singoli, può anche essere semplicemente utilizzato come strumento di pianificazione e di trasferimento del lavoro alle persone. Un’accurata registrazione dei difetti è utile perché le informazioni su tali difetti siano girate ai colleghi per un adeguato follow-up. L’uso corretto di tali mezzi di comunicazione è vitale per un’organizzazione che non possa contare su un’interazione giornaliera tra le persone che lavorano per essa.

2.4 I compiti dell’Analista nel Testing basato sul rischio

2.4.1 Panoramica

Il Responsabile del Test ha spesso la responsabilità generale di creare e gestire la strategia di testing basata sul rischio, per la quale di solito chiede il coinvolgimento dell’Analista del Test per garantire che tale strategia sia implementata correttamente. In particolare l’Analista del test dovrebbe essere coinvolto sui seguenti task:

- Identificazione del rischio
- Valutazione del rischio
- Mitigazione del rischio

Queste attività vengono eseguite iterativamente per tutto il ciclo di vita del progetto per far fronte ai rischi emergenti, ai cambiamenti di priorità e per valutare regolarmente e comunicare lo stato di rischio rilevato.

Gli Analisti di Test dovrebbero operare nell’ambito del contesto di testing basato sul rischio stabilito dal Responsabile del Test per il progetto. Possono contribuire fattivamente con il loro patrimonio di conoscenze alla determinazione del grado di rischio sul dominio di business legato al progetto stesso (come rischi legati alla sicurezza, al business e agli aspetti economici e politici).

2.4.2 Identificazione del rischio

Il processo d’identificazione del rischio assume tanto maggiore efficacia (ed è in grado di individuare un numero tanto maggiore di possibili rischi) quanti più sono gli stakeholder che sono chiamati a parteciparvi. Dato che gli Analisti di Test spesso hanno una conoscenza unica in relazione al particolare dominio del sistema sottoposto a test, essi sono particolarmente indicati per lo svolgimento di interviste con gli esperti di dominio e con gli utenti, per effettuare assessment indipendenti (utilizzando e facilitando l'utilizzo di modelli di rischio), per svolgere workshops di rischio, per condurre sessioni di brainstorming con gli utenti attuali e potenziali, per la definizione di liste di controllo del testing e per utilizzare l'esperienza passata in sistemi o progetti simili. In particolare è opportuno che l'Analista del Test lavori a stretto contatto con gli utenti e con gli esperti di dominio di altri per definire le potenziali aree di rischio di business raggiungibili durante il testing. Egli potrà dare un significativo apporto al processo di test, in special modo nell’identificazione dei potenziali effetti del rischio sugli utenti e sugli altri stakeholder.

Esempi di rischi che potrebbero essere individuati in un progetto sono:

- Problemi di accuratezza con le funzionalità del software: ad esempio calcoli errati
- Problemi di usabilità: ad esempio insufficienti “scorciatoie” da tastiera
- Problemi di apprendibilità: ad esempio mancanza di istruzioni per l'utente nei punti decisionali chiave

Altre considerazioni riguardanti il testing delle specifiche caratteristiche di qualità sono trattate nel capitolo 4 del presente syllabus.

2.4.3 Valutazione del rischio

Mentre l’identificazione del rischio è l’attività che cerca di individuare quanti più fattori di rischio possibile, la valutazione del rischio è lo studio di tali fattori, con particolare riguardo alla loro categorizzazione e alla determinazione della probabilità del loro verificarsi e all’impatto associato con ciascun rischio.

La determinazione del livello di rischio in genere comporta la valutazione, per ciascun elemento di rischio, della probabilità che esso si verifichi e dell’impatto legato a tale occorrenza. La probabilità di accadimento è normalmente interpretata come la probabilità che il potenziale problema possa verificarsi nel sistema oggetto del test e che possa essere poi osservato nel sistema quando questo sarà in produzione; in altre parole, essa si pone come rischio tecnico. L'Analista Tecnico di Test può contribuire a identificare e capire il potenziale rischio tecnico per ogni elemento di rischio, mentre l'Analista del Test contribuisce a comprendere il potenziale impatto di business quando il problema dovesse verificarsi.

L'impatto in caso di occorrenza è spesso inteso come la gravità degli effetti sugli utenti, sui clienti o sugli altri stakeholder; in altre parole, come rischio di business. L'Analista del Test dovrebbe identificare e valutare il potenziale impatto sul dominio di business o sugli utenti per ogni elemento di rischio.
I fattori che influenzano il rischio di business comprendono:

- Frequenza di utilizzo delle funzionalità interessate
- Dimensioni della perdita di business
- Potenziali responsabilità o perdite finanziarie, ecologiche o sociali
- Sanzioni legali civili o penali
- Preoccupazioni sulla sicurezza
- Multe, perdite di licenze
- Mancanza di ragionevoli soluzioni alternative
- Visibilità della funzione del sistema impattata
- Visibilità del difetto che induce una pubblicità negativa e potenziale danno d'immagine
- Perdita di clienti

Tenuto conto delle informazioni di rischio disponibili, l'Analista del Test deve stabilire i livelli di rischio di business in base alle linee guida stabilite dal Responsabile di Test. Tali livelli possono essere classificati in termini linguistici (con aggettivi come ad esempio, basso, medio, alto) o numerici. La classificazione non può essere esattamente quantitativa, non essendoci di solito un modo per misurare oggettivamente il rischio su una scala definita. Una misurazione accurata dei parametri probabilità/rischio e costi/conseguenze è di norma molto difficile, sicché la determinazione del livello di rischio è normalmente fatta sulla base della qualità. Può sì essere assegnata una classificazione numerica al valore qualitativo, ma ciò non la rende una vera misura quantitativa. Ad esempio, il Responsabile del Test può stabilire che i rischi aziendali vadano classificati con un valore da 1 a 10, dove 1 è il più alto (e quindi più rischioso) impatto sul business. Una volta assegnati la probabilità (valutazione del rischio tecnico) e l’impatto (valutazione del rischio di business), i due valori vengono di norma moltiplicati per determinare il punteggio complessivo di rischio per ciascun elemento di rischio. Questa valutazione complessiva è poi utilizzata per dare priorità alle attività di mitigazione del rischio. Alcuni modelli di testing basati sul rischio, come PRISMA ® [vanVeenendaal12], non combinano però i due valori di rischio, consentendo di affrontare separatamente i rischi tecnici ed i rischi di business.

2.4.4 Mitigazione del rischio

Durante lo svolgimento del progetto, l’Analista del Test dovrebbe svolgere le seguenti attività:

- Riduzione del rischio attraverso l’uso di appositi test case, che dimostrino in maniera incontrovertibile se il test è in stato “superato” o “fallito” e tramite la partecipazione a revisioni dei prodotti di lavoro software come requisiti, progetti e documentazioni utente
- Implementazione di appropriate attività di mitigazione del rischio, precisate nella strategia e nel piano di test
- Ri-valutazione dei rischi basata sulle ulteriori informazioni rilevate man mano che il progetto si svolge, correggendo le valutazioni sulla probabilità o sull’eventuale impatto o su entrambi i parametri, se necessario.
- Individuazione di nuovi rischi, a partire dalle informazioni raccolte durante i test

Se si parla di rischio di prodotto (o di qualità), allora il test è una forma di attenuazione di alcuni rischi. Nella misura in cui si scoprono difetti, i tester riducono il rischio fornendo la consapevolezza dei difetti stessi e l’opportunità di affrontarli prima del rilascio. Se i tester non scoprono alcun difetto, il test riduce il rischio assicurando che, date determinate condizioni (nella fattispecie quelle in cui si è svolto il test), il sistema funziona correttamente. Gli Analisti di Test possono determinare le opzioni necessarie alla attenuazione del rischio, attraverso la ricerca di dati accurati per il test, la creazione di scenari di test realistici e attuando (o supervisionando) studi di usabilità del sistema.

2.4.4.1 Prioritizzazione dei test

Il livello di rischio è un criterio che viene usato anche per la prioritizzazione dei test, ovvero nell’assegnazione di priorità ai test.

È compito dell’Analista del Test determinare, ad esempio, che esiste un rischio molto alto nella zona dell’accuratezza transazionale in un sistema di contabilità. Quindi, per attenuare tale rischio, al tester verrà chiesto di lavorare a contatto con gli esperti di altri domini di business, per sviluppare un cospicuo insieme di dati di test che possano essere eseguiti e verificati quanto all’accuratezza. Allo stesso modo, un Analista del Test può rilevare che siano dei problemi di usabilità a costituire fattori significativi di rischio per un nuovo prodotto: piuttosto che attendere che sia il test di accettazione utente a trovare tali problemi, l’Analista del Test farà bene ad anticipare in una fase quanto più precoce possibile i test di usabilità, fino al livello dei test di integrazione, per identificare e risolvere eventuali problemi di usabilità il prima possibile. Questa assegnazione di priorità deve essere considerata tra le prime attività delle fasi di progettazione, affinché la schedulazione possa pianificare il tempo necessario per la giusta quantità di test.

In alcuni casi tutti i test ad alto rischio vengono eseguiti prima di quelli a basso rischio, e i test sono eseguiti in un ordine rigoroso (detto “depth-first”, di profondità); in altri casi, viene utilizzato un approccio che preveda di selezionare un sottoinsieme di test percorrendo tutti i rischi identificati, usando sì il criterio del rischio per “pesare” la selezione eseguita, ma allo stesso tempo assicurando la copertura di tutti i rischi attraverso l’esecuzione di almeno un test (criterio “breadth-first”, o di ampiezza) per ciascuno.

Sia che il testing basato sul rischio proceda per profondità o per ampiezza, è comunque possibile che tutto il tempo schedulato per l’esecuzione dei test venga superato prima che siano eseguiti tutti i test ritenuti necessari. È proprio il criterio del testing basato sul rischio a consentire ai tester di riferire al Management lo stato in cui si è giunti in termini di livello di rischio rimanente; ed è lo stesso testing basato sul rischio a consentire al Management di decidere se estendere il test o trasferire il rischio su utenti, clienti, help desk, organi di supporto tecnico, personale operativo.

2.4.4.2 Adattare il testing per le sessioni successive

La valutazione dei rischi non è un’attività eseguita una tantum prima della fase di implementazione del test: è un processo continuo. Ogni ciclo di test pianificato per il futuro dovrebbe essere sottoposto a una nuova analisi del rischio, affinché tenga conto di questi fattori:

- Rischi di prodotto totalmente nuovi o variati in modo considerevole;
- Aree instabili o inclini al difetto, venute alla luce durante il test;

- Rischi derivanti da difetti corretti;
- Difetti tipici rilevabili durante il processo di test;
- Aree poco testate (low test coverage).

Se rimane tempo per ulteriori test, è buona norma espandere la copertura del rischio ad aree che presentino un livello di rischio inferiore.

3. Tecniche di Test - 825 minuti

Termini:

Analisi dei valori limite (Boundary, BVA), progettazione basata sul grafo causa-effetto, testing basato su checklist, metodo dell’albero di classificazione, testing combinatorio, testing delle tabelle delle decisioni, tecnica basata sui difetti, analisi di dominio, forzatura dell’errore, partizionamento di equivalenza, tecnica basata sull’esperienza, testing esplorativo, matrice ortogonale, testing di matrici ortogonali, testing basato sui requisiti, tecniche basate sulle specifiche, testing delle transizioni di stato, dichiarazione di test, test dei casi d’uso, test basato sull’esperienza.

Obiettivi di apprendimento per le revisioni

3.2 Tecniche basate sulle specifiche

TA-3.2.1 (K2) Spiegare l’utilizzo dei grafi causa-effetto
TA-3.2.2 (K3) Redigere test case a partire da determinate specifiche, attraverso l’applicazione delle tecniche di partizione in classi di equivalenza, per raggiungere un determinato livello di copertura
TA-3.2.3 (K3) Redigere test case a partire da determinate specifiche, attraverso l’applicazione delle tecniche di analisi dei valori limite, per raggiungere un determinato livello di copertura
TA-3.2.4 (K3) Redigere test case a partire da determinate specifiche, attraverso l’applicazione delle tecniche di progettazione delle tabelle delle decisioni, per raggiungere un determinato livello di copertura
TA-3.2.5 (K3) Redigere test case a partire da determinate specifiche, attraverso l’applicazione delle tecniche di transizioni di stato, per raggiungere un determinato livello di copertura
TA-3.2.6 (K3) Redigere test case a partire da determinate specifiche, attraverso l’applicazione delle tecniche di progettazione dei test combinatori, per raggiungere un determinato livello di copertura
TA-3.2.7 (K3) Redigere test case a partire da determinate specifiche, attraverso l’applicazione delle tecniche di progettazione su alberi di classificazione, per raggiungere un determinato livello di copertura
TA-3.2.8 (K3) Redigere test case a partire da determinate specifiche, attraverso l’applicazione delle tecniche di progettazione basata sui casi d’uso, per raggiungere un determinato livello di copertura
TA-3.2.9 (K3) Definire i criteri di accettazione per una determinata user story
TA-3.2.10(K3) Redigere test case a partire da determinate specifiche, attraverso l’applicazione delle tecniche di progettazione basata sull’analisi di dominio, per raggiungere un determinato livello di copertura
TA-3.2.11(K4) Analizzare un sistema, o le sue specifiche dei requisiti, per determinare i più comuni tipi di difetti riscontrabili e selezionare in tal modo la più appropriata tecnica basata sulle specifiche.

3.3 Tecniche basate sui difetti

TA-3.3.1 (K2) Descrivere l’applicazione delle tecniche di test basate sui difetti e differenziarne l’uso rispetto alle tecniche basate sulle specifiche.
TA-3.3.2 (K2) Spiegare l’uso delle tassonomie dei difetti e fornire esempi di come dette tassonomie possano essere applicate

3.4 Tecniche basate sull’esperienza

TA-3.4.1 (K2) Spiegare i principi delle tecniche basate sull’esperienza ed i benefici e gli inconvenienti in comparazione con le tecniche basate sulle specifiche e quelle basate sui difetti.
TA-3.4.2 (K3) Specificare i test eseguiti con la tecnica esplorativa su un determinato scenario e spiegare come possono essere riportati i loro risultati.

TA-3.4.3 (K2) Per una determinata situazione di progetto, determinare quali tecniche, basate sulle specifiche, sui difetti o sull’esperienza, possono essere utilizzare per conseguire determinati obiettivi.

3.1 Introduzione

Le tecniche di progettazione del test considerate in questo capitolo includono le seguenti categorie:

- Tecniche basate sulle specifiche (basate sul comportamento o blackbox)
- Tecniche basate sui difetti
- Tecniche basate sull’esperienza

Tali tecniche sono complementari e possono essere correttamente utilizzate per ogni attività di test, anche senza alcun riferimento al livello di test in cui ci si trova.

Si noti che tutte e tre le tecniche possono essere usate indifferentemente sia per i test di caratteristiche funzionali che non funzionali. Quest’ultimo aspetto sarà trattato nel prossimo capitolo.

Le tecniche di test discusse nel presente capitolo si concentrano principalmente sulla determinazione di dati di test ottimali (come ad esempio nelle partizioni di equivalenza) o sull’individuazione di sequenze di test (ad esempio, modelli di stato). È prassi comune combinare varie tecniche al fine di creare test case completi.

3.2 Tecniche basate sulle specifiche

Le tecniche basate sulle specifiche vengono applicate alle condizioni di test, per derivarne test case basati sull’analisi della base di test (relativa a un componente o ad un sistema), senza alcun riferimento alla sua struttura interna.

Comuni caratteristiche delle tecniche basate sulle specifiche sono le seguenti:

- Modelli formali o informali, (come ad es. diagrammi delle transizioni di stato o tavole di decisione), sono generati durante la progettazione del test, in accordo con la tecnica di test
- Le condizioni di test sono fatte derivare in modo sistematico da tali modelli.

Alcune tecniche forniscono anche dei criteri di copertura, che possono essere usati per fornire misurazioni sulla progettazione e sulle attività di esecuzione del test. Il completo soddisfacimento dei criteri di copertura non significa che tutto il set di test sia completo, ma, piuttosto, che quel modello non suggerisce ulteriori test, progettati con questa tecnica, che possano incrementare la copertura.

I test basati sulle specifiche sono spesso basati sui documenti dei requisiti del sistema. Dato che le specifiche dei requisiti ci dicono come ci si aspetta che il sistema si comporti, con particolare riguardo all’area della funzionalità, il far derivare i test dai requisiti è spesso parte del testing di comportamento di un sistema; in alcuni casi non esistono requisiti documentati, ma ve ne sono di impliciti (come nel caso in cui si stia sostituendo un sistema legacy).

Esiste una molteplicità di tecniche di test basate sulle specifiche, aventi per oggetto differenti tipi di software o diversi scenari. Le sezioni di seguito riportate mostrano i campi di applicabilità delle varie tecniche, le limitazioni e le difficoltà che può sperimentare l’Analista del Test, nonché le metodologie attraverso le quali la copertura di test può essere misurata e i vari tipi di difetti possono essere rilevati.

3.2.1 Partizioni di equivalenza

La tecnica per partizioni di equivalenza (EP) si usa per ridurre il numero di test case da utilizzare per verificare la gestione dei valori di input, di output, dei valori interni e relativi tempi di risposta. Il partizionamento viene utilizzato per creare delle classi, o partizioni, di equivalenza, composte di set di valori che il sistema processa nello stesso modo. Si presume che, utilizzando uno solo dei valori rappresentativi di una partizione, si garantisce la copertura per tutti i valori contenuti nella partizione stessa.

Applicabilità

Questa tecnica è applicabile a qualsiasi livello di test ed è da considerarsi appropriata quando esiste un set di valori da testare che possono essere processati nel medesimo modo e quando esistono dei set di valori utilizzati dall’applicazione che non interagiscono tra loro. Tale selezione è applicabile sia per partizioni valide sia invalide (cioè partizioni contenenti valori che verranno considerati invalidi dal software sottoposto a test) ed è particolarmente vincente quando è usata in combinazione con la tecnica di analisi dei valori limite, che espande i valori fino ad includere le parti estreme delle partizioni. Questa tecnica è comunemente usata per gli smoke-test di nuove build o nuove release, perché è in grado di testare le funzioni di base in modo molto veloce.

Limitazioni e difficoltà

Se l’assunto non è corretto e i valori all’interno della partizione non sono trattati esattamente nello stesso modo, questa tecnica può fallire e non scoprire alcuni difetti, così come è anche molto importante selezionare accuratamente le partizioni; ad esempio, un campo di input che accetti numeri negativi e positivi può essere testato con due partizioni, che contengano una i numeri positivi, l’altra quelli negativi, a causa della probabilità di un loro diverso comportamento. Se è ammesso anche il valore zero, esso può diventare una nuova partizione. È importante che l’Analista Test comprenda i fondamenti elaborativi per determinare il miglior partizionamento di valori possibile.

Copertura

La copertura è determinata dal numero delle partizioni per le quali un determinato valore è stato testato, diviso per il numero delle partizioni identificate. Da notare che l’utilizzo di più valori per una singola partizione NON aumenta la percentuale di copertura.

Tipi di difetto

Questa tecnica identifica difetti di tipo funzionale nella gestione dei vari valori di dati.

3.2.2 Analisi dei valori limite o di Boundary (BVA)

Si tratta di una tecnica che viene usata per verificare i valori che si trovano ai limiti delle partizioni di equivalenza. Ci sono due modi di approcciare la BVA: il test dei due valori e quello dei tre valori.

Con il test dei due valori, vengono utilizzati il valore limite e quello che si trova subito al di fuori del limite, secondo il minimo incremento possibile previsto. Ad esempio, se la partizione include i valori da 1 a 10 con incremento possibile di 0,5, il test dei due valori per il limite superiore utilizzerà 10 e 10,5, quello per il limite inferiore 1 e 0,5. I limiti sono definiti in base ai valori minimi e massimi definiti nella partizione di equivalenza.

Con il test dei tre valori, invece, per utilizzare lo stesso esempio sopra riportato, i valori utilizzati per il limite superiore sarebbero 9,5 10 e 10,5; quelli per il limite inferiore 0,5 1 e 1,5. La decisione se usare il test dei due o dei tre valori può essere determinata dal rischio che si associa all’oggetto del test; con il test dei tre valori si approcceranno gli oggetti soggetti a più alto rischio.

Applicabilità

La tecnica è applicabile in qualsiasi livello di test ed è considerata appropriata quando esiste una partizione in classi di equivalenza ordinata; l’ordinamento è necessario per il concetto di essere

“sul” e “oltre” il limite. Ad esempio, un intervallo di numeri è una partizione ordinata. Una partizione che si compone di tutti gli oggetti rettangolari non è una partizione ordinata e non ha valori limite.

In aggiunta ad intervalli di numeri, l’analisi dei valori limite può essere applicata alle seguenti categorie:

- Attributi numerici di variabili non numeriche (ad esempio, la lunghezza)
- Cicli, quando inclusi in casi d’uso
- Strutture di dati archiviati
- Oggetti fisici (compresa la memoria)
- Attività dipendenti dal tempo

Limitazioni/Difficoltà

Dato che l’accuratezza di questa tecnica dipende dall’identificazione accurata delle classi di equivalenza, essa è soggetta alle medesime limitazioni e difficoltà. L’Analista del Test deve porre attenzione anche agli incrementi previsti per i valori validi e invalidi, per determinare con cura i valori da testare.

Possono essere usate, per la BVA, soltanto partizioni di valori ordinate. Ad esempio, quando si testano i numeri di telefono cellulare che è possibile inserire in un foglio elettronico, esiste una partizione che contiene i numeri che è possibile inserire fino al numero massimo (limite) e un’altra partizione che inizia con il primo numero di cellulare successivo (oltre il limite).

Copertura

La copertura, su base percentuale, è determinata dal numero delle condizioni limite testate, diviso per il numero delle condizioni limite individuate (sia che si usi il metodo dei due valori, sia che si usi quello dei tre valori). Quest’operazione fornisce la percentuale di copertura del test dei valori limite.

Tipi di Difetto

La tecnica di test dei valori limite è particolarmente affidabile nella rilevazione dei difetti riguardanti la gestione dei limiti (scostamento, omissione) e può rivelare altri casi di valori oltre il limite. Questa tecnica riesce a scoprire i difetti funzionali riguardanti la gestione dei valori limite, con particolare riguardo agli errori nelle logiche “minore/maggiore di” (scostamento). Può essere utilizzata anche per trovare difetti non funzionali, come la tolleranza di carichi limite (ad esempio, il sistema supporta sino a 10000 utenti concorrenti).

3.2.3 Tabelle delle decisioni

Le tabelle delle decisioni vengono utilizzare per testare le interazioni tra combinazioni di più condizioni. Una tabella delle decisioni fornisce un metodo utile per verificare il test di tutte le possibili combinazioni di condizioni, e per verificare che tutte le possibili condizioni siano gestite dal software sottoposto a test. Il risultato che il test basato sulle tabelle decisionali intende raggiungere è quello di assicurarsi che tutte le combinazioni di condizioni, relazioni e vincoli siano state sottoposte a test.

Tuttavia, tentare di testare ogni possibile combinazione può dare vita a tabelle di decisioni molto estese. Un metodo intelligente per ridurre il numero di combinazioni, da tutte quelle possibili alle sole ritenute “interessanti”, viene definito metodo della tabella decisionale “ridotto”. Con l’utilizzo di questo metodo, le combinazioni vengono limitate a quelle che daranno un risultato differente, eliminando quei set di condizioni che non si ritengono rilevanti ai fini del risultato del test. I test non realistici o ridondanti, o ancora quelli in cui la combinazione tra condizioni non è possibile, vengono anch’essi eliminati dalla tabella. L’utilizzo del metodo “ridotto” al posto di quello “esteso” è una decisione che normalmente viene presa sulla base della valutazione del rischio.[Copeland03].

Applicabilità

Questa tecnica è normalmente applicata per i livelli di test di integrazione, di sistema e di accettazione, ma, in funzione della tipologia di codice, può essere applicata anche al test di componente, quando questo è alla base di un insieme di logiche decisionali.

È una tecnica particolarmente utile quando i requisiti sono presentati in forma di diagrammi di flusso o in tabelle di regole di business.

Le tabelle decisionali sono anche una tecnica di definizione dei requisiti; che, in alcuni casi, possono essere espressi proprio in questo modo; ma quand’anche non lo siano, e siano espressi in forma testuale o narrativa, è in tale forma che le combinazioni di decisioni devono essere analizzate.

Al momento della progettazione delle tabelle decisionali, è importante considerare le combinazioni di condizioni, anche se esse non sono state espresse, ma si prevede comunque che possano esistere. Solo quando saranno state considerate tutte le condizioni, la tecnica delle tabelle decisionali può essere efficacemente utilizzata per la progettazione del test.

Limitazioni/Difficoltà

Trovare tutte le condizioni che possono interagire, può essere particolarmente difficile soprattutto quando i requisiti non sono ben definiti o non esistono affatto. Non è infatti inusuale predisporre una serie di condizioni e accorgersi che i risultati attesi sono sconosciuti.

Copertura

Il minimo della copertura per la tecnica delle tabelle delle decisioni è avere almeno un caso di test per ogni colonna. Ciò presume che non vi siano condizioni composte e che ogni possibile condizione sia stata esaminata e riportata in una colonna. Nella determinazione dei test, è importante ricordare che dovrebbero essere testate anche le condizioni limite. Quest’ultima eventualità può comportare un aumento del numero dei test case necessari per l’adeguata copertura della tabella delle decisioni. L’analisi dei valori limite e la tecnica delle partizioni di equivalenza sono complementari alla tecnica della tabella delle decisioni.

Tipi di Difetto

I difetti tipicamente rilevabili con questa tecnica includono elaborazioni non corrette, basate su particolari combinazioni di condizioni, che generano un risultato inatteso. Durante la creazione delle tabelle delle decisioni, possono essere trovati difetti finanche nei documenti delle specifiche, i più comuni dei quali sono omissioni (nessuna indicazione su cosa davvero dovrebbe accadere in una determinata situazione) o contraddizioni. Il test può anche trovare problemi riguardanti combinazioni di condizioni non gestite o non gestite correttamente.

3.2.4 Grafi causa-effetto

Applicabilità

I grafi causa-effetto hanno lo stesso ambito di applicazione delle tabelle delle decisioni e si applicano ai medesimi livelli di testing.

In particolare la progettazione basata sul grafo causa-effetto contiene:

- combinazioni di condizioni che causano un risultato (causalità)
- combinazioni di condizioni che escludono un risultato (non causalità)
- condizioni multiple che devono verificarsi per poter causare un determinato risultato (condizione “and”)
- condizioni che devono alternativamente verificarsi per poter causare un particolare risultato (condizione “or”)

Tutte queste interrelazioni possono essere più facilmente visualizzate in un grafo causa effetto.

Limitazioni/Difficoltà

La tecnica dei grafi causa-effetto richiede, rispetto alle altre tecniche, un tempo e uno sforzo maggiori per essere appresa; richiede inoltre il supporto di strumenti. Soprattutto, i grafi causa-effetto hanno una codifica particolare, che deve essere ben compresa sia da chi li crea sia da chi li utilizza.

Copertura

Per ottenere la copertura minima, va testata ogni riga delle possibili cause-effetto, incluse le combinazioni di condizioni. I grafi causa-effetto permettono anche di definire i vincoli sui dati e quelli nel flusso logico del programma.

Tipi di Difetto

I grafi causa-effetto rendono possibile la scoperta degli stessi difetti che possono essere trovati con le tabelle delle decisioni. In aggiunta, la creazione dei grafi aiuta a definire il livello di dettaglio richiesto dalla base di test, aiutando così ad aumentare la qualità e la specificità della base di test e coadiuvando il tester nell’identificazione di requisiti mancanti.

3.2.5 Test delle transizioni di Stato

Il test delle transizioni di stato è utilizzato per verificare che il software sia in grado di entrare e uscire in e da determinati stati, attraverso transizioni valide o invalide. Gli eventi fanno sì che il software transiti da uno stato all’altro e per eseguire azioni, e per ogni evento può essere specificata una o più condizioni (dette a volte “condizioni di protezione” o “protezione di transizioni”) che influenzi, o influenzino, il percorso della transizione di stato. Ad esempio, un evento di tipo login, eseguito con utenza e password corrette, porterà il sistema in uno stato differente da un evento di tipo login eseguito con una password errata.

Le transizioni di stato vengono tracciate in un diagramma che riporta le transizioni valide tra stati in formato grafico, oppure in una tabella sulla quale sono riportate sia le transizioni valide sia quelle invalide.

Applicabilità

Il test delle transizioni di stato può trovare applicazione per quei software che hanno stati ben definiti ed eventi che causano la transizione attraverso tali stati (per esempio cambiando schermata a video); può essere usato in qualsiasi livello di test.

I software integrati, di tipo web e transazionali, sono degli ottimi candidati per questo tipo di test; buoni candidati sono anche i sistemi di controllo (tipo “controllori semaforici”).

Limitazioni/Difficoltà

Determinare gli stati è spesso la parte più impegnativa nella definizione della tabella o del diagramma di stato. Se il software ha un’interfaccia utente, possono essere usate le videate per definire gli stati; quando il software è di tipo “embedded”, gli stati dipendono da situazioni che possono verificarsi anche a livello hardware.

Oltre agli stessi stati, l’unità di base del testing delle transizioni di stato è la transizione individuale, nota anche come “0-commutazioni”. Difatti, testando tutte le transizioni di stato, sarà possibile trovare alcuni difetti relativi a tali comportamenti, ma se ne troveranno di più testando una sequenza di transizioni. Le commutazioni, gli “switch”, vengono numerate per indicare quante sono le transizioni che sono state verificate. Una sequenza di due transizioni successive viene detta 1-commutazione (1-switch), una di tre successive transizioni 2-commutazioni (2-switch) e così via. Queste commutazioni a volte vengono denominate N-1commutazioni, dove N sta per il numero di transizioni attraversate (nel caso che abbiamo visto di una transizione singola, la formula sarà 1-1, che corrisponde allo 0-commutazioni). [Bath08]

Copertura

Come negli altri tipi di tecniche di test, esiste una gerarchia dei livelli di copertura (coverage). Il minimo accettabile consiste nell’aver visitato ogni stato e attraversato ogni transizione. La copertura del 100% delle transizioni (detta anche copertura 100% 0-switch o copertura 100% dei rami logici) garantisce che ogni stato è stato visitato a meno che la progettazione del sistema o il modello di transizione di stato (diagramma o tabella) siano errati. In funzione delle relazioni tra stati e transizioni, potrebbe essere necessario visitare una transizione più di una volta per poterne eseguire altre almeno una volta.

Il termine “copertura n-commutazioni” si riferisce al numero di transizioni coperte. Ad esempio, raggiungere la copertura del 100% 1-commutazione significa che ogni sequenza valida di due transizioni successive è stata testata almeno una volta. Questo test può generare esiti negativi impossibili da verificare con una copertura 0-commutazioni.

La copertura “andata e ritorno” si applica a situazioni in cui la sequenza delle transizioni può formare dei loops. Si ha una copertura del 100% andata e ritorno quando sono stati testati tutti i cicli a partire da ogni stato per tornare allo stesso stato. Il test va eseguito per ogni stato incluso nel loop.

Per ognuno di questi approcci, un livello ancora più elevato di copertura può essere raggiunto testando tutte le transizioni non valide. I requisiti e i set di copertura indicheranno se le transizioni invalide sono incluse nei test.

Tipi di difetto

I tipici difetti riscontrabili con questo tipo di test sono quelli legati alla errata elaborazione nello stato corrente che sia il risultato di una errata elaborazione in uno stato precedente, transizioni non corrette o non supportate, stati senza via di uscita e richieste di stati o transizioni che non esistono. È possibile che, durante la creazione del modello degli stati del sistema, siano rilevati difetti nei documenti delle specifiche. I più comuni sono omissioni (nessuna informazione su che cosa dovrebbe accadere in una certa situazione) e contraddizioni.

3.2.6 Testing combinatorio

Il testing combinatorio è utilizzato quando si testa il software con diversi parametri, ognuno dei quali reca diversi valori, che danno origine a più combinazioni di quelle che sia possibile testare nel tempo disponibile. Tali parametri devono essere indipendenti e compatibili, nel senso che ogni differente opzione e ogni differente fattore possono essere combinati con qualsiasi altra opzione e fattore. Le tecniche ad albero di classificazione consentono di escludere alcune combinazioni, qualora alcune delle opzioni risultino incompatibili. Ciò non significa che i fattori combinati non si influenzino a vicenda, ma che potrebbero, anzi dovrebbero, influenzarsi reciprocamente in qualche modo.

Il testing combinatorio fornisce un mezzo per identificare un idoneo sottoinsieme di queste combinazioni, in modo da poter raggiungere un livello di copertura predeterminato. Può essere compito dell’Analista del Test selezionare il numero di oggetti (includendo quelli singoli, doppi, tripli o multipli [Copeland03]) da includere nelle combinazioni; in quest’ambito, esistono vari strumenti che possono essergli di supporto (si veda www.pairwise.org). Tali strumenti prevedono che sia i parametri sia i loro valori siano elencati (testing combinatorio o a matrice ortogonale) o rappresentati graficamente (alberi di classificazione) [Grotchmann94].

Il test combinatorio di tipo “pairwise” è applicato quando ci sono coppie di variabili in combinazione.

Le matrici ortogonali sono tabelle predefinite e molto accurate dal punto di vista matematico che consentono all’Analista del Test di sostituire gli elementi da sottoporre al test con le variabili della matrice, dando vita a una serie di combinazioni che consenta un determinato livello di copertura del test [Koomen06]. Gli strumenti ad albero di classificazione aiutano l’Analista del Test a definire la dimensione delle combinazioni da testare (ad esempio combinazioni di due, tre o più valori).

Applicabilità

Il problema di avere troppe combinazioni da sottoporre a test si può verificare in almeno due occasioni durante il processo di testing. Alcuni test case contengono vari parametri, ciascuno con un numero di possibili valori, per esempio una schermata con diversi campi di input. Oltre a ciò, alcuni sistemi possono essere configurabili in diverse dimensioni, con la conseguenza di uno spazio di configurazione potenzialmente grande. In entrambe queste situazioni il test combinatorio può giovare per identificare un sottoinsieme di combinazioni, realizzabile nella dimensione voluta.

Per parametri con un alto numero di valori, si può tentare, in fase iniziale, l’uso della partizione in classi di equivalenza o di atri meccanismi selettivi, atti a ridurre il numero dei valori di ciascun parametro; poi, in un secondo momento, viene applicata la tecnica di test combinatorio per ridurre il numero risultante di combinazioni ottenute.

Queste tecniche normalmente sono utilizzate nel testing a livello di integrazione, di sistema e di integrazione di sistema.

Limitazioni/Difficoltà

La limitazione maggiore con tali tecniche risiede nell’assunzione stessa che i risultati di pochi test debbano essere estesi, quanto a rappresentatività, a tutti i test, e che tali (pochi) test rappresentino l’utilizzo atteso del sistema. Se c’è un’interazione inattesa tra alcune variabili, potrebbe non essere scoperta se quella particolare combinazione non è stata testata. E difficile sarebbe anche, in tal caso, spiegare a un uditorio non tecnico una simile evenienza, dato che potrebbe avere difficoltà a capire la logica della riduzione dei test.

Inoltre, potrebbe essere difficile, a volte, l’identificazione dei parametri e dei loro valori. Difficile è anche eseguire manualmente un lavoro di selezione di un insieme minimale di combinazioni, atto a soddisfare un determinato livello di copertura, per cui normalmente vengono utilizzati dei tool specifici. Alcuni di questi strumenti aiutano a includere o escludere alcune combinazioni o sottocombinazioni nella o dalla selezione finale di combinazioni da testare. Questa capacità può tornare utile all’Analista del Test per dare maggiore o minore evidenza a determinati fattori, basandosi sulla conoscenza del dominio o sulle informazioni legate all’uso del prodotto.

Copertura

Vi sono diversi livelli di copertura. La copertura minima è detta 1-wise o “singleton”. Richiede che ciascun valore di ogni singolo parametro sia presente in almeno una delle combinazioni selezionate. Il livello successivo si chiama 2-wise o “pairwise” (doppio), e richiede che ogni coppia di valori di due parametri sia inclusa in almeno una combinazione. Questo tipo di impostazione può essere detto copertura “n-wise”, e prevede che ogni sottocombinazione di valori di ciascun set di n parametri siano inclusi nel set di combinazioni selezionate. Più alto è questo numero “n”, più sono le combinazioni necessarie per raggiungere il 100% di copertura. La copertura minima con queste tecniche consiste nell’avere un test case per ogni combinazione prodotta dallo strumento.

Tipi di difetto

La tipologia di difetti più comune che si possa scoprire con queste tecniche è quella riguardante la combinazione dei valori dei diversi parametri.

3.2.7 Testing dei casi d’uso

Il testing dei casi d’uso consiste in test transazionali o basati sugli scenari, tali da simulare l’uso del sistema da parte degli utenti finali. I casi d'uso sono definiti in termini di interazioni tra gli attori e il sistema volte a raggiungere un determinato risultato. Gli attori possono essere utenti o sistemi esterni.

Applicabilità

Il testing dei casi d’uso è utilizzato normalmente a livello di test di accettazione, ma può essere utilizzato anche al livello di integrazione e di componente, a seconda del comportamento atteso da parte del componente. È spesso la base sulla quale sono progettati i test di performance: gli scenari, in questo caso, vengono assegnati a utenti virtuali per simulare un carico realistico sul sistema oggetto del test.

Limitazioni / Difficoltà

Per essere validi, i casi d’uso devono consistere in transazioni utente realistiche, provenienti direttamente dagli utenti, o comunque devono essere rappresentativi di una transazione utente. Il valore di un caso d’uso si riduce drasticamente quanto meno accuratamente esso riflette il

comportamento di un utente reale. Una definizione accurata dei vari percorsi e flussi è fondamentale affinché la copertura sia completa. I casi d’uso possono essere considerati delle linee guida ma non una definizione completa di ciò che va testato, dato che essi possono anche non prevedere una definizione precisa dell’intero insieme dei requisiti da testare. Può essere consigliabile creare dei documenti, come diagrammi di flusso, a partire proprio dal caso d’uso in forma narrata/testuale, proprio per assicurarsi dell’accuratezza del testing e per verificare la bontà dello stesso caso d’uso.

Copertura

La copertura minima del testing dei casi d’uso consiste nell’avere un caso di test per la transazione positiva e uno per ogni altro percorso alternativo. I percorsi alternativi devono comprendere eccezioni, test di percorsi non corretti, e a volte appaiono come semplici “estensioni” del percorso eseguito per portare a termine la transazione positiva. La percentuale di copertura si determina dividendo il numero di percorsi testati per il totale dei percorsi percorribili (principali e alternativi).

Tipi di difetti

I difetti rilevabili con questo genere di testing sono di norma gestioni errate di scenari definiti, mancanza di gestione per percorsi alternativi, gestione errata delle condizioni proposte al test, report degli errori scomodo o non corretto.

3.2.8 Testing basato sulle User Story

In alcune metodologie di sviluppo software di tipo AGILE, come Scrum, i requisiti sono preparati sotto forma di storie utente (user story) che descrivano unità di funzionamento piccole che possono essere progettate, sviluppate, testate e dimostrate nel corso di una singola iterazione. [Cohn04]. Tali user story includono una descrizione della funzionalità da implementare e includono i criteri non funzionali e anche quelli di accettazione da raggiungere perché la user story possa considerarsi conclusa con successo.

Applicabilità

Le storie utente sono utilizzate principalmente in AGILE e negli ambienti iterativi e incrementali, per il testing sia funzionale che non funzionale. Vengono usate per il test a tutti i livelli, per dimostrare il funzionamento del software da parte dello sviluppatore, prima che “passi di mano” al livello successivo di test (integrazione, performance etc.)

Limitazioni / Difficoltà

Dato che le user story sono piccole parti di funzionalità, potrebbe essere necessario come requisito la produzione di driver o stub per poterle testare. Ciò potrebbe richiedere una particolare abilità in programmazione e uso di strumenti come le API, che possano essere d’aiuto in tale compito. Il Test Analyst deve essere coinvolto nella redazione delle user story per garantirne la testabilità e, nonostante la creazione di driver e stub sia compito principalmente dello sviluppatore, un Test Analyst può essere coinvolto anche in questa attività e nell’utilizzo degli strumenti di test API. Quando viene utilizzato un modello di sviluppo che prevede l’integrazione continua, come accade in gran parte dei progetti AGILE, la necessità di driver e stub è ridotta al minimo.

Copertura

La copertura minima del testing basato sulle storie utente consiste nel verificare che ciascuno dei criteri di accettazione specificati è stato soddisfatto.

Tipi di difetti

Sono normalmente difetti funzionali, nei quali il test basato sulla user story non raggiunge la funzionalità specificata. Possono inoltre verificarsi difetti a livello dell’integrazione della nuova funzionalità con quella già esistente. Inoltre, dato che le user story sono sviluppate di solito in condizioni di isolamento, possono essere riscontrati difetti relativi a performance, interfaccia, gestione degli errori. Il Test Analyst deve essere in grado di eseguire entrambe le tipologie di test (della funzione singola e di integrazione) ogniqualvolta una nuova user story viene rilasciata al test.

3.2.9 Analisi di dominio

Si considera un “dominio” un insieme valori definito. Si dice dominio “unidimensionale” l’intervallo dei possibili valori che può assumere una variabile (“uomini di età compresa tra 24 e 66 anni”); “multidimensionale” invece se si tratta di intervalli di valori che possono assumere più variabili in interazione tra loro (“uomini di età compresa tra 24 e 66 anni e di peso compreso tra i 69 e i 90 Kg”). Ogni caso di test per un dominio multidimensionale deve includere valori per ogni variabile coinvolta nel testing.

L’analisi di dominio applicata ad un dominio monodimensionale in genere utilizza le tecniche di partizionamento in classi di equivalenza e di analisi dei valori limite. Una volta definite le partizioni, l’Analista del test seleziona i valori di ogni partizione che rappresentano un valore che è all’interno della partizione (IN), uno al di fuori (OUT), sul limite della partizione (ON) e appena fuori di esso (OFF). Determinati tali valori, ogni partizione può essere testata insieme alle sue condizioni limite. [Black07]

Quando si ha a che fare con domini multidimensionali il numero di test case generati utilizzando questa metodologia aumenta esponenzialmente in relazione al numero di variabili in gioco, mentre invece un approccio basato sulla teoria del dominio porta ad una crescita lineare. Inoltre, poiché l'approccio formale prevede l’esistenza di una teoria sui difetti (un modello degli errori, o fault model), cosa che invece non accade nel partizionamento in classi di equivalenza e nell’analisi dei valori limite, un set anche più piccolo di test, basato però su tale tecnica, sarà in grado di scoprire difetti nei domini multidimensionali in modo più efficace rispetto a set di test anche più estesi creati con una tecnica di tipo euristico. Quando si ha a che fare con domini multidimensionali, il modello di test può essere costruito come tabella di decisione (o "matrice di dominio"). L’identificazione dei valori da utilizzarsi nei test case sui domini multidimensionali è estremamente probabile che richieda un supporto di tipo computazionale.

Applicabilità

L’analisi di dominio è una combinazione tra le tabelle delle decisioni, le partizioni di equivalenza e l’analisi dei valori limite, volta a creare un set di test più ridotto possibile che però copra correttamente le aree importanti e quelle in cui è probabile un errore. Normalmente viene applicata quando il metodo delle tabelle delle decisioni risulta poco maneggevole a causa della numerosità delle variabili potenzialmente coinvolte; può essere eseguita a tutti i livelli di test, ma con maggior frequenza essa viene applicata ai livelli di integrazione e di sistema.

Limitazioni / Difficoltà

Un’analisi di dominio completa richiede una comprensione profonda del software, al fine di identificare i vari domini e le possibili interazioni tra di essi. Se un dominio rimane non identificato, il test può essere molto fallace; ma è probabile che il dominio venga comunque identificato perché le variabili OFF e OUT vi “càpitano” sopra. L’analisi di dominio è una tecnica molto efficace quando si lavora a contatto con gli sviluppatori, al fine di circoscrivere le aree di test.

Copertura

Il minimo di copertura consiste nell’avere un test per ogni valore IN, OUT, ON, e OFF di ciascun dominio identificato. Quando c’è una sovrapposizione di valori (ad esempio quando il valore OUT di un dominio è anche un valore IN di un altro dominio) non v’è necessità di duplicare i test. Per tale motivo i test realmente necessari a coprire spesso sono meno di quattro per dominio.

Tipi di difetti

Includono problemi funzionali all’interno del dominio, la gestione dei valori limite, delle interazioni tra variabili e degli errori (particolarmente per i valori che non si trovano in un dominio valido).

3.2.10 Combinazione di tecniche di testing

A volte, per creare dei test vengono combinate più tecniche. Per esempio, le condizioni identificate in una tabella delle decisioni possono essere rivedute secondo il principio delle partizioni di equivalenza, per scoprire i vari modi in cui esse possano essere soddisfatte. I test case generati dovrebbero coprire non soltanto tutte le combinazioni di condizioni, ma anche, per le condizioni che sono state partizionate, i test case aggiuntivi eventualmente necessari alla copertura delle partizioni di equivalenza. Quando seleziona la tecnica di test da seguire, l’Analista del test dovrebbe fare considerazioni in merito all’applicabilità, alle limitazioni e alle difficoltà, ai possibili risultati conseguibili in termini di copertura e rilevazione dei difetti. Potrebbe non esistere una sola “migliore tecnica” di test per ogni situazione: la combinazione di tecniche può consentire la miglior copertura, purché il tempo e i profili professionali delle risorse a disposizione, consentano che esse siano correttamente applicate.

3.3 Tecniche basate sui difetti

3.3.1 Testing eseguito con tecniche basate sui difetti

Si tratta di quelle tecniche in cui un difetto riscontrato viene posto alla base della progettazione dei test, che derivano in modo sistematico da quanto si conosce in merito al tipo di difetto.

A differenza del test basato sulle specifiche, che deriva i propri casi dalle specifiche di progetto, il test basato sui difetti deriva i test dalla tassonomia degli errori (come ad esempio gli elenchi di errori divisi per categoria), che possono essere completamente indipendenti dal software sottoposto al test.

Le tassonomie possono includere liste di tipi di difetti, cause di origine, sintomi di errore e altri dati relativi ai difetti. La tecnica descritta può utilizzare anche liste di rischi o scenari di rischio identificati, e considerarli come base per il testing.

Questa tecnica di test consente inoltre al tester di raggiungere un tipo preciso di difetto o di lavorare sistematicamente all’interno di una determinata tassonomia di difetti conosciuti o comuni di un determinato tipo. L’Analista del Test utilizza i dati derivanti dalla tassonomia per determinare l’obiettivo del test, che è quello di trovare un tipo specifico di difetto. A partire da tale informazione, procede alla creazione dei casi e delle condizioni di test che faranno venire alla luce il difetto ricercato, sempre che esso esista.

Applicabilità

Il testing basato sui difetti può essere applicato in qualsiasi livello del test, ma più frequentemente si usa nel test di sistema. Esistono tassonomie standard che vengono applicate a molti tipi di software. Questo tipo di testing non legato a un prodotto è particolarmente di aiuto alle aziende per la creazione di test specifici.

Utilizzando queste tassonomie specifiche per azienda, è possibile tracciare le metriche riguardanti i difetti non solo attraverso progetti diversi, ma addirittura attraverso organizzazioni differenti.

Limitazioni/Difficoltà

Esiste una molteplicità di tassonomie di difetti, e possono essere focalizzate su un tipo particolare di testing, come quello di usabilità. Così, è importante scegliere con cura la tassonomia, se esiste, che si ritiene applicabile al software sottoposto a test. Ad esempio, potrebbero non essere disponibili tassonomie per software innovativi. Alcune aziende hanno compilato la loro tassonomia specifica sulla base di difetti probabili o frequentemente riscontrati. Quale che sia la tassonomia usata, è fondamentale che la copertura attesa sia definita prima dello startup dei test.

Copertura

La tecnica fornisce dei criteri di copertura che si usano per determinare quando tutti i test case che si ritengono utili sono stati creati. In pratica, il criterio di copertura dei test eseguiti secondo una tecnica basata sui difetti tende a essere meno sistematico di quello dei test eseguiti su una tecnica basata sul comportamento, dato che sono definite soltanto le regole generali di copertura, mentre è ampio il margine di discrezionalità su quale sia il limite della copertura. “Copertura”, così, non significa che il set di test è completo, ma solo che i difetti considerati non suggeriscono nessun altro test utile alla loro scoperta.

Tipi di difetti

I tipi di difetto rilevati dipendono molto dalla tassonomia in uso. Quando se ne usa una relativa alla GUI, ad essa saranno relativi la maggior parte dei difetti scoperti, ma nulla impedisce che ne siano scoperti altri di altra categoria, come sottoprodotto del test specifico eseguito.

3.3.2 Tassonomie dei difetti

Le tassonomie dei difetti sono liste di tipi di difetto categorizzate. Questi elenchi possono essere molto generici ed essere utilizzati come linee guida di alto livello, oppure molto specifici e puntuali. Per esempio, una tassonomia di difetti (generica) relativa all’interfaccia utente potrebbe contenere elementi generali come funzionalità, gestione degli errori, visualizzazione grafica e prestazioni. Una tassonomia dettagliata potrebbe invece includere un elenco di tutti i possibili oggetti dell'interfaccia utente (in particolare per una interfaccia utente grafica) e potrebbe indicare l'uso improprio di questi oggetti, come ad esempio:

- Campo di tipo “testo”
  - Dati validi non accettati
  - Dati non validi accettati
  - Lunghezza del campo in input non verificata
  - Caratteri speciali non rilevati
  - Messaggi di errore poco esplicativi
  - Impossibilità per l’utente di correggere dati errati
  - Regole non rispettate
- Campo di tipo “data”
  - Date valide non accettate
  - Date non valide non rifiutate
  - Mancata verifica degli intervalli ammessi
  - Gestione erronea della precisione dei dati (ad esempio ore-minuti-secondi)
  - Impossibilità per l’utente di correggere dati errati
  - Regole non rispettate (esempio: la data fine deve essere posteriore alla data inizio)

Vi sono molte tassonomie, che vanno da quelle formali (che si trovano sul mercato), a quelle progettate per scopi specifici da aziende varie. Quelle sviluppate internamente ad un’azienda possono essere usate per raggiungere e scoprire determinati difetti comunemente presenti all’interno della stessa azienda.

Quando si crea una nuova tassonomia o se ne personalizza una esistente, è importante definire prima gli obiettivi o i risultati che ci si attendono dall’applicazione della tassonomia. Ad esempio, l'obiettivo potrebbe essere quello di identificare problemi legati all'interfaccia utente scoperti in sistemi di produzione o individuare questioni relative alla gestione dei campi di input.

Per creare una tassonomia:

1. Creare un obiettivo e definire il livello di dettaglio
2. Selezionare una tassonomia proposta da usare come base
3. Definire valori o difetti rilevati dall’esperienza passata all’interno dell’organizzazione o al di fuori di essa.

Più dettagliata è la tassonomia, più tempo ci vorrà per le attività legate al suo sviluppo e alla sua manutenzione, ma ciò si tradurrà in una maggior riproducibilità dei risultati del test. Vi è il rischio che

tassonomie dettagliate possano essere per questo motivo ridondanti, ma permettono ai team di test di suddividersi il lavoro senza incorrere in una perdita di informazioni o copertura.

Una volta che la tassonomia appropriata è stata selezionata, può essere utilizzata per la creazione di condizioni e test case. Una tassonomia basata-sui-rischi può aiutare la focalizzazione dello sforzo di test su una zona a rischio specifico. Un altro uso delle tassonomie può essere quello rivolto a particolari settori del test, come usabilità, prestazioni, ecc.
Liste di tassonomie sono disponibili in varie pubblicazioni, su IEEE e su Internet.

3.4 Tecniche basate sull’esperienza

Le tecniche di testing basate sull’esperienza utilizzano l'abilità e la capacità intuitiva dei tester, sviluppatesi con la loro esperienza su applicazioni o tecnologie simili. Sono tecniche molto efficaci quanto alla scoperta di difetti, ma non molto appropriate, quanto meno non quanto altre tecniche, per raggiungere una specifica copertura o per produrre delle procedure di test riutilizzabili. Nei casi in cui la documentazione sul sistema è scarna, il tempo riservato al test è particolarmente ristretto o ancora il team di test ha una forte esperienza sul sistema da sottoporre a test, la tecnica basata sull’esperienza può essere una buona alternativa a un approccio più strutturato. Viceversa, le tecniche basate sull’esperienza sono inappropriate laddove i sistemi richiedano una documentazione di test dettagliata, un alto livello di ripetibilità o una particolare attenzione alla precisione nella copertura del test.

Da notare che, a differenza delle altre tecniche descritte in questa sede, per i test basati sull’esperienza non vi sono criteri formali di calcolo della copertura.

3.4.1 Error Guessing

Usando la tecnica dell’error guessing (letteralmente “indovinare l’errore”), l’Analista del test mette a frutto la sua esperienza per forzare l’accadimento di potenziali errori che possano essere stati commessi durante la fase di progettazione e sviluppo del software. Nel momento in cui tali errori attesi vengono individuati, l’Analista del test può determinare quali sono i migliori metodi per scoprire i difetti che ne vengono generati. Ad esempio, se l’Analista di Test si aspetta che il software mostri un esito negativo quando si inserisce una password errata, sarà necessario progettare una serie di test che prevedano l’inserimento di valori differenti nel campo password, per verificare che davvero l’errore è stato commesso ed ha avuto come effetto un difetto che può essere visto come esito negativo al momento in cui il test viene fatto girare.

Oltre a essere usata come tecnica di test, quella dell’error guessing è una tecnica utile durante la fase di analisi del rischio, allo scopo di identificare potenziali problemi (Myers97)

Applicabilità

La tecnica dell’error guessing viene applicata principalmente durante il test di integrazione e di sistema, ma può essere usata a qualsiasi livello del testing. Spesso, viene utilizzata in aggiunta ad altre tecniche e giova ad ampliare l’ambito dei test case già esistenti. Viene utilizzata anche quando si testa una nuova release di un software, per verificare gli errori più comuni, prima di iniziare la fase più rigorosa di testing guidato. In tale senso, molto utili possono essere le checklist e le tassonomie, che costituiscono una guida all’esecuzione del test.

Limitazioni / Difficoltà

Con questa tecnica, è difficile assicurare una copertura, in quanto questa varia in modo considerevole a seconda dell’esperienza dell’Analista del test. Ben si può capire come questa tecnica sia migliore quando utilizzata da un tester di comprovata esperienza, che abbia familiarità con i tipi di difetto comunemente presenti nel tipo di codice sottoposto a test. La tecnica dell’error guessing è usata con frequenza, ma altrettanto poco frequente è la sua documentazione e così è meno riproducibile di altre forme di test.

Copertura

Quando si usa una tassonomia, la copertura può essere determinata dai dati appropriati relativi agli errori e ai tipi di difetto. Senza tassonomia, la copertura è limitata in base all’esperienza e alla conoscenza del tester, oltre dal tempo disponibile. La resa di questa tecnica varia a seconda di quanto accuratamente il tester riesce a raggiungere le aree problematiche.

Tipi di Difetto

Difetti tipici sono quelli definiti nella particolare tassonomia, o “forzati” dall’Analista del test; in definitiva quelli che non sono stati trovati con il test basato sulle specifiche.

3.4.2 Testing basato sulle liste di controlli

Quando applica la tecnica basata sulle liste di controlli, l’Analista del Test esperto fa uso di una lista di alto livello che comprende oggetti da notare, controllare, ricordare, o una serie di regole e criteri in base a cui un prodotto deve essere verificato. Queste liste di controlli sono generate sulla base di standard, esperienze pregresse e altre considerazioni varie. Se ad esempio per fare un test su un’applicazione si usa una lista di controlli relativa all’interfaccia utente, siamo in presenza di un test basato su liste di controlli.

Applicabilità

Il test basato su liste di controlli viene utilizzato negli ambienti in cui vi sia un team di test dotato di una certa esperienza e familiarità con il software sottoposto a test o con l’area cui la lista di controlli si riferisce (ad esempio, per applicare con successo una lista relativa ai controlli su un’interfaccia utente, l’Analista del test dovrà essere sufficientemente esperto con le interfacce, ma non necessariamente con il software oggetto del test). Dato che le liste di controlli sono di alto livello e tendono a sottintendere i passi dettagliati che normalmente si trovano nei test case e nelle procedure di test, l’esperienza conoscitiva del tester è fondamentale per supplire a tali mancanze. Proprio a causa della non presenza di step dettagliati, le liste di controlli non necessitano di grande manutenzione e possono essere adattate a molti rilasci simili a quello in cui vengono usate. Esse possono essere utilizzate in uno qualsiasi dei livelli di test, compresi il regression testing e lo smoke test.

Limitazioni/Difficoltà

Proprio la natura delle liste di controlli e il loro livello “alto” possono nuocere alla riproducibilità dei risultati del test. È possibile che tester diversi usino diversi approcci per adempiere alla medesima lista di controlli ed è possibile che ottengano risultati differenti anche quando usano la stessa lista. Questo può dare origine a una maggiore copertura, ma spesso sacrifica la riproducibilità. Inoltre, l’uso di liste di controlli può generare una errata confidenza, superiore a quella realmente dovuta, dato che il livello di copertura raggiunto viene giudicato direttamente dal tester. Le liste di controlli sono spesso fatte derivare da test case più dettagliati o da altre liste di test e tendono a crescere con il tempo. La manutenzione è necessaria per garantire che continuino a coprire gli aspetti importanti del software sottoposto a test.

Copertura

La copertura varia a seconda della lista di controllo, ma soprattutto a seconda dell’Analista del Test che la esegue.

Tipi di Difetto

I difetti tipici che questa tecnica scopre includono gli errori risultanti dalla variazione dei dati, delle sequenze dei vari step di esecuzione, o del flusso di lavoro durante il test. Usare le checklist può essere utile per mantenere aggiornato il test quando si rendono disponibili nuove combinazioni di dati durante l’esecuzione del test.

3.4.3 Testing esplorativo

Durante il testing esplorativo, il tester è impegnato simultaneamente ad apprendere notizie circa il prodotto da testare e i suoi difetti, a pianificare il lavoro da fare, a progettare ed eseguire i test, a riportarne i risultati. Il tester man mano che procede con l’esecuzione “aggiusta il tiro” dei test e prepara una documentazione di tipo “leggero”. [Whittaker09]

Applicabilità

Un buon testing esplorativo è pianificato, interattivo e creativo. Richiede una documentazione scarna sul sistema da testare e normalmente viene usato in situazioni in cui la documentazione non è disponibile o comunque è inadeguata alle altre tecniche di test. Normalmente serve per integrare gli altri test e come base per lo sviluppo di test case addizionali.

Limitazioni/Difficoltà

Il test esplorativo può presentare delle difficoltà nelle fasi di gestione e pianificazione. La copertura può essere sporadica e difficile la riproducibilità. Uno dei metodi di gestione del test esplorativo prevede l’utilizzo di documenti per definire le aree da coprire in una sessione di test e di time-boxing per determinare di quanto tempo si dispone per eseguire il test. Alla fine della sessione (o delle sessioni) di test, il Responsabile del test può convocare una riunione di debriefing per raccogliere i risultati dei test eseguiti e per programmare le successive sessioni. Questo genere di riunione è difficilmente realizzabile per team di test particolarmente numerosi o per grandi progetti.

Altra difficoltà riscontrabile nelle sessioni di test esplorativo è quella di tracciare accuratamente i test eseguiti in un sistema di gestione del test. Un metodo per ovviare a tale difficoltà consiste nel creare test case che siano in realtà sessioni esplorative: ciò consente di tracciare sia il tempo necessario per l’esecuzione dei test, sia la copertura raggiunta.

Dato che la riproducibilità, nel caso dei test esplorativi, può essere difficile, questo può causare problemi se si rende necessario rieseguire uno step per riprodurre un errore. Alcune aziende usano le potenzialità dei tool di test automation per catturare quello che fa un tester esplorativo. Questo fornisce una registrazione di tutte le attività eseguite durante la sessione esplorativa (o un’altra qualsiasi tipologia di sessione di test basato sull’esperienza). Può essere sì noioso esaminare tutti i passi e i dettagli per trovare la reale causa di un errore, ma è comunque una registrazione pedissequa di tutti gli step eseguiti.

Copertura

Possono essere creati documenti per specificare task, obiettivi e risultati da raggiungere. Le sessioni esplorative, quindi, vengono pianificate per raggiungere tali obiettivi e risultati. Il documento può anche specificare dove concentrare lo sforzo di test, che cosa è da considerarsi all’interno o all’esterno del perimetro della sessione di test, quali risorse sono necessarie per completare i test pianificati. Una sessione può essere usata, ad esempio, per la ricerca di particolari tipi di difetto e di altre aree potenzialmente foriere di problemi, che possono essere raggiunti senza la necessità di uno script di test formale.

Tipi di Difetto

I difetti tipicamente scoperti da un test esplorativo sono quelli basati su scenari non raggiunti durante i test funzionali, su problemi relativi a limitazioni funzionali e quelli relativi a flussi di lavoro. Talvolta si rilevano anche problematiche di sicurezza e di performance.

3.4.4 La tecnica migliore

Le tecniche basate sui difetti e sull’esperienza richiedono un’elevata conoscenza dei difetti e altre esperienze di test per aumentare la capacità di individuazione dei difetti; esse vanno dal “quick test”, dove non c’è un’attività formalmente pianificata da eseguire, alle sessioni pre-pianificate, fino ad arrivare alle sessioni formalmente descritte.

Sono di solito molto utili, ma di più nei seguenti casi:

- Non sono disponibili le specifiche
- La documentazione del sistema sottoposto al test è scarna
- Non c’è tempo sufficiente per progettare e creare le procedure di test.
- I tester sono molto esperti nel dominio e/o nella tecnologia del sistema da testare
- La lontananza dalla formalità del test guidato è un obiettivo per raggiungere una copertura che sia la più alta possibile
- Devono essere sottoposti ad analisi gli esiti negativi di tipo operativo

Queste tecniche sono anche molto utili quando vengono usate insieme alle tecniche basate sul comportamento, in quanto vanno spesso a riempire i vuoti nella copertura che risultano dalle consuete mancanze di tali tecniche. Come nelle tecniche basate sulle specifiche, non esiste una tecnica “perfetta” in tutte le situazioni che si possono presentare. È importante però che l’Analista del test capisca bene i pro e i contro per ogni tecnica e sia in grado di scegliere la migliore (o il migliore insieme di tecniche) per la situazione in atto, considerando il tipo di progetto in corso, la pianificazione, la possibilità di accesso alle informazioni, la professionalità dei tester e tutti gli altri fattori che possano influenzare la sua scelta.

4. Il testing della qualità del software – 120 minuti

Termini:

testing di accessibilità, testing di accuratezza, attrattività, valutazione euristica, test di interoperabilità, apprendibilità, operabilità, testing di idoneità, SUMI, comprensibilità, testing di usabilità, WAMMI

Obbiettivi di apprendimento

4.2 Caratteristiche di qualità per il testing sul Dominio di Business

TA-4.2.1 (K2) Spiegare con esempio quali tecniche di test sono appropriate per il test sulle caratteristiche di accuratezza, idoneità, interoperabilità e conformità.

TA-4.2.2 (K2) Definire, per le caratteristiche di accuratezza, idoneità e interoperabilità, i difetti tipici da controllare.

TA-4.2.3 (K2) Per le caratteristiche di accuratezza, idoneità e interoperabilità definire il momento in cui tali caratteristiche devono essere testate all'interno del ciclo di vita.

TA-4.2.4 (K3) Dato un particolare contesto di progetto, individuare quali approcci sono da preferire per la verifica e la validazione sia dell'implementazione dei requisiti di usabilità, sia il raggiungimento delle aspettative di soddisfazione da parte dell'utente.

4.1 Introduzione

Mentre i capitoli precedenti descrivono le specifiche tecniche di test a disposizione del tester, questo capitolo considera l'applicazione di tali tecniche nella valutazione dei principali attributi usati per descrivere le qualità delle applicazioni e dei sistemi.

Questo syllabus parla degli attributi di qualità che possono essere oggetto di valutazione da parte di un Analista del Test; quelli da valutarsi da parte di un Analista Tecnico di Test sono oggetto del Syllabus Advanced Technical Test Analyst. Quale guida per la descrizione delle caratteristiche di qualità del prodotto è stata adottata quella contenuta in ISO9126; possono essere usati anche altri standard, come ISO25000 (che ha sostituito la ISO9126). Le caratteristiche di qualità ISO sono distinte in caratteristiche di qualità del prodotto (o attributi), ciascuna delle quali può avere delle sotto-caratteristiche (sottoattributi). Come risulta dalla tabella di séguito riportata, ognuna di queste caratteristiche o sottocaratteristiche sono oggetto di esame nei Syllabi Analista del Test e Analista Tecnico di Test.

Caratteristica	Sotto-caratteristica	Analista del Test	Analista Tecnico di Test
Funzionalità	Idoneità, accuratezza, interoperabilità, conformità	X	
	Sicurezza		X
Affidabilità	Maturità (robustezza), resistenza agli errori, recuperabilità, conformità		X
Usabilità	Comprensibilità, apprendibilità, operabilità, attrattività, conformità	X	

Efficienza	Performance (rispetto dei tempi di esecuzione), corretto utilizzo delle risorse, conformità		X
Manutenibilità	Analizzabilità, variabilità, stabilità, testabilità, conformità		X
Portabilità	Adattabilità, installabilità, coesistenza con altri software, sostituibilità, conformità		X

L'Analista del Test dovrebbe concentrarsi sulle caratteristiche di funzionalità e usabilità; in seconda battuta, potrebbe essere coinvolto nel test di accessibilità, dato che pur non essendo riportata come una sottocaratteristica dell'usabilità, essa ne è normalmente considerata una parte.

I test delle altre caratteristiche di qualità sono normalmente responsabilità dell'Analista Tecnico di Test, o almeno questa è l'impostazione che viene seguita da ISTQB nei suoi Syllabi, anche se nelle diverse aziende l'organizzazione del lavoro può variare.

La sottocaratteristica della conformità è presente, come si può notare, in ognuna delle caratteristiche di qualità. Nel caso di alcuni ambienti critici dal punto di vista della sicurezza o estremamente formali, ogni caratteristica qualitativa del software può dover fare i conti con specifici standard e regolamenti (ad esempio la conformità funzionale può indicare che la funzionalità deve esser conforme a uno standard specifico, come può essere l'uso di un particolare protocollo di comunicazione, perché si possano scambiare dati con un particolare chip); e, dato che tali standard possono variare anche di molto in relazione all'azienda in cui si opera, non potranno essere presi in esame in questo contesto. Tuttavia, se un Analista del Test sta lavorando in un ambiente in cui sono presenti dei requisiti di conformità, è fondamentale che comprenda a fondo tali requisiti e assicuri che sia il test sia la documentazione ad esso relativa vi corrispondano.

Per tutti gli attributi e i sottoattributi di qualità di cui trattiamo in questa sezione, i rischi tipici devono essere ben evidenziati, in modo da poter definire e documentare l'appropriata strategia di test. Il test sulle caratteristiche di qualità richiede una particolare attenzione alla tempistica relativa al modello ciclo-di-vita del software, agli strumenti necessari, alla disponibilità del software e documentazione e all'esperienza tecnica dei tester. Se prescinde da una strategia adeguata ad affrontare ogni caratteristica e le sue precise necessità riguardo al test, il tester potrebbe non avere adeguate risorse di tempo per progettare, far partire, eseguire i test. Alcuni di questi test, come quelli di usabilità, possono richiedere l'allocazione di risorse con skill particolari, pianificazioni estese, laboratori e spazi dedicati, strumenti specifici, profili professionali estremamente specialistici e, in molti casi, un quantitativo di tempo importante. In molti casi, il test di usabilità viene condotto da un gruppo di tester separato, esperto in questo genere di testing o con esperienza utente specifica.

Il testing sulle caratteristiche e sulle sottocaratteristiche deve essere integrato lungo tutto il ciclo di test, con risorse adeguate allo sforzo. Ognuna di queste aree ha le sue specifiche necessità, presenta specifici problemi e può ricorrere in momenti differenti durante il ciclo-di-vita del software, come vedremo qui di séguito.

L'Analista del Test può non essere responsabile per le caratteristiche di qualità che richiedono un approccio particolarmente tecnico, ma è comunque importante che ne sia a conoscenza e comprenda le aree in cui i vari tipi di test si sovrappongono. Ad esempio, un prodotto che fallisce in un performance test molto probabilmente fallirà anche in un test di usabilità, se è troppo lento per l'uso che normalmente ne fa un utente. Allo stesso modo, un prodotto che presenti problemi di

interoperabilità con alcuni componenti probabilmente non è pronto per i test di portabilità, dato che i problemi basilari possono non venire alla luce a causa del cambiamento dell’ambiente.

4.2 Gli attributi di qualità per il test sui domini di business.

Il test funzionale è l’obiettivo primario dell’Analista del Test, ed è concentrato su “che cosa” il prodotto fa. La base di test per un test funzionale è generalmente un documento con requisiti o specifiche, esperienze dirette sul dominio, necessità implicite.

I test funzionali variano in base al livello di test in cui sono condotti e possono essere influenzati anche dal modello ciclo-di-vita dello sviluppo del software. Così, un test funzionale condotto durante il test di integrazione si concentrerà sulla funzionalità dei moduli che si interfacciano tra loro, relativi a una singola funzione definita. A livello di testing di sistema, il test funzionale includerà il test su tutta l’applicazione. Per i sistemi di sistemi, il test funzionale riguarderà principalmente il test end to end attraverso i sistemi integrati. In un ambiente Agile, il test funzionale è normalmente limitato alla funzionalità resasi disponibile nella particolare iterazione in cui ci si trova.

Durante il test funzionale, si impiega una grande varietà di tecniche (cfr. capitolo 3); esso può essere condotto da un tester dedicato, da un esperto del dominio, da uno sviluppatore (come spesso accade a livello di componente).

In aggiunta al test funzionale, di cui parliamo in questa sezione, ci sono due attributi di qualità che vengono considerati appartenenti ad aree di test non funzionali (“come” funziona il prodotto). Esse sono l’usabilità e l’accessibilità.

In questa sezione considereremo i seguenti attributi:

- Attributi di qualità funzionali
  - Accuratezza
  - Idoneità
  - Interoperabilità
- Attributi di qualità non funzionali
  - Usabilità
  - Accessibilità

4.2.1 Test di accuratezza

L’accuratezza funzionale include il test che valuta l’aderenza dell’applicazione ai requisiti (specificati esplicitamente o impliciti) e può includere la accuratezza computazionale. Il testing di accuratezza impiega molte delle tecniche descritte nel capitolo 3 e spesso usa la specifica o un sistema di tipo “legacy” come oracolo del test. Il test di accuratezza può essere eseguito in ogni livello del modello ciclo-di-vita e ricerca gestioni errate di dati o situazioni.

4.2.2 Testing di idoneità

Il testing di idoneità, comprende la valutazione e la validazione dell’appropriatezza di un insieme di funzioni per il proprio (o per i propri) specifici scopi. Questo test può essere basato su casi d’uso e viene eseguito normalmente durante il test di sistema, ma può anche essere eseguito nei momenti più avanzati del test di integrazione. I difetti venuti alla luce in questo tipo di test costituiscono una indicazione che il sistema non sarà in grado di soddisfare le necessità dell’utente in modo accettabile.

4.2.3 Testing di Interoperabilità

Il testing di interoperabilità verifica il grado della possibilità che hanno due o più sistemi di scambiarsi informazioni e di usarle di conseguenza. Il testing deve poter raggiungere tutti gli ambienti

possibili (comprese quindi eventuali variazioni di tipo hardware, software, middleware, sistema operativo etc) per assicurarsi che lo scambio di dati venga eseguito correttamente. Nella realtà questo può essere fattibile per un numero relativamente piccolo di combinazioni di sistemi; in tal caso, il testing di interoperabilità può esser limitato a un insieme “rappresentativo” di sistemi. Per specificare i test di interoperabilità è necessario che le combinazioni dei sistemi da sottoporre a test siano ben identificate, configurate e disponibili al test. Questi ambienti poi sono sottoposti al test usando una selezione di test case funzionali che vada a toccare i vari punti in cui vi sia uno scambio di dati.

L’interoperabilità concerne il “come” differenti software interagiscono tra loro. Un software con buone caratteristiche di interoperabilità può essere integrato con maggior facilità con altri sistemi, senza la necessità di interventi pesanti. Un metro di giudizio relativo all’interoperabilità può essere proprio il numero di modifiche e lo sforzo necessario per eseguire tali interventi.

Il testing di interoperabilità, per esempio, può concentrarsi su queste caratteristiche:

- Uso di standard di comunicazione estremamente comuni come XML
- Capacità di riconoscimento automatico di cosa è necessario per comunicare con i sistemi con cui si interfaccia e relativo aggiustamento.

Il test di interoperabilità può essere particolarmente significativo per le aziende che sviluppano software e strumenti COTS (Commercial Off The Shelf) e per quelle che sviluppano “sistemi di sistemi”.

Questa tipologia di test viene eseguita durante il test di integrazione di componente e di sistema, avente ad oggetto l’interazione del sistema col suo ambiente. Al livello di integrazione del sistema, questo tipo di test viene eseguito per determinare se il sistema, in una fase di sviluppo completata, interagisce correttamente con gli altri sistemi. Dato che i sistemi possono interagire a livelli multipli, l’Analista del Test deve comprendere queste interazioni ed essere in grado di creare le condizioni che stimoleranno le varie interazioni. Ad esempio, se due sistemi si scambiano dati, l’Analista del Test deve poter creare i dati necessari e le transazioni necessarie ad eseguire lo scambio dei dati. Fondamentale è ricordare che tutte queste interazioni possono non essere specificate all’interno della documentazione; oppure, molte di esse possono essere definite solo nei documenti architetturali e di progettazione del sistema. L’Analista del Test deve essere in grado di esaminare tali documenti per determinare i punti di interazione tra i sistemi e tra i sistemi e i loro ambienti, per essere sicuri che tutti siano stati testati.

Al testing di interoperabilità sono applicabili le tecniche di test come le tabelle delle decisioni, i diagrammi di transizione di stato, il testing dei casi d’uso e quello combinatorio.

Il tipo di difetti rilevabile con il testing di interoperabilità include essenzialmente la erronea gestione dell’interscambio di dati tra componenti che interagiscono.

4.2.4 Testing di Usabilità

È importante comprendere perché gli utenti possono incontrare difficoltà nell’uso del sistema. Per arrivare a tale comprensione, è necessario considerare che il termine “utente” comprende un’ampia tipologia di persone, che vanno da esperti in IT, a bambini, a persone diversamente abili.

Vi sono delle istituzioni nazionali (come il British Royal National Institute for the Blind) che raccomandano che le pagine web siano accessibili ai disabili, ai non vedenti, ai parzialmente vedenti, alle persone con mobilità ridotta, ai non udenti, agli utenti con capacità cognitive ridotte. Se si verifica che un’applicazione o un sito web sono accessibili a utenti che rientrano in queste particolari categorie, ciò comporta che l’accessibilità sia garantita a chiunque altro.

I test di usabilità verificano la facilità con cui gli utenti possono utilizzare o imparare a usare il sistema per raggiungere un determinato obiettivo in uno specifico contesto. Esso misura:

- Efficacia - capacità del software di consentire agli utenti di raggiungere uno specifico risultato con accuratezza e completezza in uno specifico contesto d'uso
- Efficienza - capacità del prodotto di consentire agli utenti di utilizzare un ammontare corretto di risorse in relazione all'efficacia raggiunta in uno specifico contesto d'uso
- Soddisfazione - capacità del software di soddisfare l'utente in uno specifico contesto d'uso

Gli attributi che vengono misurati includono:

- Comprensibilità - Attributi del software che concernono lo sforzo richiesto all'utente per riconoscere il concetto logico che sta alla base del software e la sua applicabilità
- Apprendibilità - Attributi del software che concernono lo sforzo dell'utente per imparare a usare l'applicazione
- Operabilità - Attributi del software che concernono lo sforzo necessario all'utente per condurre azioni effettive ed efficienti attraverso l'applicazione.
- Attrattività - La capacità del software di essere gradevole per l'utente

Il testing di usabilità viene eseguito di norma in due fasi successive:

- Testing di usabilità formativa: viene eseguito iterativamente durante le fasi di progettazione e prototipazione del software per guidare tali fasi identificando gli errori di usabilità nella progettazione.
- Testing di usabilità complessiva: viene eseguito a séguito dell'implementazione per misurare l'usabilità del prodotto e identificare i problemi nell'ambito di un componente o di un sistema completi.

I tester che verificano l'usabilità devono avere uno skill tale da includere esperienza (o almeno conoscenza) nelle aree di:

- Sociologia
- Psicologia
- Conformità agli standard nazionali (inclusi gli standard di accessibilità)
- Ergonomicità

4.2.4.1 Condurre un Test di Usabilità

La validazione della reale implementazione di un sistema andrebbe eseguita in condizioni più simili possibile a quelle nelle quali il sistema verrà usato. Ciò può comportare la necessità di allestire un laboratorio di usabilità con video camere, postazioni di lavoro simulate, utenti, ecc., così che lo staff di sviluppo possa osservare gli effetti del sistema così come è stato realizzato su persone vere. Il test di usabilità formale spesso richiede un'adeguata preparazione per gli "utenti" (che possono essere reali o simulati) cui vengono forniti script o semplicemente istruzioni da seguire. In caso di test non formale, agli utenti è consentito "sperimentare" il software, in modo che gli osservatori possano capire quanto sia facile o difficile per gli utenti capire come eseguire i loro compiti.

Molti test di usabilità vengono condotti dall'Analista del Test come parte di altri test, ad esempio durante i test funzionali a livello di sistema. Possono giovare, almeno per tenere traccia dei difetti durante tutto il modello ciclo-di-vita del software e come aiuto in fase di approccio al test, le linee guida di usabilità. Senza linee guida può risultare complicato determinare che cosa sia "inaccettabile" dal punto di vista dell'usabilità. Ad esempio: è irragionevole pensare che un utente debba cliccare dieci volte per accedere a un'applicazione? Senza linee guida specifiche, l'Analista del Test potrebbe trovarsi in difficoltà a difendere la correttezza dell'apertura di un difetto, dato che lo sviluppatore potrebbe, in un caso come questo, obiettare che il software "works as designed" e rigettare il difetto. Risulta, così, molto importante che nei requisiti siano definite le specifiche di usabilità oppure si abbiano a disposizione linee guida di usabilità applicate a progetti similari.
Le linee guida possono includere cose come l'accessibilità delle istruzioni, la chiarezza dei messaggi, i numeri di click di mouse necessari per completare una attività, i messaggi di errore, gli indicatori dei processi in corso (che avvisino l'utente che il sistema sta lavorando e non è in grado di accettare

ulteriori input in quel momento), linee guida per i layout delle videate, uso di colori e suoni di sistema che possono influenzare l'esperienza degli utenti.

4.2.4.2 Le specifiche di test di usabilità

Le tecniche principali per i test di usabilità sono:

- Ispezione, valutazione o revisione
- Interazione dinamica con i prototipi
- Esecuzione di una verifica e di una validazione dell'implementazione reale del sistema
- Controlli sulle prestazioni e questionari.

Ispezione, valutazione, revisione

L'ispezione o revisione delle specifiche e dei progetti da una prospettiva di usabilità che abbia ad oggetto il livello di coinvolgimento dell'utente può avere molta influenza sui costi, se scopre difetti in una fase precoce del progetto. La valutazione euristica (che consiste in una ispezione sistematica del progetto dell'interfaccia utente avente ad oggetto l'usabilità) può essere di aiuto per trovare problemi a livello di progetto, in modo da potervi provvedere come fossero parte di un processo iterativo. Ciò prevede che un piccolo gruppo di valutatori esaminino l'interfaccia e giudichino la sua conformità con i principi di usabilità riconosciuti (gli "euristici", appunto). Le revisioni sono più efficaci quando l'interfaccia utente è ben visibile. Ad esempio, è più facile capire e interpretare degli screen shot di esempio, piuttosto che una descrizione narrativa della funzionalità. La visualizzazione è fondamentale per una revisione adeguata della documentazione dal punto di vista dell'usabilità.

Interazione dinamica con i prototipi

Quando vengono sviluppati dei prototipi, l'Analista del Test dovrebbe avere la possibilità di utilizzarli, in modo da essere poi d'aiuto agli sviluppatori in fase di evoluzione dei prototipi stessi, inserendo nella progettazione i feedback ricevuti dagli utenti. In tal senso, i prototipi possono essere ben rifiniti, per dare agli utenti una vista più realistica possibile di come sarà il prodotto finito.

Verifica e validazione della reale implementazione del sistema

Quando i requisiti specificano le caratteristiche di usabilità per il software (come il numero di mouse click per raggiungere un determinato effetto) è possibile creare test case appositi per certificare che l'implementazione reale del sistema includa tali caratteristiche.

Per eseguire la validazione della reale implementazione, i test specificati come funzionali a livello di sistema possono essere sviluppati come scenari di test di usabilità. Questi misurano specifici attributi di usabilità, come l'apprendibilità e l'operabilità, più che risultati funzionali.

Gli scenari di test di usabilità possono essere sviluppati per verificare specificamente sintassi e semantica. Per sintassi si intende la struttura o la grammatica che sta alla base dell'interfaccia (ad esempio, cosa può essere validamente inserito in un campo); per semantica invece il significato e il risultato (ad esempio messaggi di sistema comprensibili e significativi e risultati forniti all'utente).

Vengono spesso usate le tecniche in black box (descritte nel cap. 3.2), con particolare riferimento ai casi d'uso che possono essere redatti in formato testo o UML (Unified Modeling Language).

Gli scenari per i test di usabilità devono includere istruzioni per l'utente, schedulazioni di tempo per interviste pre e post esecuzione test (per dare istruzioni e raccogliere feedback) e protocolli comuni per condurre le sessioni. Il protocollo include una descrizione di come il test deve essere portato avanti, delle tempistiche, delle cose da notare e di come tenere traccia delle sessioni, nonché dei metodi da usare per interviste e controlli.

Controlli e questionari

Le tecniche per controlli e questionari possono essere applicate per raccogliere osservazioni e feedback riguardo i risultati raggiunti dall'utente con il sistema. Esistono controlli standardizzati e liberamente disponibili come il SUMI (Software Usability Measurement Inventory) e il WAMMI (Website Analysis and MeasureMent Inventory); permettono di tenere come punto di riferimento un

database di precedenti misurazioni di usabilità. Oltre a ciò, il SUMI fornisce misure concrete di usabilità e questo può andare a costituire un set di criteri di accettazione e completamento.

4.2.5 Testing di Accessibilità

L’accessibilità è un test di fondamentale importanza soprattutto in considerazione di coloro che hanno particolari necessità o restrizioni nell’uso del sistema. Ci si riferisce, in particolare, alle disabilità. I Test di accessibilità possono considerare alcuni importanti standard, come il Web Content Accessibility Guidelines, e delle legislazioni di riferimento, come il Disability Discrimination Act (per Regno Unito e Australia), o la Section 508 (per gli Stati Uniti). L’accessibilità, come l’usabilità, deve essere oggetto di considerazione durante le fasi di progettazione. Il relativo test spesso viene eseguito durante la fase di integrazione e continua durante il test di sistema e di accettazione. Si considerano difetti i casi in cui il software non collima con i regolamenti e con gli standard definiti.

5. Revisioni - 165 minuti

Parole chiave
Nessuna

Obiettivi di apprendimento per le revisioni

5.1 Overview
TA/TTA-5.1.1 (K2) Spiegare perché la preparazione di una revisione è importante

5.2 Uso delle checklist nelle revisioni
TA-5.2.1 (K4) Analizzare un caso d’uso e identificare i problemi rilevati da una checklist redatta in base al Syllabus
TA-5.2.2 (K4) Analizzare una specifica dei requisiti o una storia utente e identificare i problemi rilevati da una checklist redatta in base al Syllabus

5.1 Introduzione

Un processo di revisione ben orchestrato prevede pianificazione, partecipazione e follow-up. L’Analista del Test deve parteciparvi attivamente, fornendo il suo particolare punto di vista e può essere prevista una fase di training per poter meglio comprendere il ruolo che può essergli assegnato in ogni fase di revisione. Tutti i partecipanti alla revisione devono essere informati dei benefici legati a una efficace esecuzione del processo di revisione, che se ben eseguita può costituire il migliore e il più conveniente contributo alla qualità del software.

A prescindere dal tipo di revisione che si deve condurre, l’Analista del Test deve comunque prevedere del tempo per prepararla; ciò dovrà includere il tempo per rivedere il lavoro prodotto, quello per controllare documenti correlati per verificare la continuità tra di essi, il tempo per determinare cosa potrebbe mancare in quanto si è prodotto. Senza un tempo adeguato di preparazione, il compito dell’Analista del Test si riduce a una mera correzione di quanto c’è già nel documento, anziché essere una partecipazione fattiva a un processo che valorizzi il lavoro del team e dia il migliore risultato possibile. Una buona revisione include capire cosa c’è scritto, determinare che cosa manca, verificare che il prodotto descritto può coesistere con altri prodotti che sono stati sviluppati o si trovano in fase di sviluppo. Ad esempio, quando si revisiona un piano di test a livello di integrazione, l’Analista del Test deve considerare anche gli oggetti che devono essere integrati: quali sono le condizioni necessarie perché essi siano pronti per l’integrazione? Ci sono dipendenze da documentare? Ci sono dati per testare i punti di integrazione?
Una revisione non è un lavoro isolato e limitato al prodotto che si sta rivedendo: deve considerare anche l’interazione di questo con gli altri che fanno parte del sistema.

Va considerato che spesso l’autore di un prodotto in fase di revisione si sente sottoposto a critica. L’Analista del Test deve assicurarsi che l’approccio alla revisione sia tale che l’autore consideri la revisione stessa un momento di lavoro comune volto a creare il miglior prodotto possibile. È chiaro che usando tale approccio i commenti dovranno essere interpretati in modo costruttivo e orientati al prodotto, non all’autore. Ad esempio, se un’istruzione risulta ambigua, è meglio dire “Non capisco che cosa dovrei testare per verificare che questo requisito sia stato correttamente implementato; puoi aiutarmi a capirlo?” che non “Questo requisito è ambiguo, nessuno sarà in grado di comprenderlo”. L’Analista del Test deve assicurarsi che l’informazione inserita nel lavoro prodotto sia sufficiente a supportare il lavoro del tester. Se l’informazione non c’è, non è chiara, non è sufficientemente dettagliata, è probabile che si sia in presenza di un difetto, che l’autore deve correggere. Se si mantiene un approccio costruttivo e non critico, i commenti saranno ricevuti con maggior favore e le riunioni saranno molto più produttive.

5.2 L’uso delle liste di controlli nelle revisioni

Le liste di controlli servono per ricordare ai partecipanti alle revisioni di verificare punti particolari durante le revisioni stesse. Sono utili anche per “spersonalizzare” la revisione (ad esempio “è la stessa checklist che usiamo in ogni revisione, non ci stiamo concentrando solo sul tuo lavoro”). Possono essere generiche e usate in tutte le revisioni, o specifiche e focalizzate su particolari caratteristiche di qualità, aree o tipi di documenti. Ad esempio, una lista di controlli generica di verifica delle proprietà generali di un documento può riguardare la presenza di identificativi univoci, la mancanza di riferimenti “to be defined” o simili, un’adeguata formattazione o altre caratteristiche di questo genere; una lista di controlli specifica per un documento di requisiti può contenere controlli sull’uso delle parole “può” e “potrebbe”, verificare la testabilità di ogni requisito e così via. La tipologia di formato dei requisiti può peraltro indirizzare il tipo di documento di controllo da adottare: un documento dei requisiti di tipo narrativo prevede senz’altro un criterio di revisione differente rispetto a uno basato sui diagrammi di flusso.

Le liste di controlli possono anche essere orientate a uno skill di programmatore/architetto o a uno di tester (più appropriata nel caso dell’Analista del Test) e possono comprendere alcuni degli oggetti che vedremo di séguito.

Le liste di controlli usate per i requisiti, per i casi d’uso e per le storie utente sono generalmente a un livello diverso rispetto a quello usato per il codice o per l’architettura del sistema.

Le liste di controlli orientate ai requisiti possono includere:

- Testabilità di ogni requisito
- Criteri di accettazione per ogni requisito
- Disponibilità di casi d’uso che richiamino la struttura, se applicabili
- Identificazione univoca di ciascun requisito – caso d’uso – storia utente.
- Versionabilità di ciascun requisito – caso d’uso – storia utente.
- Tracciabilità di ogni requisito a partire dai requisiti di business o marketing
- Tracciabilità tra requisiti e casi d’uso, se applicabile

Quanto sopra è solo a titolo esemplificativo, in quanto è fondamentale ricordare che se un requisito non è sottoponibile a test, cioè è definito in modo che L’Analista del Test non possa determinare come possa testarlo, allora siamo in presenza di un difetto verificato in quel requisito.

Un requisito che reciti “Il software dovrebbe essere molto facile da usare” non è testabile; come può L’Analista del Test determinare se il software è “facile da usare” o addirittura “molto facile da usare”? Se invece il requisito dice “Il software deve essere conforme agli standard di usabilità definiti nel documento”, e il documento esiste nella realtà, allora il requisito è testabile. Anzi, è un requisito generale, perché si applica a tutti gli oggetti dell’interfaccia e, nel caso particolare, può permettere di generare una serie di test case individuali su una banale applicazione. La tracciabilità da questo requisito o magari dal documento in cui si definiscono gli standard di usabilità ai test case può rivelare qualche criticità dato che se il requisito di usabilità di riferimento cambia, tutti i test ne vengono travolti e devono essere rivisti e modificati se necessario.

Una lista di controlli elementare per la revisione dei test case può includere:

- Lo scenario principale è definito chiaramente?
- Tutti i percorsi alternativi sono stati identificati, compresa la gestione degli errori?
- I messaggi dell’interfaccia utente sono stati correttamente definiti?
- C’è un solo percorso principale da esplorare (dovrebbe esserlo, altrimenti ci sono più casi d’uso)?
- Si può testare ogni percorso?

Una lista di controlli elementare per l’usabilità riferita a un’interfaccia utente può includere:

- È stato definito ogni campo e la funzione ad esso correlata?
- Tutti i messaggi di errore sono stati definiti?
- Tutti gli avvisi all’utente sono definiti e congruenti?
- L’ordine di tabulazione è definito?
- Ci sono le alternative su tastiera alle azioni eseguibili con il mouse?
- Sono state correttamente definite le scorciatoie (short cut tipo copia e incolla)?
- Ci sono dipendenze tra i campi (esempio: la data del campo X deve essere posteriore a quella del campo Y)?
- É stato definito il layout delle videate?
- Tale layout combacia con quello definito nei requisiti specifici?
- C’è un indicatore per l’utente che evidenzia il fatto che il sistema si trova in fase di elaborazione?
- È stato rispettato il requisito del numero minimo di mouseclick (sempre che sia stato definito)?
- Il flusso logico a video rispetta il caso d’uso?
- La videata rispetta i requisiti di apprendibilità?
- C’è una guida in linea disponibile per l’utente?
- Esiste un testo che appare al passaggio del mouse?
- In quali termini di attrattività può essere valutato (elemento soggettivo)?
- L’uso dei colori è conforme a quello utilizzato nelle altre applicazioni e agli standard aziendali?
- Gli effetti sonori sono stati usati correttamente e sono configurabili?
- La videata risponde ai criteri di localizzazione?
- L’utente riesce a capire che cosa deve fare (comprensibilità – elemento soggettivo)?
- L’utente riesce a ricordare che cosa deve fare (apprendibilità – elemento soggettivo)?

In un progetto di tipo AGILE, i requisiti prendono normalmente la forma di storia utente, che rappresentano micro aree di funzionalità passibili di dimostrazione. Mentre un caso d’uso è una transazione utente trasversale a molte aree funzionali, una storia utente è più isolata e normalmente limitata al tempo che serve per il suo sviluppo.

Una lista di controlli per una storia utente può includere:

- La storia utente è appropriata per la fase o iterazione di test in corso?
- I criteri di accettazione sono definiti e testabili?
- La funzionalità è stata chiaramente definita?
- Ci sono dipendenze con altre storie utente?
- La storia utente è nella corretta disposizione temporale?
- Contiene una funzionalità definita?

Chiaramente, se la storia utente si riferisce a una nuova interfaccia, è più corretto usare sia una lista di controlli generica di tipo storia utente, che una relativa al test sull’interfaccia utente. La via migliore perché una revisione porti a una qualità più alta del prodotto è quella di usare una combinazione di più liste di controlli. Potrà essere quindi d’aiuto per l’Analista del Test sia usare le liste di controllo generiche (come quelle descritte nel Syllabus del Foundation Level di ISTQB), sia svilupparne di specifiche come quelle sopra menzionate.

Ulteriori informazioni sono disponibili in [Gilb93] e [Weigers02].

6. Gestione dei Difetti - 120 minuti

Termini

Difetto Tassonomia dei difetti, fase di contenimento, analisi causale

Obiettivi di apprendimento per la Gestione dei Difetti

6.2 Quando un difetto può essere rilevato?

TA-6.2.1 (K2) Spiegare come la fase di contenimento sia in grado di ridurre i costi

6.3 Campi del report sui difetti

TA-6.3.1 (K2) Spiegare le informazioni che possono essere necessarie per la documentazione di un difetto non-funzionale

6.4 Classificazione dei Difetti

TA-6.4.1 (K3) Individuare, raccogliere e registrare informazioni di classificazione di un dato difetto

6.5 Analisi causale

TA-6.5.1 (K2) Spiegare lo scopo di analisi delle cause

6.1 Introduzione

Gli Analisti del Test valutano il comportamento del sistema in termini di esigenze di business e degli utenti, ad esempio, l'utente saprebbe cosa fare di fronte ad un certo messaggio o comportamento dell’applicazione. Confrontando il reale comportamento con il risultato atteso, l’Analista del Test determina se il sistema funziona correttamente. Un'anomalia (chiamato anche un incidente) è un evento inaspettato che richiede ulteriori indagini. L'anomalia può essere un guasto causato da un difetto. Un'anomalia può causare o meno la creazione di un rapporto di difetto. Un difetto è un problema reale che deve essere risolto.

6.2 Quando può un difetto essere rilevato?

Un difetto può essere rilevato attraverso un test statico ed i sintomi del difetto, l’esito negativo, possono essere rilevati attraverso test dinamici. Ogni fase del ciclo di vita di sviluppo del software dovrebbe prevedere metodi per individuare ed eliminare potenziali esiti negativi. Ad esempio, durante la fase di sviluppo, le revisioni del codice e della progettazione dovrebbero essere utilizzate per rilevare i difetti. Durante il testing dinamico, i test case sono utilizzati per rilevare esiti negativi.

Quanto prima un difetto viene individuato e corretto, tanto minore è il costo della qualità per il sistema nel suo complesso. Ad esempio, test statici possono trovare difetti prima che sia possibile effettuare test dinamici. Questo è uno dei motivi per cui il testing statico è un metodo conveniente per produrre software di alta qualità.

Il sistema di rilevamento dei difetti dovrebbe consentire all’Analista del Test di registrare la fase del ciclo di vita in cui il difetto è stato introdotto e la fase in cui è stato trovato. Se le due fasi coincidono, allora è stato conseguito un perfetto contenimento fase. Ciò significa che il difetto è stato

introdotto e trovato nella stessa fase e non ci sono "fughe" ad una fase successiva. Un esempio potrebbe essere un requisito errato che è identificato durante il riesame dei requisiti e lì viene corretto. Ciò non solo è un uso efficiente della revisione dei requisiti, ma impedisce anche che il difetto causi del lavoro in più che lo renderebbe più costoso per l'organizzazione. Se un requisito non corretto sfugge dal riesame dei requisiti e viene successivamente implementato dagli sviluppatori, testato dall’Analista del Test e rilevato dall'utente durante test di accettazione, tutto il lavoro svolto su tale requisito costituisce una perdita di tempo (per non parlare dell’utente che può aver perso la fiducia nel sistema).

Il contenimento di fase è un modo efficace per ridurre i costi dei difetti.

6.3 Campi del Rapporto del Difetto

I campi (parametri) inseriti in un rapporto del difetto sono destinati a fornire informazioni sufficienti affinché il rapporto sia gestibile. Un rapporto del difetto è gestibile se è :

- Completo – contiene tutte le informazioni necessarie in relazione
- Conciso – non contiene nessuna informazione estranea
- Preciso - le informazioni contenute sono corrette e indicano chiaramente i risultati attesi ed effettivi, nonché i passi necessari per riprodurre il difetto
- Obiettivo – è una relazione professionalmente scritta dei fatti

Le informazioni registrate in un rapporto del difetto dovrebbero essere suddivise in campi di dati. Meglio definiti sono i campi, più facile risulta rapportare i difetti individuati e produrre relazioni di sintesi e di tendenze sugli altri difetti. Quando per ogni campo sono disponibili un numero definito di opzioni, l’utilizzo di elenchi a discesa dei valori possibili può far diminuire notevolmente il tempo necessario di registrazione del difetto. Gli elenchi a discesa sono efficaci solo quando il numero di opzioni è limitato e l'utente non ha bisogno di scorrere un lungo elenco per trovare l'opzione corretta. I diversi tipi di rapporti dei difetti richiedono informazioni diverse e lo strumento di gestione dei difetti dovrebbe essere sufficientemente flessibile per predisporre i campi corretti a seconda del tipo di difetto. I dati devono essere registrati in campi distinti, possibilmente supportati da funzioni di convalida dei dati di input, al fine di evitare errori di immissione e per garantire un reporting efficace.

I rapporti dei difetti sono scritti per errori rilevati durante il testing funzionale e non-funzionale. Le informazioni contenute in un rapporto dei difetti devono identificare chiaramente lo scenario in cui è stato rilevato il problema, comprese le misure ed i dati necessari per riprodurre questo scenario, nonché fornire i risultati previsti ed effettivi. I rapporti dei difetti non-funzionali possono richiedere ulteriori informazioni relative all’ambiente, ad altri parametri di prestazione (ad esempio, le dimensioni del carico di sistema), alla sequenza dei passi di esecuzione ecc. Quando si documenta un errore di usabilità, è importante indicare ciò che l'utente si aspetta che il software faccia. Ad esempio, se lo standard di usabilità prevede che l'operazione debba essere completata con meno di quattro clic del mouse, la relazione del difetto deve indicare il numero di clic che sono stati richiesti rispetto allo standard indicato. Nei casi in cui uno standard non sia disponibile e i requisiti non prevedevano aspetti della qualità non-funzionale del software, il tester può utilizzare il testing della "persona ragionevole" per determinare se l'usabilità è inaccettabile. In questo caso, le aspettative di quella "persona ragionevole" devono essere chiaramente indicate nel rapporto del difetto. Poiché i requisiti non-funzionali sono a volte mancanti, la documentazione degli esiti negativi non-funzionali presenta più difficoltà nell’evidenziare il comportamento "atteso" rispetto al "reale".

Mentre l'obiettivo principale nella stesura di un rapporto del difetto è di ottenere una correzione per il problema, le informazioni sul difetto devono essere fornite anche ai fini di una classificazione accurata, dell'analisi dei rischi e del miglioramento dei processi.

6.4 Classificazione dei Difetti

Ci sono più livelli di classificazione che un rapporto dei difetti può ricevere durante tutto il suo ciclo di vita. La corretta classificazione dei difetti è parte integrante del reporting corretto dei difetti. Le classificazioni sono utilizzate per raggruppare i difetti, per valutare l'efficacia del testing, per valutare l'efficacia del ciclo di sviluppo e per evidenziare delle proiezioni interessanti.

Le più comuni informazioni di classificazione di nuovi difetti comprendono:

- Attività del progetto in cui il difetto viene rilevato - ad esempio, revisione, audit, ispezione, codifica, testing
- Fase del progetto in cui il difetto è stato introdotto (se nota) - ad esempio, requisiti, progettazione, progettazione di dettaglio, codifica
- Fase del progetto in cui il difetto è stato rilevato - ad esempio, definizione requisiti, progettazione, progettazione di dettaglio, codifica, revisione del codice, test unitario, test di integrazione, test di sistema, test di collaudo
- Causa presunta difetto - ad esempio, requisiti, progettazione, interfacce, codifica, dati
- Ripetibilità - ad esempio, una volta sola, intermittente, riproducibile
- Sintomo - ad esempio, caduta sistema, blocco, errore di interfaccia utente, errore di sistema, prestazioni

Una volta che il difetto è stato studiato, può essere possibile un’ulteriore classificazione:

- Causa originale - (cioè l’errore commesso che ha provocato il difetto), ad esempio, processo, errore di codifica, errori utenti, errore di testing, problema di configurazione, problemi nei dati, software di terze parti, software esterno, documentazione
- Origine - (il prodotto di lavoro in cui l'errore è stato commesso), ad esempio, requisiti, progettazione, progettazione di dettaglio, architettura, progettazione di database, documentazione utente, documentazione di test
- Tipo - ad esempio, problema di logica, problema di calcolo, problema di temporizzazione, gestione dei dati, aggiornamenti funzionali

Quando il difetto è stato corretto (o è stato rinviato o non ha ottenuto conferma), possono essere disponibili ulteriori informazioni di classificazione, come ad esempio:

- Risoluzione - ad esempio, modifica del codice, modifica documentazione, rinvio, non problema
- Azione correttiva - per esempio, revisione dei requisiti, revisione del codice, test unitario, documentazione configurazione, preparazione dei dati, non modifica apportata

In aggiunta a queste categorie di classificazione, i difetti sono spesso classificati in base alla gravità e alla priorità. Inoltre, a seconda del progetto, può essere utile una classificazione basata sull'impatto di sicurezza, sull’impatto di pianificazione del progetto, sui costi e sul rischio del progetto e sulla qualità del prodotto. Queste classificazioni possono essere prese in considerazione negli accordi (SLA o similari) relativi a quanto velocemente una correzione verrà consegnata.

L'ultimo input alla classificazione deriva dalla risoluzione finale. I difetti sono spesso raggruppati in base alla loro risoluzione, ad esempio: corretto / verificato, chiuso / non è un problema, differito, aperto / irrisolto. Questa classificazione, relativa al ciclo di vita dei difetti, è spesso utilizzata durante tutto il progetto.

I valori delle classificazioni utilizzate da un'organizzazione sono spesso personalizzati. Quanto sopra sono solo esempi di alcuni dei valori più comuni utilizzati nell'ICT. È importante che i valori di classificazione, per essere utili, siano consistentemente utilizzati. Troppi campi di classificazione rendono costosa, in termini di tempo, l'apertura e l'elaborazione di un difetto, per cui è importante valutare il valore aggiunto dei dati che vengono raccolti contro il costo incrementale per ogni elaborazione di difetto. La possibilità di personalizzare i valori di classificazione raccolti da uno strumento è spesso un fattore importante nella selezione dello stesso.

6.5 Analisi Causale

Lo scopo dell’analisi causale è di determinare la causa del difetto e di fornire dati per supportare il processo di modifica che rimuoverà le principali cause responsabili di una quota significativa dei difetti. L’analisi causale è effettuata di solito da chi diagnostica e risolve il problema o che dichiara che il problema non deve o non può essere risolto. Questo è di solito lo sviluppatore. L’individuazione di un valore preliminare della causa di un difetto viene di solito fatta dall'Analista del Test, che formula un'ipotesi riguardo a ciò che ha probabilmente causato il problema. Quando la correzione è confermata, l'Analista del Test verifica il valore della causa inserita dallo sviluppatore. Quando la causa viene determinata, è anche prassi comune determinare o confermare la fase in cui il difetto è stato introdotto.

Le cause tipiche dei difetti includono:

- Requisito non chiaro
- Requisito mancante
- Requisito errato
- Implementazione della progettazione non corretta
- Implementazione dell'interfaccia errata
- Errore di logica del codice
- Errore di calcolo
- Errore hardware
- Errore di interfaccia
- Dati non validi

Le informazioni sulle cause vengono aggregate per determinare i problemi più comuni che conducono alla creazione di difetti. Ad esempio, se un gran numero di difetti sono causati da requisiti poco chiari, avrebbe senso applicare uno sforzo maggiore nel condurre efficaci riesami dei requisiti. Allo stesso modo se l’implementazione delle interfacce costituisce un problema tra i gruppi di sviluppo, potrebbero essere necessarie sessioni di progettazione comuni.

Utilizzando le informazioni sulle cause per il miglioramento dei processi, si aiuta l'organizzazione a monitorare i benefici di efficaci modifiche di processo e di quantificare i costi dei difetti che possono essere attribuiti ad una causa particolare. Questo può aiutare a richiedere risorse per le modifiche di processo che richiedano l'acquisto di ulteriori strumenti ed attrezzature, nonché la modifica della schedulazione temporale. Il syllabus ISTQB Livello Esperto "Migliorare il processo di Test" [ISTQB_EL_ITP] tratta l'analisi causale in modo più dettagliato.

7. Strumenti di Test – 45 minuti

Termini

Testing guidato-da-parole-chiave, strumenti di preparazione dati di test, strumenti di progettazione test, strumenti di esecuzione test

Obiettivi di apprendimento per Strumenti di Test

7.2 Strumenti di Test ed Automazione

TA-7.2.1 (K2) Spiegare i vantaggi di utilizzare strumenti di preparazione dati di test, di progettazione dei test e di esecuzione dei test
TA-7.2.2 (K2) Spiegare il ruolo dell’Analista del Test nell’automazione “keyword-driven”
TA-7.2.3 (K2) Spiegare i passaggi per la risoluzione di un errore nell’esecuzione di test automatizzati

7.1 Introduzione

Gli strumenti di test possono migliorare notevolmente l'efficienza e l'accuratezza delle attività di test, ma solo se sono strumenti adeguati e implementati in modo corretto. Gli strumenti di test devono essere trattati come un aspetto altrettanto importante di una ben gestita organizzazione di test. La sofisticazione e l'applicabilità degli strumenti di test variano notevolmente ed il loro mercato è in continua evoluzione. Gli strumenti sono di solito ottenuti da fornitori commerciali, da siti freeware o shareware.

7.2 Strumenti di Test ed Automazione

Gran parte del lavoro di un Analista del Test richiede l'utilizzo efficace di strumenti. Sapere quali strumenti e quando utilizzarli può aumentare l'efficienza dell’ Analista del Test e può contribuire a fornire una migliore copertura del testing nel tempo consentito.

7.2.1 Strumenti di progettazione di test

Gli strumenti di progettazione dei test sono utilizzati per facilitare la creazione dei test case e dei dati di test. Questi strumenti possono essere alimentati da specifici formati di documentazione dei requisiti, da modelli (ad esempio, UML), o da input forniti dall’Analista del Test. Gli strumenti di progettazione dei test sono spesso disegnati e costruiti per lavorare con formati particolari e con prodotti particolari, quali strumenti di gestione dei requisiti.

Gli strumenti di progettazione dei test sono in grado di fornire informazioni per l'Analista del Test, utili per determinare i tipi di test che sono necessari per ottenere il livello desiderato di copertura del testing, la fiducia nel sistema o azioni di mitigazione del rischio di prodotto. Ad esempio, gli strumenti di alberi di classificazione generano (e evidenziano) l'insieme di combinazioni necessarie per raggiungere una copertura completa sulla base del criterio di copertura scelto. Questa informazione poi può essere utilizzata dall'Analista del Test per determinare i test case che devono essere eseguiti.

7.2.2 Strumenti di preparazione dei dati di test

Gli Strumenti di preparazione dei dati di test offrono diversi vantaggi. Alcuni strumenti sono in grado di analizzare un documento (come un documento di requisiti) o anche il codice sorgente per

determinare i dati necessari durante il testing per ottenere un certo livello di copertura. Altri strumenti possono estrarre un insieme di dati da un sistema di produzione e "mascherare" o "anonimizzare" alcuni campi per eliminare tutte le informazioni personali, pur mantenendo la loro integrità interna. I dati mascherati possono quindi essere utilizzati per il testing senza il rischio di una perdita di sicurezza o uso scorretto delle informazioni personali. Ciò è particolarmente importante quando siano necessari grandi quantità di dati realistici. Altri strumenti possono generare dati di test da insiemi di parametri di input (ad esempio, per test casuali); alcuni di questi analizzano la struttura del database per determinare quali input sono richiesti all’Analista dei Test.

7.2.3 Strumenti di esecuzione di test automatizzati

Gli strumenti di esecuzione dei test sono per lo più utilizzati dagli Analisti del Test per eseguire i test e verificarne l'esito, a tutti i livelli di test. L'obiettivo per utilizzare uno strumento di esecuzione del test è tipicamente uno o più dei seguenti:

- Ridurre i costi (in termini di sforzo e / o di tempo)
- Eseguire altri test
- Eseguire gli stessi test in molteplici ambienti
- Rendere più ripetibile l'esecuzione dei test
- Eseguire test impossibili da eseguire manualmente (ad esempio, test di validazione di dati di grandi dimensioni)

Questi obiettivi spesso si sovrappongono all’obiettivo principale di aumentare la copertura riducendo i costi.

7.2.3.1 Applicabilità

Il ritorno dell’investimento per gli strumenti di esecuzione dei test è di solito più alto quando si automatizzano i test di regressione, per la riduzione del previsto sforzo di manutenzione tramite una esecuzione ripetuta dei test. L’automazione degli “smoke test” può essere particolarmente efficace per l'uso frequente dei test, la necessità di un risultato rapido e, anche se il costo di manutenzione può essere maggiore, la possibilità di disporre di un metodo automatico per valutare una nuova “build” in ambiente di integrazione continua.

Gli strumenti di esecuzione dei test sono comunemente utilizzati nei test di sistema e di integrazione. Alcuni strumenti, in particolare strumenti di verifica delle API, possono essere utilizzati pure a livello di test di componente. L’utilizzo degli strumenti nelle aree dove sono più efficacemente applicabili contribuirà a migliorare il ritorno degli investimenti.

7.2.3.2 Nozioni di base degli strumenti di test

Gli strumenti di esecuzione dei test eseguono una serie di istruzioni scritte in un linguaggio di programmazione, spesso chiamato “linguaggio di scripting”. Le istruzioni per lo strumento sono ad un livello molto dettagliato, che specifica gli input, l'ordine ed i valori specifici utilizzati per gli input, nonché gli output previsti. Questo può rendere gli script dettagliati particolarmente sensibili alle variazioni del software in test, in particolare quando lo strumento interagisce con l'interfaccia grafica utente (GUI). La maggior parte degli strumenti di esecuzione dei test includono un comparatore, che offre la possibilità di confrontare un risultato effettivo con un risultato atteso memorizzato.

7.2.3.3 Implementazione dell’automazione del testing

La tendenza nell’automazione dell’esecuzione dei test (come nella programmazione) è di passare da dettagliate istruzioni di basso livello a linguaggi di alto livello, utilizzando librerie, macro e

sotto-programmi. Tecniche di progettazione come “keyword-driven” e “action word-driven” catturano una serie di istruzioni e fanno poi riferimento ad esse con una particolare "keyword" o "action word". In questo modo l'Analista dei Test può scrivere i test case in un linguaggio umano, ignorando il linguaggio di programmazione di base e funzioni di basso livello. Utilizzando questa tecnica di scrittura modulare si ottiene una più facile manutenzione durante modifiche alle funzionalità ed all’interfaccia del software in test. [Bath08] . L'uso di keyword nell’esecuzione di script automatizzati è discusso più sotto.

Dei modelli possono essere utilizzati per guidare la creazione delle "keyword" o "action word". Osservando i modelli dei processi di business, che sono spesso inclusi nei documenti di requisiti, l'Analista del Test è in grado di determinare quali processi di business chiave debbano essere testati. Possono poi essere definiti i passi di questi processi, inclusi i punti di decisione da verificarsi durante i processi stessi. I punti di decisione possono diventare le "action word" che lo strumento di test può importare ed utilizzare a partire da fogli di calcolo di "keyword" o "action word". La modellazione dei processi è un importante metodo di documentazione dei processi di business e può essere utilizzata per identificare i processi chiave ed i punti di decisione. La modellazione può essere fatta manualmente o utilizzando strumenti che utilizzano input basati su regole di business e descrizioni di processo.

7.2.3.4 Migliorare il successo dello sforzo di automazione

Nel determinare quali test automatizzare, ogni caso di test o suite di test candidati devono essere valutati per verificare se l'automazione valga la pena. Molti progetti di automazione che non hanno avuto successo si sono basati sull’automazione di test case manuali più prontamente disponibili, senza controllare il beneficio reale. Beneficio che può essere ottimale per un certo insieme di test case (una suite), ma che può essere limitato o negativo per altri insiemi.

In sede di attuazione di un progetto di automazione del testing devono essere considerati i seguenti aspetti.

Possibili vantaggi:

- Il tempo di esecuzione automatica dei test diviene più prevedibile
- Il test di regressione e la convalida dei difetti saranno più veloci e più affidabili nel prosieguo del progetto
- Lo stato e la crescita professionale del test o del team può essere migliorata
- L’automazione può essere particolarmente utile nei cicli di vita di sviluppo iterativo ed incrementale fornendo migliori test di regressione per ogni “build” o iterazione
- La copertura di alcuni tipi di test può essere possibile solo con strumenti automatizzati (ad esempio, convalida di grandi insiemi di dati)
- L’esecuzione può essere più conveniente rispetto al testing manuale per massicci input di dati, test di grandi conversioni e comparazioni, tramite input e verifiche più veloci e consistenti

Possibili rischi:

- Test manuali, inefficaci, incompleti o non corretti, possono essere automatizzati "come sono"
- Il testware può essere difficile da mantenere, richiedendo numerose rettifiche quando il software in test viene modificato
- Il coinvolgimento diretto del tester in esecuzione può essere ridotto, con conseguente minor rilevazione di difetti
- Il team di test può non avere skill sufficienti per utilizzare gli strumenti in modo efficace

- Possono essere automatizzati, solo perché esistono e sono stabili, anche test irrilevanti, che non contribuiscono alla copertura complessiva del testing
- I test possono diventare improduttivi man mano che il software si stabilizza (paradosso pesticida)

Durante l’implementazione di uno strumento di automazione dell’esecuzione dei test, non è sempre consigliabile automatizzare i test manuali così come sono, ma è buona norma ridefinire i test case per un migliore utilizzo dell'automazione. Questo include la formattazione dei test case, il riutilizzo di modelli, l'utilizzo di variabili in input invece di utilizzare valori hard-coded e lo sfruttamento di tutti i vantaggi dello strumento di test. Gli strumenti di esecuzione dei test di solito sono in grado di gestire test multipli, gruppi di test, test ripetuti e modifiche nell'ordine di esecuzione, fornendo sempre comunque funzionalità di analisi e reporting.

Per molti strumenti di automazione dell’esecuzione dei test, sono necessarie buone capacità di programmazione per creare test efficienti ed efficaci (script) e suite di test. E’ frequente che le grandi suite di test automatizzati siano molto difficili da gestire ed aggiornare, se non sono state progettate con cura. Un’adeguata formazione in materia di strumenti di test, di programmazione e di tecniche di progettazione è pertanto utile, per assicurarsi di ottenere dagli strumenti tutti i vantaggi possibili.

Durante la pianificazione dei test, è importante prevedere il tempo per eseguire periodicamente i test automatizzati manualmente, così da mantenere la conoscenza di come funziona il test, verificarne il corretto funzionamento e riesaminare la validità dei dati di input e della copertura.

7.2.3.5 Automazione keyword-driven

Le keywords (a volte indicate come action words) sono per lo più, ma non esclusivamente, utilizzate per rappresentare ad alto livello interazioni di business con un sistema (per esempio, "annullare l'ordine"). Ogni keyword (parola chiave) è in genere utilizzata per rappresentare una sequenza di interazioni dettagliate tra un attore e il sistema in test. Delle sequenze di parole chiave (compresi i pertinenti dati di test) vengono utilizzate per descrivere i test case. [Buwalda01]

Una keyword è implementata come uno o più script di test eseguibili. Gli strumenti leggono i test case come una sequenza di keywords che richiamano gli script di esecuzione dei test. Gli script sono progettati in modo altamente modulare per consentire la mappatura facilitata a specifiche keywords. Sono perciò necessarie buone capacità di programmazione per implementare questi script modulari.

I vantaggi principali dell’automazione dei test keyword-driven sono:

- Le keywords che si riferiscono ad una particolare applicazione o dominio di business possono essere definite da esperti del settore, il che può rendere più efficiente il compito di progettare i test case.
- Una persona con competenze di dominio può beneficiare dell’esecuzione automatica dei test case (una volta che le keywords siano state implementate come script), senza la necessità di comprendere il codice sottostante l’automazione.
- I test case scritti utilizzando le keywords sono più facili da mantenere, perché è probabile che abbiano meno bisogno di modifiche, se alcuni dettagli del software in test sono variati.
- Le specifiche dei test case sono indipendenti dalla loro implementazione, poiché le keywords possono essere implementate utilizzando una varietà di linguaggi di scripting e di strumenti.

Gli script di automazione (il reale codice di automazione), che utilizzano le informazioni keyword/action word, sono di solito scritti dagli sviluppatori e dagli Analisti Tecnici del Test, mentre gli Analisti del Test di solito creano e mantengono i dati keyword/action word. Benché l’automazione keyword-driven sia in genere effettuata durante la fase di test del sistema, lo sviluppo del relativo codice può iniziare già nelle fasi di integrazione. In un ambiente iterativo, lo sviluppo dei test automatizzati è un processo continuo.

Una volta creati le keyword di input ed i relativi dati, l’Analista del Test di solito si assume la responsabilità dell'esecuzione delle test case keyword-driven e di analizzare gli eventuali errori che possono verificarsi. Quando si verifica un’anomalia, l'Analista del Test deve ricercarne la causa per determinare se il problema deriva dalle keywords, dai dati di input, dallo script di automazione o dall'applicazione in fase di test. Generalmente la fase iniziale della ricerca del problema è la riesecuzione manuale dello stesso test con gli stessi dati, per vedere se la causa dell’anomalia è l'applicazione stessa. Se con questo non viene evidenziata l’anomalia, l'Analista del Test dovrebbe rivedere la sequenza dei test che hanno portato all’esito negativo, per determinare se il problema si è verificato in un passo precedente (forse con la creazione di dati non corretti), ma si è evidenziato poi nell'elaborazione successiva. Se l'Analista del Test è in grado di determinare la causa del problema, deve consegnare le informazioni per la sua risoluzione all’Analista Tecnico del Test o allo sviluppatore per ulteriori analisi.

7.2.3.6 Cause del fallimento del progetto di automazione

I progetti di automazione dell’esecuzione dei test spesso non riescono a raggiungere i loro obiettivi. Gli errori possono essere dovuti a scarsa flessibilità nell'uso dello strumento, a capacità di programmazione insufficienti nel test team o ad una aspettativa non realistica dei problemi che possono essere risolti con l'automazione. E’ importante notare che qualsiasi automazione dell’esecuzione dei test richiede una gestione accurata, uno sforzo notevole, competenze adeguate ed un'attenzione continua, così come per ogni progetto di sviluppo software. Occorre dedicare sufficiente tempo alla creazione di una architettura sostenibile, seguendo delle pratiche di progettazione corrette, fornendo una efficace gestione della configurazione e seguendo buone pratiche di codifica. Gli script di test automatizzati devono essere testati in quanto è probabile che presentino difetti e potrebbe essere necessario ottimizzarne le prestazioni. Uno strumento di usabilità dovrebbe essere preso in considerazione, non solo per lo sviluppatore, ma anche per le persone che utilizzeranno lo strumento di automazione per eseguire gli script. Potrebbe essere necessario progettare un'interfaccia tra lo strumento e l'utente, che fornisca l'accesso ai test case in un modo logicamente organizzato per il tester, ma fornendo sempre l'accessibilità necessaria per lo strumento.

8. Riferimenti

8.1 Standard

[ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE) Chapters 1 and 4

[ISO9126] ISO/IEC 9126-1:2001, Software Engineering - Software Product Quality, Chapters 1 and 4

[RTCA DO-178B/ED-12B]: Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12B.1992.
Chapter 1

8.2 Documenti ISTQB

[ISTQB_AL_OVIEW] ISTQB Advanced Level Overview, Version 0.1
[ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 0.1
[ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 0.1
[ISTQB_FL_SYL] ISTQB Foundation Level Syllabus, Version 2011
[ISTQB_GLOSSARY] Standard glossary of terms used in Software Testing, Version 2.2, 2012

8.3 Libri

[Bath08] Graham Bath, Judy McKay, “The Software Test Engineer’s Handbook”, Rocky Nook, 2008, ISBN 978-1-933952-24-6
[Beizer95] Boris Beizer, “Black-box Testing”, John Wiley & Sons, 1995, ISBN 0-471-12094-4
[Black02]: Rex Black, “Managing the Testing Process (2nd edition)”, John Wiley & Sons: New York, 2002, ISBN 0-471-22398-0
[Black07]: Rex Black, “Pragmatic Software Testing”, John Wiley and Sons, 2007, ISBN 978-0-470-12790-2
[Buwalda01]: Hans Buwalda, “Integrated Test Design and Automation”, Addison-Wesley Longman, 2001, ISBN 0-201-73725-6
[Cohn04]: Mike Cohn, “User Stories Applied: For Agile Software Development”, Addison-Wesley Professional, 2004, ISBN 0-321-20568-5
[Copeland03]: Lee Copeland, “A Practitioner's Guide to Software Test Design”, Artech House, 2003, ISBN 1-58053-791-X
[Craig02]: Rick David Craig, Stefan P. Jaskiel, “Systematic Software Testing”, Artech House, 2002, ISBN 1-580-53508-9
[Gerrard02]: Paul Gerrard, Neil Thompson, “Risk-based e-business Testing”, Artech House, 2002, ISBN 1-580-53314-0
[Gilb93]: Tom Gilb, Graham Dorothy, “Software Inspection”, Addison-Wesley, 1993, ISBN 0-201-63181-4
[Graham07]: Dorothy Graham, Erik van Veenendaal, Isabel Evans, Rex Black “Foundations of Software Testing”, Thomson Learning, 2007, ISBN 978-1-84480-355-2
[Grochmann94]: M. Grochmann (1994), Test case design using Classification Trees, in: conference proceedings STAR 1994
[Koomen06]: Tim Koomen, Leo van der Aalst, Bart Broekman, Michiel Vroon "TMap NEXT, for result driven testing", UTN Publishers, 2006, ISBN 90-72194-80-2
[Myers79]: Glenford J. Myers, “The Art of Software Testing”, John Wiley & Sons, 1979, ISBN 0-471-46912-2
[Splaine01]: Steven Splaine, Stefan P. Jaskiel, “The Web-Testing Handbook”, STQE Publishing, 2001, ISBN 0-970-43630-0
[vanVeenendaal12]: Erik van Veenendaal, “Practical risk-based testing – The PRISMA approach”, UTN Publishers, The Netherlands, ISBN 9789490986070

[Wiegers03]: Karl Wiegers, “Software Requirements 2”, Microsoft Press, 2003, ISBN 0-735-61879-8
[Whittaker03]: James Whittaker, “How to Break Software”, Addison-Wesley, 2003, ISBN 0-201-79619-8
[Whittaker09]: James Whittaker, “Exploratory Software Testing”, Addison-Wesley, 2009, ISBN 0-321-63641-4

8.4 Altri Riferimenti

I seguenti riferimenti indicano le informazioni disponibili su Internet o altrove. Anche se questi riferimenti sono stati controllati al momento della pubblicazione del presente Syllabus Advanced Level, l'ISTQB non può essere ritenuta responsabile se i riferimenti non sono più disponibili.

- Capitolo 3

- Czerwonka, Jacek: www.pairwise.org
- Bug Taxonomy: www.testingeducation.org/a/bsct2.pdf
- Sample Bug Taxonomy based on Boris Beizer’s work: inet.uni2.dk/~vinter/bugtaxst.doc
- Good overview of various taxonomies: testingeducation.org/a/bugtax.pdf
- Heuristic Risk-Based Testing By James Bach
- From “Exploratory & Risk-Based Testing (2004) www.testingeducation.org"
- Exploring Exploratory Testing , Cem Kaner and Andy Tikam , 2003
- Pettichord, Bret, “An Exploratory Testing Workshop Report”, www.testingcraft.com/exploratorypettichord

- Capitolo 4

- www.testingstandards.co.uk

9. Indice