Certified Tester Advanced Level Test Analyst (CTAL-TA) Syllabus

v3.1.0

International Software Testing Qualifications Board

Storico delle Revisioni

Versione	Data	Osservazioni
2012 v2.0	19.10.2012	Prima versione come Syllabus AL-TA separato
2019 v3.0	19.10.2019	Aggiornamento principale con revisione completa e riduzione dell'ambito
V3.1.0	03.03.2021	Aggiornamento minore con riscrittura del capitolo 3.2.3 e diversi miglioramenti nel testo

Per maggiori dettagli, vedere le release note

Sommario

Ringraziamenti

Questo documento è stato prodotto da una task force di Test Analyst del Working Group Advanced Level di International Software Testing Qualifications Board: Mette Bruhn-Pedersen (Working Group Chair); Matthias Hamburg (Product Owner); Wim Decoutere, István Forgács, Adam Roman, Jan Sabak, Marc-Florian Wendland (Autori).

La task force ringrazia Paul Weymouth e Richard Green per la stesura tecnica, Gary Mogyorodi per le verifiche di conformità del Glossario, e i Member Board per i loro commenti relativi alla review della versione 2019 del Syllabus che è stata pubblicata e per le modifiche proposte.

Le seguenti persone hanno partecipato alla review e ai commenti di questo Syllabus:
Gery Ágnecz, Armin Born, Chenyifan, Klaudia Dussa-Zieger, Chen Geng (Kevin), Istvan Gercsák, Richard Green, Ole Chr. Hansen, Zsolt Hargitai, Andreas Hetz, Tobias Horn, Joan Killeen, Attila Kovacs, Rik Marselis, Marton Matyas, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Pe'er, Palma Polyak, Nishan Portoyan, Meile Posthuma, Stuart Reid, Murian Song, Péter Sótér, Lucjan Stapp, Benjamin Timmermans, Chris van Bael, Stephanie van Dijck, Paul Weymouth.

Questo documento è stato rilasciato da ISTQB® in data 23 Febbraio 2021

La versione 2019 di questo documento è stata prodotto da un team dedicato del Working Group International Software Testing Qualifications Board Advanced Level: Graham Bath, Judy McKay, Mike Smith

Le seguenti persone hanno partecipato alla review, ai commenti e alla votazione della versione 2019 di questo Syllabus:
Laura Albert, Markus Beck, Henriett Braunné Bokor, Francisca Cano Ortiz, Guo Chaonian, Wim Decoutere, Milena Donato, Klaudia Dussa-Zieger, Melinda Eckrich-Brajer, Péter Földházi Jr, David Frei, Chen Geng, Matthias Hamburg, Zsolt Hargitai, Zhai Hongbao, Tobias Horn, Ágota Horváth, Beata Karpinska, Attila Kovács, József Kreisz, Dietrich Leimsner, Ren Liang, Claire Lohr, Ramit Manohar Kaul, Rik Marselis, Marton Matyas, Don Mills, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Peer, Pálma Polyák, Meile Posthuma, Lloyd Roden, Adam Roman, Abhishek Sharma, Péter Sótér, Lucjan Stapp, Andrea Szabó, Jan te Kock, Benjamin Timmermans, Chris Van Bael, Erik van Veenendaal, Jan Versmissen, Carsten Weise, Robert Werkhoven, Paul Weymouth.

La versione 2012 di questo documento è stata prodotta da un team dedicato del Sub Working Group Advanced Level di International Software Testing Qualifications Board– Advanced Test Analyst: Judy McKay (Chair), Mike Smith, Erik van Veenendaal.

Ai tempi in cui il Syllabus Advanced Level era stato completato, il Working Group Advanced Level era composto dai seguenti membri (in ordine alfabetico):
Graham Bath, Rex Black, Maria Clara Choucair, Debra Friedenberg, Bernard Homès (Vice Chair), Paul Jorgensen, Judy McKay, Jamie Mitchell, Thomas Mueller, Klaus Olsen, Kenji Onishi, Meile Posthuma, Eric Riou du Cosquer, Jan Sabak, Hans Schaefer, Mike Smith (Chair), Geoff Thompson, Erik van Veenendaal, Tsuyoshi Yumoto.

Le seguenti persone hanno partecipato alla review, ai commenti e alla votazione della versione 2012 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.

0. Introduzione a questo Syllabus

0.1 Scopo di questo Syllabus

Questo Syllabus costituisce la base per l'International Software Testing Qualification a livello Advanced Level per il Test Analyst. ISTQB® fornisce questo Syllabus come segue:

1. Ai Board nazionali, per tradurlo nella loro lingua locale e per accreditare i Training Provider. I Board nazionali possono adattare il Syllabus alle loro specifiche esigenze linguistiche e aggiungere i riferimenti per adattarlo alle pubblicazioni locali.
2. Agli organismi di certificazione, per derivare le domande d'esame nella loro lingua locale adattate agli obiettivi di apprendimento del Syllabus stesso.
3. Ai Training Provider, per produrre materiale didattico e determinare i metodi di insegnamento appropriati.
4. Ai candidati alla certificazione, per prepararsi all'esame di certificazione (come parte di un corso di formazione o autonomamente).
5. Alla comunità internazionale di System Engineering e Software Engineering, per promuovere la professione del testing dei sistemi e del software e come base per libri e articoli.

ISTQB® può consentire ad altre entità di utilizzare questo Syllabus per altre finalità, a condizione che richiedano e ottengano anticipatamente autorizzazione scritta da parte dello stesso ISTQB®.

0.2 Il Certified Tester Advanced Level nel Software Testing

La certificazione Advanced Level Core si compone di tre Syllabi separati relativi ai seguenti ruoli:

- Test Manager
- Test Analyst
- Technical Test Analyst

ISTQB® Advanced Level Overview 2019 è un documento separato [ISTQB_AL_OVIEW] che include le seguenti informazioni:

- Business Outcomes di ogni Syllabus
- Matrice di tracciabilità tra i Business Outcome e gli Obiettivi di Apprendimento
- Riassunto di ogni Syllabus
- Relazioni tra i Syllabi

0.3 Obiettivi di Apprendimento Esaminabili e Livelli di Conoscenza

Gli Obiettivi di Apprendimento supportano i Business Outcome e vengono utilizzati per creare gli esami per il conseguimento della certificazione Advanced Test Analyst.

I livelli di conoscenza degli obiettivi di apprendimento relativi ai livelli K2, K3 e K4 sono specificati all'inizio di ogni capitolo e sono classificati come segue:

- K2: Comprendere
- K3: Applicare
- K4: Analizzare

Le definizioni di tutti i termini elencati come parole chiave appena sotto i titoli dei capitoli (K1) devono essere ricordati, anche se non esplicitamente menzionate negli obiettivi di apprendimento.

0.4 L'esame Advanced Level Test Analyst

L'esame per la certificazione Advanced Level Test Analyst si baserà su questo Syllabus. Le risposte alle domande d'esame possono richiedere l'uso di materiale basato su più di un paragrafo di questo Syllabus. Tutti i paragrafi del Syllabus sono esaminabili, ad eccezione dell'introduzione e delle Appendici. Standard, libri e altri Syllabi ISTQB® sono inclusi come riferimenti, ma il loro contenuto non è esaminabile, al di là di quanto sintetizzato in questo Syllabus relativamente al loro contenuto.

L'esame è formato da 40 domande a scelta multipla. Per superare l'esame deve essere ottenuto almeno il 65% dei punti totali.

Gli esami possono essere sostenuti come parte di un corso di formazione accreditato o intrapresi in modo indipendente (ad es. presso un centro d'esame o in un esame pubblico). Il completamento di un corso di formazione accreditato non è un prerequisito per l'esame.

0.5 Requisiti di Ammissione per l'Esame

È necessario ottenere la certificazione Certified Tester Foundation Level prima di sostenere l'esame di certificazione Advanced Level Test Analyst.

0.6 Aspettative di Esperienza

Nessuno degli obiettivi di Apprendimento per l'Advanced Test Analyst presuppone un'esperienza specifica.

0.7 Accreditamento dei Corsi

Un Member Board riconosciuto da ISTQB® può accreditare i Training Provider, il cui materiale didattico segue questo Syllabus. I Training Provider dovrebbero ottenere le linee guida per l'accreditamento dal Member Board o dall'ente che effettua l'accreditamento. Un corso accreditato viene riconosciuto essere conforme a questo Syllabus, ed è autorizzato ad avere un esame ISTQB® come parte del corso stesso.

0.8 Livello di Dettaglio del Syllabus

Il livello di dettaglio di questo Syllabus consente di svolgere corsi di formazione ed esami consistenti a livello internazionale. Per raggiungere questo obiettivo, il Syllabus consiste di:

- Un'indicazione di obiettivi didattici che descrivono i propositi dell'Advanced Level Test Analyst
- Una lista di elementi che gli studenti devono ricordare
- Gli Obiettivi di Apprendimento per ogni area di conoscenza, che descrivono il risultato dell'apprendimento cognitivo da raggiungere
- Una descrizione dei concetti chiave, inclusi i riferimenti a determinate fonti, come ad esempio standard o letteratura universalmente accettata

Il contenuto del Syllabus non è una descrizione dell'intera area di conoscenza; riflette il livello di dettaglio che deve essere coperto dai corsi di formazione Advanced Level. Si focalizza su materiali che possono essere applicati a tutti i progetti software, incluso lo sviluppo software Agile. Questo Syllabus non contiene obiettivi di apprendimento specifici relativi a un particolare ciclo di vita dello sviluppo software (SDLC, Software Development Lifecycle), ma descrive come questi concetti si applicano nello sviluppo software Agile, in altri tipi di cicli di vita iterativi e incrementali, e nei cicli di vita sequenziali.

0.9 Come è Organizzato questo Syllabus

Sono identificabili sei capitoli con contenuti esaminabili. Nell'intestazione principale di ogni capitolo è specificato il tempo minimo per l'insegnamento e per l'esercitazione da dedicare al capitolo; non vengono forniti i tempi per i paragrafi interni al capitolo. Per i corsi di formazione accreditati, il Syllabus richiede un minimo di 20 ore e 30 minuti di istruzione, distribuiti nei sei capitoli come segue:

- Capitolo 1: Le Attività del Test Analyst nel Processo di Test (150 minuti)
- Capitolo 2: Le Attività del Test Analyst nel Testing basato sul Rischio (60 minuti)
- Capitolo 3: Tecniche di Test (630 minuti)
- Capitolo 4: Testing delle Caratteristiche di Qualità del Software (180 minuti)
- Capitolo 5: Review (120 minuti)
- Capitolo 6: Strumenti di Test e Test Automation (90 minuti)

1. Le Attività del Test Analyst nel Processo di Test - 150 min.

Parole Chiave

analisi dei test, base di test, condizione di test, criteri di uscita, dati di test, esecuzione dei test, implementazione dei test, procedura di test, progettazione dei test, schedulazione dell'esecuzione dei test, test, test case, test case di alto livello, test case di basso livello, test suite

Obiettivi di Apprendimento per le Attività del Test Analyst nel Processo di Test

1.1 Introduzione

Nessun Obiettivo di Apprendimento

1.2 Testing nel Ciclo di Vita del Software

TA-1.2.1 (K2) Spiegare come e perché le tempistiche e il livello di coinvolgimento del Test Analyst variano quando si lavora con diversi modelli di ciclo di vita del software

1.3 Analisi dei Test

TA-1.3.1 (K2) Riassumere i compiti del Test Analyst durante le attività di analisi

1.4 Progettazione dei Test

TA-1.4.1 (K2) Spiegare perché le condizioni di test dovrebbero essere comprese dagli stakeholder
TA-1.4.2 (K4) Per un determinato progetto, selezionare il livello di progettazione appropriato per i test case (di alto livello o di basso livello)
TA-1.4.3 (K2) Spiegare i problemi da considerare nella progettazione dei test case

1.5 Implementazione dei Test

TA-1.5.1 (K2) Riassumere i compiti del Test Analyst durante le attività di implementazione dei test

1.6 Esecuzione dei Test

TA-1.6.1 (K2) Riassumere i compiti del Test Analyst durante le attività di esecuzione dei test

1.1 Introduzione

Nel Syllabus Foundation Level di ISTQB®, la descrizione del processo di test include le seguenti attività:

- Pianificazione dei test
- Monitoraggio e controllo dei test
- Analisi dei test
- Progettazione dei test
- Implementazione dei test
- Esecuzione dei test
- Completamento dei test

In questo Syllabus Advanced Level Test Analyst, vengono considerate in modo più approfondito le attività che hanno particolare rilevanza per il Test Analyst. Questo fornisce un ulteriore raffinamento del processo di test, per adattarsi meglio ai diversi modelli di ciclo di vita dello sviluppo software (SDLC, Software Development Lifecycle).

Determinare i test appropriati, progettarli, implementarli ed eseguirli sono le principali aree di concentrazione per il Test Analyst. Sebbene sia importante comprendere gli altri passi del processo di test, la maggior parte del lavoro del Test Analyst si focalizza normalmente sulle seguenti attività:

- Analisi dei test
- Progettazione dei test
- Implementazione dei test
- Esecuzione dei test

Le altre attività del processo di test sono adeguatamente descritte nel Foundation Level e non hanno bisogno di ulteriori elaborazioni in questo Advanced Level.

1.2 Testing nel Ciclo di Vita dello Sviluppo Software

Quando si definisce una strategia di test, dovrebbe essere preso in considerazione il SDLC globale. Il momento di coinvolgimento del Test Analyst è differente in base ai diversi SDLC; anche la quantità di coinvolgimento, il tempo richiesto, le informazioni disponibili e le aspettative possono variare notevolmente. Il Test Analyst deve essere consapevole dei tipi di informazione da fornire agli altri ruoli correlati dell'organizzazione, come:

- Requirements Engineering e Requirements Management – Riscontri sulle review dei requisiti
- Project Management – Input alla schedulazione
- Configuration Management e Change Management – Risultati del testing di verifica della build, informazioni sul controllo delle versioni
- Sviluppo del software - Notifiche dei difetti rilevati
- Manutenzione del software – Report sui difetti, efficacia nell'eliminazione dei difetti, e testing confermativo
- Supporto tecnico - Documentazione accurata per i workaround e i problemi noti
- Produzione di documentazione tecnica (ad es. specifiche di progettazione del database, documentazione relativa all'ambiente di test) - Input a questi documenti e relativa review tecnica dei documenti

Le attività di test devono essere allineate con il SDLC selezionato la cui natura può essere sequenziale, iterativa, incrementale o un ibrido di questi. Ad esempio, nel V-model sequenziale, il processo di test applicato a livello di test di sistema potrebbe allinearsi come segue:

- La pianificazione dei test di sistema viene svolta in modo concorrente alla pianificazione di progetto, e il monitoraggio e controllo dei test continuano fino al completamento dei test. Questo

influenzerà gli input alla schedulazione forniti dal Test Analyst per gli scopi del project management.
- L'analisi e la progettazione dei test di sistema si allineano a documenti quali la specifica dei requisiti di sistema, le specifiche di progettazione di sistema e architetturale (di alto livello) e la specifica di progettazione dei componenti (di basso livello).
- L'implementazione dell'ambiente del test di sistema potrebbe iniziare durante la progettazione di sistema, sebbene si verifichi generalmente in modo concorrente alle attività di codifica e al testing di componente, con le attività di implementazione dei test di sistema che si estendono spesso fino a pochi giorni prima dell'inizio dell'esecuzione dei test di sistema.
- L'esecuzione dei test di sistema inizia quando i criteri di ingresso sono soddisfatti o, se necessario, esclusi, il che significa in genere che almeno il testing di componente e spesso anche il testing di integrazione dei componenti hanno soddisfatto i loro criteri di uscita. L'esecuzione dei test di sistema continua fino a quando non vengono soddisfatti i criteri di uscita del test di sistema.
- Le attività di completamento dei test di sistema si verificano dopo aver soddisfatto i criteri di uscita del test di sistema.

I modelli iterativi e incrementali possono non seguire lo stesso ordine di attività e possono escluderne alcune. Ad esempio, un modello iterativo può utilizzare una serie ridotta di attività di test per ogni iterazione. L'analisi, la progettazione, l'implementazione e l'esecuzione dei test possono essere condotte per ogni iterazione, mentre la pianificazione di alto livello viene svolta all'inizio del progetto e le attività di chiusura vengono eseguite alla fine del progetto.

In uno sviluppo software Agile è comune utilizzare un processo meno formalizzato e un relazione di collaborazione molto più stretta con gli stakeholder di progetto, che permette di implementare più facilmente modifiche all'interno del progetto. Potrebbe non esistere un ruolo di Test Analyst ben definito. La documentazione di test è meno completa e la comunicazione è più breve e frequente.

Lo sviluppo software Agile coinvolge il testing fin dall'inizio. Il testing inizia dall'avvio dello sviluppo del prodotto, quando gli sviluppatori svolgono la progettazione e l'architettura iniziale. Le review possono non essere formalizzate ma sono continue man mano che il software evolve. Ci si aspetta un coinvolgimento per tutta la durata del progetto e che il team esegua i compiti del Test Analyst.

I modelli iterativi e incrementali vanno dallo sviluppo software Agile, in cui esiste un'aspettativa di cambiamento man mano che i requisiti del cliente evolvono, fino a modelli ibridi, per esempio uno sviluppo iterativo/incrementale combinato con un approccio V-model. In tali modelli ibridi, i Test Analyst dovrebbero essere coinvolti negli aspetti di pianificazione e progettazione delle attività sequenziali, per poi passare a un ruolo più interattivo durante le attività iterative/incrementali.

Qualunque sia il SDLC utilizzato, i Test Analyst devono comprendere le aspettative di coinvolgimento e le tempistiche di tale coinvolgimento. I Test Analyst forniscono un contributo efficace alla qualità del software aggiustando le loro attività e il loro momento di coinvolgimento allo specifico SDLC piuttosto che attenersi a un modello di ruolo prestabilito.

1.3 Analisi dei Test

Durante la pianificazione dei test, viene definito l'ambito del progetto di test. Durante l'analisi dei test, i Test Analyst utilizzano questa definizione di ambito per:
- Analizzare la base di test
- Identificare difetti di vario tipo nella base di test
- Identificare e prioritizzare le condizioni di test e le caratteristiche da testare
- Catturare la tracciabilità bidirezionale tra ogni elemento della base di test e le condizioni di test associate

- Eseguire attività associate al testing basato sul rischio (si veda il Capitolo 2)

Per permettere ai Test Analyst di procedere in modo efficace con l'analisi dei test, devono essere soddisfatti i seguenti criteri di ingresso:

- Esiste una base di conoscenza (ad es. requisiti, user story) che descrive l'oggetto di test, che può essere usata come base di test (si veda [ISTQB_FL_SYL], Paragrafi 1.4.2 e 2.2 o una lista di altre possibili fonti della base di test).
- Questa base di test ha superato la review con risultati ragionevoli ed è stata opportunamente aggiornata dopo la review. Se devono essere definiti test case di alto livello (si veda il paragrafo 1.4.1), può non essere ancora necessario definire completamente la base di test. Nello sviluppo software Agile, questo ciclo di review sarà iterativo poiché le user story vengono raffinate all'inizio di ogni iterazione.
- Sono disponibili un budget e una schedulazione approvati per eseguire le attività di test rimanenti per questo oggetto di test.

Le condizioni di test sono in genere identificate dall'analisi della base di test insieme agli obiettivi del test (come definito nella pianificazione dei test). In alcune situazioni, in cui la documentazione può essere vecchia o inesistente, le condizioni di test possono essere identificate durante discussioni con gli stakeholder rilevanti (ad es. in workshop o durante la pianificazione dell'iterazione). Nello sviluppo software Agile, i criteri di accettazione, che sono definiti come parte delle user story, sono spesso usati come base per la progettazione dei test.

Sebbene le condizioni di test siano in genere specifiche dell'elemento da testare, esistono alcune considerazioni standard per il Test Analyst.

- Di solito è consigliabile definire le condizioni di test a differenti livelli di dettaglio. Inizialmente, vengono identificate condizioni di alto livello per definire gli obiettivi generali per il testing, come ad esempio la “Funzionalità dello schermo x”. Successivamente, vengono identificate condizioni più dettagliate come base di specifici test case, come ad esempio "Lo schermo x rifiuta un numero di account di lunghezza inferiore a 1 rispetto a quella corretta". L'uso di questo tipo di approccio gerarchico per la definizione delle condizioni di test può aiutare a garantire che la copertura sia sufficiente per gli elementi di alto livello. Inoltre, questo approccio consente a un Test Analyst di iniziare a lavorare sulla definizione delle condizioni di test di alto livello per le user story che non sono ancora state raffinate.
- Se i rischi di prodotto sono stati definiti, allora dovrebbero essere identificate le condizioni di test che saranno necessarie per indirizzare ogni rischio di prodotto, tracciandole rispetto a quell’elemento di rischio.

L'applicazione delle tecniche di test (come identificate all’interno della strategia di test e/o del test plan) possono essere utili nelle attività di analisi dei test e possono essere utilizzate per supportare i seguenti obiettivi:

- Identificare le condizioni di test
- Ridurre la probabilità di dimenticare importanti condizioni di test
- Definire condizioni di test più precise e accurate

Dopo che le condizioni di test sono state identificate e raffinate, può essere eseguita una review di queste condizioni di test con gli stakeholder, per garantire che i requisiti siano chiaramente compresi e che il testing sia allineato con gli obiettivi del progetto.

Al termine delle attività di analisi dei test per una determinata area (ad es. una funzione specifica), il Test Analyst dovrebbe sapere quali test specifici devono essere progettati per quell'area.

1.4 Progettazione dei Test

Sempre aderendo all'ambito determinato durante la pianificazione dei test, il processo di test continua mentre il Test Analyst progetta i test che verranno implementati ed eseguiti. La progettazione dei test include le seguenti attività:

- Determinare in quali aree di test sono appropriati i test case di basso livello o i test case di alto livello
- Determinare le tecniche di test che consentiranno di raggiungere la copertura necessaria. Le tecniche che possono essere utilizzate sono stabilite durante la pianificazione dei test
- Utilizzare tecniche di test per progettare i test case e gli insiemi di test case che coprono le condizioni di test identificate
- Identificare i dati di test necessari a supportare le condizioni di test e i test case
- Progettare l'ambiente di test e identificare qualsiasi infrastruttura richiesta, compresi gli strumenti
- Catturare la tracciabilità bidirezionale (ad es. tra base di test, condizioni di test e test case)

I criteri di prioritizzazione identificati durante l'analisi del rischio e la pianificazione dei test dovrebbero essere applicati durante tutto il processo, dall'analisi e progettazione fino all'implementazione ed esecuzione dei test.

In base ai tipi di test da progettare, uno dei criteri di ingresso per la progettazione dei test può essere la disponibilità di strumenti che verranno utilizzati durante la progettazione.

Durante la progettazione dei test, il Test Analyst deve considerare almeno quanto segue:

- Alcuni elementi di test vengono indirizzati meglio definendo solo le condizioni di test piuttosto che approfondire la definizione di test script, che forniscono la sequenza di istruzioni necessarie per eseguire un test. In questo caso, le condizioni di test dovrebbero essere definite per essere utilizzate come guida per il testing senza script predeterminati (testing unscripted).
- I criteri di pass/fail dovrebbero essere chiaramente identificati.
- I test dovrebbero essere progettati per essere comprensibili da altri tester, non solo dall'autore. Se l'autore non è la persona che esegue il test, altri tester dovranno leggere e capire i test precedentemente specificati per comprendere gli obiettivi del test e la relativa importanza del test.
- I test devono anche essere comprensibili agli altri stakeholder, come gli sviluppatori (che possono eseguire la review dei test) e gli auditor (che possono dover approvare i test).
- I test dovrebbero coprire tutti i tipi di interazione con l'oggetto di test e non dovrebbero essere limitati alle interazioni delle persone attraverso l'interfaccia visibile all'utente. Possono anche includere, ad esempio, l'interazione con altri sistemi ed eventi tecnici o fisici. (si veda [IREB_CPRE] per ulteriori dettagli).
- I test dovrebbero essere progettati per testare le interfacce tra i vari oggetti di test, oltre ai comportamenti degli oggetti stessi.
- L'effort di progettazione dei test deve essere prioritizzato e bilanciato per allinearlo ai livelli di rischio e al valore di business.

1.4.1 Test Case di Alto Livello e Test Case di Basso Livello

Uno dei compiti del Test Analyst è quello di determinare il miglior livello di progettazione dei test case per una determinata situazione. I test case di alto livello e di basso livello sono coperti in [ISTQB_FL_SYL]. Alcuni dei vantaggi e degli svantaggi nel loro utilizzo sono descritti nei seguenti elenchi:

I test case di basso livello offrono i seguenti vantaggi:

- Staff di test inesperto può fare affidamento su informazioni dettagliate fornite nell'ambito del progetto. I test case di basso livello forniscono tutte le informazioni e le procedure specifiche necessarie al tester per eseguire il test case (compresi eventuali requisiti relativi ai dati) e verificare i risultati ottenuti.
- I test possono essere rieseguiti da tester differenti e dovrebbero essere ottenuti gli stessi risultati dei test.
- Possono essere rivelati nella base di test dei difetti non ovvi.
- Il livello di dettaglio consente una verifica indipendente dei test, come gli audit, se richiesto.
- Il tempo impiegato per l'implementazione dei test case automatizzati può essere ridotto.

I test case di basso livello presentano i seguenti svantaggi:
- Possono richiedere un notevole sforzo, sia per la creazione che per la manutenzione.
- Tendono a limitare l'ingegnosità del tester durante l'esecuzione.
- Richiedono che la base di test sia ben definita.
- La loro tracciabilità verso le condizioni di test può richiedere uno sforzo maggiore rispetto ai test case di alto livello.

I test case di alto livello offrono i seguenti vantaggi:
- Forniscono linee guida per quello che dovrebbe essere testato, e consentono al Test Analyst di variare i dati effettivi o persino la procedura seguita durante l'esecuzione dei test.
- Possono fornire una copertura dei rischi migliore rispetto ai test case di basso livello perché possono variare ogni volta che vengono eseguiti.
- Possono essere definiti in fase iniziale durante il processo dei requisiti.
- Quando il test viene eseguito, viene utilizzata l'esperienza del Test Analyst nel testing e nell'oggetto di test.
- Possono essere definiti quando non è richiesta alcuna documentazione dettagliata e formale.
- Sono più adatti per il riutilizzo in diversi cicli di test, quando è possibile utilizzare dati di test differenti.

I test case di alto livello presentano i seguenti svantaggi:
- Sono meno riproducibili, rendendo difficile la verifica. Questo perché manca la descrizione dettagliata presente nei test case di basso livello.
- Può essere necessario uno staff di test più esperto per eseguirli
- Quando si automatizza a partire dai test case di alto livello, la mancanza di dettaglio può causare la validazione di risultati ottenuti errati oppure la mancanza di elementi che dovrebbero essere validati.

I test case di alto livello possono essere utilizzati per sviluppare i test case di basso livello quando i requisiti diventano più definiti e stabili. In questo caso, la creazione dei test case viene svolta in modo sequenziale: a partire dai test case di alto livello vengono definiti solo i test case di basso livello che saranno usati per l'esecuzione.

1.4.2 Progettazione dei Test Case

I test case sono progettati dall'elaborazione e dal raffinamento graduale delle condizioni di test identificate, utilizzando le tecniche di test (si veda il capitolo 3). I test case dovrebbero essere ripetibili, verificabili e tracciabili rispetto alla base di test (ad es. i requisiti).

La progettazione dei test include l'identificazione di quanto segue:
- Obiettivo (cioè, l'obiettivo osservabile e misurabile dell'esecuzione dei test)
- Precondizioni, come i requisiti di progetto o dell'ambiente di test locale e i piani per il loro rilascio, lo stato del sistema prima dell'esecuzione dei test, ecc.

- Requisiti dei dati di test (sia i dati di input per il test case, sia i dati che devono essere presenti nel sistema per l'esecuzione del test case)
- Risultati attesi con i criteri espliciti di pass/fail
- Postcondizioni, come i dati interessati dai test, lo stato del sistema dopo l'esecuzione dei test, i trigger per l'elaborazione successiva, ecc.

Una sfida particolare può essere la definizione del risultato atteso di un test. Calcolarlo manualmente è spesso noioso e soggetto a errori; se possibile, potrebbe essere preferibile trovare o creare un oracolo del test automatizzato. Nell'individuare il risultato atteso, i tester si preoccupano non solo degli output sullo schermo, ma anche dei dati e delle postcondizioni dell'ambiente. Se la base di test è chiaramente definita, identificare il risultato corretto dovrebbe essere teoricamente semplice. Tuttavia, la documentazione della base di test potrebbe essere vaga, contraddittoria, mancante di copertura delle aree chiave, o mancante del tutto. In questi casi, un Test Analyst deve avere esperienza in materia o avere accesso ad essa. Inoltre, anche quando la base di test è ben specificata, interazioni complesse di stimoli e risposte complicate possono rendere difficile la definizione dei risultati attesi; pertanto, un oracolo del test è essenziale. Nello sviluppo software Agile, l'oracolo del test potrebbe essere il Product Owner. L'esecuzione del test case senza poter determinare in alcun modo la correttezza dei risultati ottenuti potrebbe avere un beneficio o un valore aggiunto molto basso, generando spesso test report invalidi o una falsa confidenza nel sistema.

Le attività sopra descritte possono essere applicate a tutti i livelli di test, anche se la base di test sarà differente. Quando si analizzano e progettano i test, oltre all'obiettivo del test è importante ricordare il livello target per il test. Questo aiuta a determinare il livello di dettaglio richiesto e tutti gli strumenti che possono essere necessari (ad es. driver e stub a livello del test di componente).

Durante lo sviluppo delle condizioni di test e dei test case, viene normalmente creata della documentazione, che genera i prodotti di lavoro del test. In pratica, quanto dovrà essere documentato nei prodotti di lavoro del test varia notevolmente e può essere influenzato da:

- I rischi di progetto (cosa deve/non deve essere documentato)
- Il valore aggiunto che la documentazione apporta al progetto
- Gli standard da seguire e/o le normative da rispettare
- Il SDLC o l'approccio utilizzato (ad es. lo scopo di un approccio Agile è avere documentazione "quanto basta")
- Il requisito di tracciabilità verso la base di test attraverso l'analisi e la progettazione dei test

In base all'ambito del testing, l'analisi e la progettazione dei test indirizzano le caratteristiche di qualità per l'oggetto di test. Lo standard ISO 25010 [ISO25010] fornisce un riferimento utile. Durante il testing di sistemi hardware/software, possono essere applicate ulteriori caratteristiche.

Le attività di analisi e di progettazione dei test possono essere migliorate se svolte insieme alla review e all'analisi statica. Infatti, condurre l'analisi e la progettazione dei test è spesso una forma di testing statico, perché durante questa attività possono essere rilevati problemi nei documenti della base di test. L'analisi e la progettazione dei test basati sulla specifica dei requisiti sono un modo eccellente per prepararsi per un meeting di review dei requisiti. La lettura dei requisiti da utilizzare per la creazione di test richiede la comprensione del requisito e la capacità di determinare un modo per validarlo. Questa attività rivela spesso requisiti mancanti, requisiti che non sono chiari, non sono verificabili o che non hanno definiti i criteri di accettazione. Allo stesso modo, i prodotti di lavoro del test, come i test case, le analisi del rischio e i test plan possono essere sottoposti a review.

Durante la progettazione dei test, possono essere definiti i requisiti dettagliati richiesti per l'infrastruttura di test, anche se in pratica questi non possono essere finalizzati fino all'implementazione dei test. Deve essere ricordato che l'infrastruttura di test può non includere soltanto oggetti di test e testware. Ad esempio, i requisiti di infrastruttura possono includere locali, attrezzature, personale, software,

strumenti, periferiche, apparecchiature di comunicazione, autorizzazioni degli utenti e tutti gli altri elementi necessari per eseguire i test.

I criteri di uscita per l'analisi e la progettazione dei test varieranno in base ai parametri del progetto, ma tutti gli elementi discussi in questi due paragrafi dovrebbero essere considerati inclusi nei criteri di uscita definiti. È importante che i criteri di uscita siano misurabili, che siano state fornite tutte le informazioni richieste per i passi successivi, e che tutta la preparazione necessaria sia stata eseguita.

1.5 Implementazione dei Test

L'implementazione dei test prepara il testware necessario per l'esecuzione dei test in base all'analisi e alla progettazione dei test. L'implementazione dei test include le seguenti attività:

- Sviluppare le procedure di test e, potenzialmente, creare test script automatizzati
- Organizzare le procedure di test e i test script automatizzati (se esistono) in test suite da eseguire in uno specifico ciclo di test (test run)
- Consultare il Test Manager per la prioritizzazione dei test case e delle test suite da eseguire
- Creare una schedulazione dell'esecuzione dei test, inclusa l'allocazione delle risorse, per consentire l'inizio dell'esecuzione dei test (si veda [ISTQB_FL_SYL], paragrafo 5.2.4)
- Finalizzare la preparazione dei dati di test e degli ambienti di test
- Aggiornare la tracciabilità tra la base di test e il testware, come le condizioni di test, i test case, le procedure di test, i test script e le test suite.

Durante l'implementazione dei test, i Test Analyst identificano un'efficiente ordine di esecuzione dei test case e creano le procedure di test. La definizione delle procedure di test richiede l'identificazione accurata dei vincoli e delle dipendenze che potrebbero influenzare la sequenza di esecuzione dei test. Le procedure di test documentano qualsiasi precondizione iniziale (ad es. il caricamento dei dati di test da un repository di dati) e qualsiasi attività successiva all'esecuzione (ad es. il ripristino dello stato del sistema).

I Test Analyst identificano le procedure di test e i test script automatizzati che possono essere raggruppati (ad es. tutti quelli relativi al testing di un particolare processo di business di alto livello) e li organizzano in test suite. Questo permette di eseguire insieme i test case correlati.

I Test Analyst organizzano le test suite all'interno di una schedulazione dell'esecuzione dei test in modo che risulti un'efficiente esecuzione dei test. Se viene utilizzata una strategia di test basata sul rischio, il livello di rischio sarà l'elemento principale da prendere in considerazione per determinare l'ordine di esecuzione dei test case. Possono esistere altri fattori che determinano l'ordine di esecuzione dei test case, come la disponibilità delle persone giuste, delle attrezzature, dei dati e della funzionalità da testare.

Non è insolito che il codice venga rilasciato in differenti momenti e che l'effort del test debba essere coordinato in base alla sequenza con cui il software diventa disponibile per il testing. Soprattutto nei modelli di sviluppo iterativi e incrementali, è importante che il Test Analyst si coordini con il team di sviluppo per garantire che il software sia rilasciato per il testing in un ordine testabile.

Il livello di dettaglio e la complessità associata al lavoro svolto durante l'implementazione dei test possono essere influenzati dal dettaglio delle condizioni di test e dei test case. In alcuni casi vengono applicate delle normative, e i prodotti di test dovrebbero fornire prove di conformità agli standard applicabili, come lo standard americano DO-178C (in Europa, ED 12C). [RTCA DO-178C/ED-12C].

Come specificato sopra, i dati di test sono necessari per la maggior parte del testing e in alcuni casi questi insiemi di dati possono essere piuttosto grandi. Durante l'implementazione, i Test Analyst creano i dati di input e i dati ambientali da caricare nei database e in altri archivi similari. Questi dati devono

essere "adatti allo scopo" (“fir for purpose”) per consentire il rilevamento di difetti. I Test Analyst possono anche creare dei dati da utilizzare durante il testing keyword-driven e data-driven (si veda il Paragrafo 6.2), e durante il testing manuale.

L'implementazione dei test riguarda anche l'ambiente (o ambienti) di test. Durante questa attività, l’ambiente deve essere completamente configurato e verificato prima dell'esecuzione dei test. È essenziale un ambiente di test "fit for purpose", ovvero un ambiente di test che dovrebbe essere in grado di consentire la rilevazione dei difetti durante il testing controllato, di funzionare normalmente quando non si verificano failure, e di replicare adeguatamente, se richiesto, l’ambiente di produzione o dell’utente finale per i livelli di test più alti. Possono essere necessarie modifiche all'ambiente di test durante l'esecuzione dei test, dovute a modifiche non previste, risultati di test o altre considerazioni. Se si verificano modifiche all'ambiente durante l'esecuzione, è importante valutare l'impatto delle modifiche sui test che sono già stati eseguiti.

Durante l'implementazione dei test, i Test Analyst dovrebbero verificare che i responsabili della creazione e della manutenzione dell'ambiente di test siano conosciuti e disponibili, e che tutto il testware, gli strumenti di supporto ai test e i processi associati siano pronti per l'uso. Questo include il configuration management, il defect management, il test management e il logging dei test. Inoltre, i Test Analyst devono verificare le procedure che raccolgono i dati per la valutazione dello stato attuale rispetto ai criteri di uscita e al reporting dei risultati del test.

È saggio utilizzare un approccio bilanciato per l'implementazione dei test, che viene determinato durante la pianificazione dei test. Ad esempio, le strategie di test analitiche basate sul rischio sono spesso combinate con strategie di test reattive. In questo caso, una percentuale dell’effort di implementazione dei test è allocata al testing che non segue script predeterminati (testing unscripted).

Il testing unscripted non dovrebbe essere casuale o senza scopo, poiché potrebbe essere imprevedibile in termini di durata e copertura, e potrebbe rilevare pochi difetti. Piuttosto, dovrebbe essere condotto in sessioni time-boxed, ciascuna con una direzione iniziale fornita da un test charter, ma con la libertà di allontanarsi dalle indicazioni del test charter nel caso in cui durante la sessione vengano scoperte opportunità di test potenzialmente più produttive. Nel corso degli anni, i tester hanno sviluppato una varietà di tecniche di test basate sull’esperienza, come gli attacchi [Whittaker03], l’error guessing [Myers11] e il testing esplorativo [Whittaker09]. L'analisi dei test, la progettazione dei test e l'implementazione dei test si verificano ancora, ma vengono svolti principalmente durante l'esecuzione dei test.

Quando si seguono queste strategie di test reattive, i risultati di ogni test influenzano l'analisi, la progettazione e l'implementazione dei test successivi. Queste strategie sono leggere (lightweight) e spesso efficaci nella ricerca di difetti, ma esistono alcuni svantaggi, tra i quali:

- È richiesta competenza da parte del Test Analyst
- La durata può essere difficile da prevedere
- La copertura può essere difficile da tracciare
- Senza una buona documentazione o il supporto di strumenti, la ripetibilità può non essere fattibile

1.6 Esecuzione dei Test

L'esecuzione dei test viene condotta in base alla schedulazione dell’esecuzione dei test, e include le seguenti attività: (si veda [ISTQB_FL_SYL])

- Eseguire i test manuali, incluso il testing esplorativo
- Eseguire i test automatizzati
- Confrontare i risultati ottenuti con i risultati attesi
- Analizzare le anomalie per stabilire le loro probabili cause

- Fare un reporting dei difetti in base ai failure osservati
- Registrare i risultati ottenuti dall'esecuzione dei test
- Aggiornare la tracciabilità tra la base di test e il testware per considerare i risultati dei test
- Eseguire i regression test

Le attività di esecuzione dei test sopra elencate possono essere condotte dal tester o dal Test Analyst.

Di seguito sono riportate tipiche attività aggiuntive che possono essere eseguite dal Test Analyst:

- Riconoscere cluster di difetti che possono indicare la necessità di testing aggiuntivo su una parte particolare dell'oggetto di test
- Fornire suggerimenti per future sessioni di testing esplorativo in base alle scoperte rilevate dal testing esplorativo
- Identificare nuovi rischi dalle informazioni ottenute durante lo svolgimento delle attività di esecuzione dei test
- Fornire suggerimenti per migliorare i prodotti di lavoro generati dall'attività di implementazione dei test (ad esempio, miglioramenti delle procedure di test)

2. Le Attività del Test Analyst nel Testing Basato sul Rischio - 60 min.

Parole Chiave
rischio di prodotto, identificazione del rischio, mitigazione del rischio, testing basato sul rischio

Obiettivi di Apprendimento per le Attività del Test Analyst nel Testing Basato sul Rischio

Le Attività del Test Analyst nel Testing Basato sul Rischio

TA-2.1.1 (K3) Per una determinata situazione, partecipare all'identificazione del rischio, eseguire una valutazione del rischio e proporre una mitigazione del rischio appropriata

2.1 Introduzione

I Test Manager hanno spesso la responsabilità globale di stabilire e gestire una strategia di test basata sul rischio. I Test Manager richiedono generalmente il coinvolgimento di un Test Analyst per garantire che l'approccio basato sul rischio sia implementato correttamente.

I Test Analyst dovrebbero essere attivamente coinvolti nelle seguenti attività del testing basato sul rischio:

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

Queste attività vengono svolte in modo iterativo durante il SDLC per affrontare i nuovi rischi, cambiare le priorità, e per valutare e comunicare regolarmente lo stato dei rischi (si veda [vanVeenendaal12] e [Black02] per ulteriori dettagli). Nello sviluppo software Agile, le tre attività sono spesso combinate nella cosiddetta sessione di rischio con il focus su un'iterazione o su un rilascio.

I Test Analyst dovrebbero operare all'interno del framework del testing basato sul rischio stabilito per il progetto dal Test Manager. Dovrebbero contribuire grazie alla loro conoscenza dei rischi del dominio di business che sono inerenti al progetto, come i rischi relativi alla safety funzionale, ai problemi economici e di business, e ai fattori politici.

2.2 Identificazione del Rischio

Prendendo in considerazione il più ampio campione possibile di stakeholder, è molto probabile che il processo di identificazione del rischio rilevi il maggior numero possibile di rischi significativi.

I Test Analyst possiedono spesso una conoscenza unica riguardo al particolare dominio di business del sistema sotto test. Questo significa che sono particolarmente adatti per svolgere le seguenti attività:

- Condurre interviste con esperti del dominio e utenti
- Condurre valutazioni indipendenti
- Utilizzare template di rischio
- Partecipare a workshop sui rischi
- Partecipare a sessioni di brainstorming con utenti reali e potenziali
- Definire checklist di test
- Considerare esperienze passate con sistemi o progetti similari

In particolare, i Test Analyst dovrebbero lavorare a stretto contatto con gli utenti e gli altri esperti del dominio (ad esempio i Requirements Engineer o i Business Analyst) per determinare le aree a rischio di business che dovrebbero essere indirizzate durante il testing. Nello sviluppo software Agile, questa stretta relazione con gli stakeholder consente di condurre l'identificazione del rischio in modo sistematico, ad esempio durante gli iteration planning meeting.

I rischi che potrebbero essere identificati in un progetto includono:

- Problemi di correttezza funzionale, ad es. calcoli errati
- Problemi di usabilità, ad es. tasti di scelta rapida (shortcut) insufficienti
- Problemi di portabilità, ad es. impossibilità di installare un'applicazione su piattaforme particolari

2.3 Valutazione del Rischio

Mentre l'identificazione del rischio riguarda l'identificazione del maggior numero possibile di rischi pertinenti, la valutazione del rischio riguarda lo studio di questi rischi identificati. In particolare, viene svolta la classificazione di ogni rischio e la valutazione del relativo livello di rischio.

Determinare il livello di rischio implica in genere la valutazione, per ogni elemento di rischio, della probabilità del rischio e dell'impatto del rischio. La probabilità del rischio è generalmente interpretata come la probabilità che il problema potenziale possa esistere nel sistema sotto test e possa essere osservato quando il sistema è in produzione. I Technical Test Analyst dovrebbero contribuire a trovare e comprendere la potenziale probabilità per ogni elemento di rischio, mentre i Test Analyst contribuiscono a comprendere il potenziale impatto di business del problema qualora si verifichi (nello sviluppo software Agile questa distinzione basata sul ruolo potrebbe essere meno forte).

L'impatto del rischio viene spesso interpretato come la severità dell'effetto sugli utenti, clienti o altri stakeholder. In altre parole, deriva dal rischio di business. Per ogni elemento di rischio, i Test Analyst dovrebbero contribuire all'identificazione e alla valutazione del potenziale impatto sul dominio di business o sull'utente. I fattori che influenzano il rischio di business includono:

- Frequenza di utilizzo della funzionalità impattata
- Perdita di business
- Danno finanziario
- Perdite o responsabilità ecologiche o sociali
- Sanzioni legali civili o penali
- Problemi di safety funzionale
- Multe, perdita di licenze
- Mancanza di soluzioni alternative (workaround) nel caso in cui le persone non riescano più lavorare
- Visibilità della funzionalità
- Visibilità del failure che porta a pubblicità negativa e danno potenziale all'immagine
- Perdita di clienti

In base alle informazioni disponibili sul rischio, i Test Analyst devono stabilire i livelli di rischio di business in base alle linee guida fornite dal Test Manager. Questi potrebbero essere classificati utilizzando una scala ordinale (valori numerici, oppure basso/medio/alto), oppure i colori del semaforo (rosso/arancio/giallo). Una volta che sono stati assegnati la probabilità del rischio e l'impatto del rischio, i Test Manager utilizzano questi valori per determinare il livello di rischio per ogni elemento di rischio. Il livello di rischio viene quindi utilizzato per prioritizzare le attività di mitigazione del rischio [vanVeenendaal12].

2.4 Mitigazione del Rischio

Durante il progetto, i Test Analyst dovrebbero cercare di eseguire le seguenti operazioni:

- Ridurre il rischio di prodotto progettando test case efficaci che dimostrino in modo non ambiguo se i test sono passati o falliti, e partecipando alle review dei prodotti di lavoro software, come i requisiti, le progettazioni e la documentazione utente
- Implementare appropriate attività di mitigazione del rischio identificate nella strategia di test e nel test plan (ad esempio, testare un processo di business particolarmente ad alto rischio utilizzando particolari tecniche di test)
- Rivalutare i rischi conosciuti sulla base di ulteriori informazioni raccolte durante lo svolgimento del progetto, modificando in modo appropriato la probabilità del rischio, l'impatto del rischio, o entrambi
- Identificare nuovi rischi in base alle informazioni ottenute durante il testing

Quando si parla di rischi di prodotto, il testing fornisce un contributo essenziale per mitigare tali rischi. Rilevando i difetti, i tester riducono il rischio fornendo la consapevolezza dei difetti e l'opportunità di correggerli prima del rilascio. Se i tester non rilevano difetti, allora il testing riduce il rischio fornendo l'evidenza che, in determinate condizioni (cioè le condizioni testate), il sistema funziona correttamente. I Test Analyst aiutano a determinare le opzioni di mitigazione del rischio investigando le opportunità per raccogliere dati di test accurati, creare e testare scenari utente realistici, e condurre o supervisionare studi di usabilità.

2.4.1 Prioritizzare i Test

Il livello di rischio viene anche utilizzato per prioritizzare i test. Un Test Analyst potrebbe determinare se esiste un rischio elevato nell'accuratezza delle transazioni di un sistema bancario. Di conseguenza, per mitigare il rischio, il tester può lavorare con altri esperti del dominio di business per raccogliere un insieme di dati campione, che possono essere elaborati e verificati rispetto all'accuratezza. Allo stesso modo, un Test Analyst potrebbe determinare che i problemi di usabilità sono un rischio significativo per un nuovo oggetto di test. Piuttosto che aspettare di rilevare problemi durante uno user acceptance test, il Test Analyst potrebbe prioritizzare un test di usabilità in fase iniziale, basato su un prototipo, per aiutare a identificare e risolvere i problemi di progettazione dell'usabilità prima dello user acceptance test. Questa prioritizzazione deve essere presa in considerazione il prima possibile nelle fasi di pianificazione, in modo che la schedulazione possa tenere conto del testing necessario nei tempi richiesti.

In alcuni casi, tutti i test relativi ai rischi più elevati vengono eseguiti prima di qualsiasi test relativo ai rischi più bassi, e i test vengono eseguiti seguendo in modo rigoroso l'ordine di rischio (approccio "depth-first"). In altri casi, viene utilizzato un approccio di campionamento per selezionare un campione di test tra tutte le aree di rischio identificate utilizzando il livello di rischio per pesare la selezione, garantendo allo stesso tempo la copertura di ogni rischio almeno una volta (approccio "breadth-first").

Indipendentemente dall'approccio usato per il testing basato sul rischio, depth-first o breadth-first, è possibile che il tempo allocato per il testing possa essere utilizzato senza che tutti i test vengano eseguiti. Il testing basato sul rischio permette ai tester di riportare al management in termini di livello di rischio residuo rispetto a un specifico momento, e permette al management di decidere se estendere il testing o se trasferire il rischio residuo sugli utenti, i clienti, l'help desk/supporto tecnico e/o lo staff operativo.

2.4.2 Adeguare il Testing per Futuri Cicli di Test

La valutazione del rischio non è un'attività eseguita solo una volta prima dell'inizio dell'implementazione dei test; è un processo continuo. Ogni ciclo futuro di test pianificato dovrebbe essere sottoposto a nuove analisi del rischio per tener conto di fattori quali:

- Eventuali rischi di prodotto nuovi o modificati in modo significativo
- Aree instabili o soggette a failure rilevate durante il testing
- Nuovi rischi generati da difetti corretti
- Difetti tipici rilevati durante il testing
- Aree che sono state testate poco (bassa copertura dei requisiti)

3. Tecniche di Test - 630 min.

Parole Chiave

analisi ai valori limite, error guessing, partizionamento di equivalenza, tassonomia dei difetti, tecnica dell'albero di classificazione, tecnica di test basata sui difetti, tecnica di test basata sull'esperienza, tecnica di test black-box, test charter, testing basato sull'esperienza, testing checklist-based, testing degli use case, testing della tabella delle decisioni, testing delle transizioni di stato, testing esplorativo, testing pairwise

Obiettivi di Apprendimento per le Tecniche di Test

3.1 Introduzione

Nessun Obiettivo di Apprendimento

3.2 Tecniche di Test Black-Box

TA-3.2.1 (K4) Analizzare uno o più elementi di specifica e progettare i test case applicando il partizionamento di equivalenza

TA-3.2.2 (K4) Analizzare uno o più elementi di specifica e progettare i test case applicando l'analisi ai valori limite

TA-3.2.3 (K4) Analizzare uno o più elementi di specifica e progettare i test case applicando il testing della tabella delle decisioni

TA-3.2.4 (K4) Analizzare uno o più elementi di specifica e progettare i test case applicando il testing delle transizioni di stato

TA-3.2.5 (K2) Spiegare come i diagrammi dell'albero di classificazione supportano le tecniche di test

TA-3.2.6 (K4) Analizzare uno o più elementi di specifica e progettare i test case applicando il testing pairwise

TA-3.2.7 (K4) Analizzare uno o più elementi di specifica e progettare i test case applicando il testing degli use case

TA-3.2.8 (K4) Analizzare un sistema, o le sue specifiche dei requisiti, per determinare i probabili tipi di difetti da rilevare, e selezionare le appropriate tecniche di test black-box

3.3 Tecniche di Test Basate sull'Esperienza

TA-3.3.1 (K2) Spiegare i principi delle tecniche di test basate sull'esperienza, e i benefici e gli svantaggi rispetto alle tecniche di test black-box e alle tecniche basate sui difetti

TA-3.3.2 (K3) Identificare i test esplorativi da un determinato scenario

TA-3.3.3 (K2) Descrivere l'applicazione delle tecniche di test basate sui difetti e differenziarne l'uso dalle tecniche di test black-box

3.4 Applicazione delle Tecniche di Test più Appropriate

TA-3.4.1 (K4) Per una determinata situazione di progetto, determinare quali tecniche di test black-box o tecniche di test basate sull'esperienza dovrebbero essere applicate per raggiungere obiettivi specifici

3.1 Introduzione

Le tecniche di test considerate in questo capitolo sono suddivise nelle seguenti categorie:

- Black-box
- Basate sull'esperienza

Queste tecniche sono complementari e possono essere utilizzate come appropriate per qualsiasi attività di test, indipendentemente dal livello di test da eseguire.

Si noti che entrambe le categorie di tecniche possono essere utilizzate per testare le caratteristiche di qualità funzionali e non funzionali. Il testing delle caratteristiche del software verrà discusso nel prossimo capitolo.

Le tecniche di test discusse in questi paragrafi possono focalizzarsi principalmente sul determinare dati di test ottimali (ad es. dalle partizioni di equivalenza) o sul derivare procedure di test (ad es. da modelli a stati). È normale combinare le tecniche per creare test case completi.

3.2 Tecniche di Test Black-Box

Le tecniche di test black-box sono introdotte nel Syllabus ISTQB® Foundation Level [ISTQB_FL_SYL].

Le caratteristiche comuni delle tecniche di test black-box includono:

- I modelli, ad es. gli state transition diagram e le tabelle delle decisioni, vengono creati durante la progettazione dei test in base alla tecnica di test
- Le condizioni di test sono derivate sistematicamente da questi modelli

Le tecniche di test forniscono generalmente criteri di copertura, che possono essere utilizzati per misurare le attività di progettazione ed esecuzione dei test. Soddisfare completamente i criteri di copertura non significa che l'insieme di test sia completo, ma piuttosto che il modello non suggerisce più alcun test aggiuntivo, basato su tale tecnica, per aumentare la copertura.

Il testing black-box si basa generalmente su documentazione di specifica, come una specifica dei requisiti di sistema o le user story. Poiché la documentazione di specifica dovrebbe descrivere il comportamento del sistema, in particolare nell'area dell'adeguatezza funzionale, derivare test dai requisiti è spesso parte del testing del comportamento del sistema. In alcuni casi può non esistere documentazione di specifica ma esistono requisiti impliciti, come la sostituzione dell'adeguatezza funzionale di un sistema legacy.

Esistono diverse tecniche di test black-box. Queste tecniche si rivolgono a diversi tipi di software e scenari. I paragrafi seguenti mostrano per ogni tecnica l'applicabilità, alcune limitazioni e difficoltà che il Test Analyst può riscontrare, il metodo con cui viene misurata la copertura e i tipi di difetti che possono essere rilevati.

Per ulteriori dettagli, fare riferimento a [ISO29119-4], [Bath14], [Beizer95], [Black07], [Black09], [Copeland04], [Craig02], [Forgács19], [Koomen06] e [Myers11].

3.2.1 Partizionamento di Equivalenza

Il partizionamento di equivalenza (EP, Equivalence Partitioning) è una tecnica utilizzata per ridurre il numero di test case necessari per testare efficacemente la gestione di input, output, valori interni e valori temporali. Il partizionamento viene utilizzato per creare partizioni di equivalenza (spesso chiamate classi di equivalenza) che vengono create da insiemi di valori che richiedono di essere elaborati nello stesso modo. Selezionando un valore rappresentativo da una partizione, si assume la copertura per tutti gli elementi nella stessa partizione.

Generalmente molti parametri determinano il comportamento dell'oggetto di test. Quando nei test case si combinano partizioni di equivalenza di parametri differenti, possono essere applicate varie tecniche.

Applicabilità
Questa tecnica è applicabile a qualsiasi livello di test ed è appropriata quando ci si aspetta che tutti i membri di un insieme di valori da testare siano gestiti nello stesso modo, e quando gli insiemi di valori utilizzati dall'applicazione sono indipendenti tra loro (non interagiscono). Una classe di equivalenza può essere un qualsiasi insieme non vuoto di valori, per esempio: ordinato, non ordinato, discreto, continuo, infinito, finito, o anche composto da un singolo elemento. La selezione degli insiemi di valori è applicabile a partizioni valide e invalide (ovvero, partizioni contenenti valori che dovrebbero essere considerati invalidi per il software sotto test).

Il partizionamento di equivalenza è più forte quando è utilizzato in combinazione con l'analisi ai valori limite, che espande i valori dei test per includere i valori ai limiti delle partizioni. Il partizionamento di equivalenza, utilizzando i valori delle partizioni valide, è una tecnica comunemente usata per lo smoke testing di una nuova build o di un nuovo rilascio, poiché determina rapidamente se le funzionalità di base funzionano.

Limitazioni/Difficoltà
Se l'assunzione non è corretta e i valori nella partizione non vengono gestiti esattamente allo stesso modo, questa tecnica potrebbe dimenticare dei difetti. È anche importante selezionare accuratamente le partizioni. Ad esempio, un campo di input che accetta numeri positivi e negativi potrebbe essere testato meglio come due partizioni valide, una per i numeri positivi e una per i numeri negativi, a causa della probabilità di una gestione differente. A seconda che sia consentito o meno lo zero, questo potrebbe diventare un'altra partizione. È importante che un Test Analyst comprenda il processo sottostante per determinare il migliore partizionamento dei valori. Questo può richiedere supporto nella comprensione della progettazione del codice.

Il Test Analyst dovrebbe anche tener conto delle possibili dipendenze tra le partizioni di equivalenza di parametri differenti. Ad esempio, in un sistema di prenotazione di voli aerei, il parametro "adulto accompagnatore" può essere usato solo in combinazione con la classe età "bambino".

Copertura
La copertura viene determinata prendendo il numero di partizioni per le quali è stato testato un valore e dividendo quel numero per il numero di partizioni che sono state identificate. La copertura del partizionamento di equivalenza viene quindi dichiarata in percentuale. L'uso di più valori per una singola partizione non aumenta la percentuale di copertura.

Se il comportamento dell'oggetto di test dipende da un singolo parametro, ogni partizione di equivalenza, valida o invalida, dovrebbe essere coperta almeno una volta.

Nel caso di più di un parametro, il Test Analyst dovrebbe selezionare un tipo di copertura semplice o combinatoriale in base al rischio [Offutt16]. Distinguere tra combinazioni che contengono solo partizioni valide e combinazioni che contengono uno o più partizioni invalide è quindi essenziale. Riguardo alle combinazioni con partizioni di equivalenza solo valide, il minimo requisito è una semplice copertura di tutte le partizioni valide per tutti i parametri. Il minimo numero di test case necessari in tale test suite è uguale al numero più grande di partizioni valide di un parametro tra tutti i parametri, assumendo che i parametri siano indipendenti tra loro. Tipi di copertura più completi relativi alle tecniche combinatorie includono la copertura pairwise (si veda il paragrafo 3.2.6), o la copertura completa di qualsiasi combinazione di partizioni valide. Le partizioni di equivalenza invalide dovrebbero essere testate almeno singolarmente, cioè in combinazione con le partizioni valide degli altri parametri, per evitare il mascheramento dei difetti (defect masking). Quindi per una copertura semplice, ogni partizione invalida

contribuisce alla test suite con un test case. In caso di alto rischio, ulteriori combinazioni possono essere aggiunte alla test suite, per es. costituite di sole partizioni invalide o di coppie di partizioni invalide.

Tipi di Difetti
Un Test Analyst utilizza questa tecnica per trovare difetti nella gestione di diversi valori dei dati.

3.2.2 Analisi ai Valori Limite

L'analisi ai valori limite (BVA, Boundary Value Analysis) viene utilizzata per testare la gestione appropriata dei valori esistenti sui limiti delle partizioni di equivalenza ordinate. Esistono due comuni approcci all'analisi ai valori limite: il testing dei valori limite a due valori e il testing dei valori limite a tre valori. Durante il testing dei valori limite a due valori, vengono utilizzati il valore limite (sul limite) e il valore che si trova appena al di fuori del limite (con la più piccola precisione di incremento possibile, in base all'accuratezza richiesta). Ad esempio, per valori di una valuta che ha due posizioni decimali, se la partizione ha incluso i valori da 1 a 10, i due valori per il test del limite superiore saranno 10 e 10,01. I due valori limite per il test del limite inferiore saranno 1 e 0,99. I limiti sono definiti dai valori massimo e minimo nella partizione di equivalenza definita.

Per il testing dei valori limite a tre valori, vengono utilizzati i valori prima del limite, sul limite e dopo il limite. Nell'esempio precedente, i tre valori per il test del limite superiore saranno 9,99, 10 e 10,01. I tre valori per il test del limite inferiore saranno 0,99, 1 e 1,01. La decisione relativa all'utilizzo del testing ai valori limite a due valori oppure a tre valori limite dovrebbe essere basata sul rischio associato all'elemento da testare, essendo l'approccio del valori limite a tre valori utilizzato per gli elementi di rischio più elevato.

Applicabilità
Questa tecnica è applicabile a qualsiasi livello di test ed è appropriata quando esistono partizioni di equivalenza ordinate. Per questo motivo la tecnica di analisi ai valori limite è spesso condotta insieme alla tecnica di partizionamento di equivalenza. Le partizioni di equivalenza sono richieste ordinate perché deve essere ben definito il concetto di essere dentro il limite e fuori dal limite. Ad esempio, un intervallo di numeri è una partizione ordinata. Una partizione che consiste di alcune stringhe di test può anche essere ordinata, ad es. per ordine lessicografico, ma l'ordinamento non è rilevante dal punto di vista del business, quindi i valori limite non dovrebbero essere il focus principale. In aggiunta agli intervalli numerici, le partizioni per cui l'analisi ai valori limite può essere applicata includono:

- Attributi numerici di variabili non numeriche (ad es. lunghezza)
- Numero di cicli di esecuzione di loop, compresi i loop negli state transition diagram
- Numero di elementi di iterazione nelle strutture dati memorizzate, come gli array
- Dimensione degli oggetti fisici (ad es. la memoria)
- Durata delle attività

Limitazioni/Difficoltà
Poiché l'accuratezza di questa tecnica dipende dall'accurata identificazione delle partizioni di equivalenza per identificare correttamente i limiti, l'analisi ai valori limite è soggetta alle stesse limitazioni e difficoltà del partizionamento di equivalenza. Il Test Analyst dovrebbe anche essere consapevole della precisione nei valori validi e invalidi, per poter determinare accuratamente i valori da testare. Solo le partizioni ordinate possono essere utilizzate per l'analisi ai valori limite, ma questo non è limitato a un intervallo di input validi. Ad esempio, quando si esegue il testing del numero di celle supportate da un foglio di calcolo, esiste una partizione che contiene il numero di celle fino al massimo numero consentito, incluso (il limite), e un'altra partizione che inizia con una cella oltre il numero massimo (oltre il limite).

Copertura
La copertura viene determinata prendendo il numero di condizioni limite testate e dividendolo per il numero di condizioni limite identificate (utilizzando il metodo a due o a tre valori). La copertura è espressa in percentuale.

Tipi di Difetti
L'analisi ai valori limite rileva in modo affidabile lo spostamento o l'omissione di limiti, e può rilevare casi di limiti extra. Questa tecnica rileva difetti relativi alla gestione dei valori limite, in particolare errori sulla logica “minore di” e “maggiore di” (ad es., spostamento del valore limite). Può anche essere utilizzata per trovare difetti non funzionali, ad esempio un sistema supporta 10.000 utenti simultanei ma non 10.001.

3.2.3 Testing della Tabella delle Decisioni

Una tabella delle decisioni è una rappresentazione tabulare di un insieme di condizioni e di relative azioni, di regole dichiarate che indicano quali azioni dovranno accadere per quell’insieme di valori delle condizioni [OMG-DMN]. I Test Analyst possono utilizzare le tabelle delle decisioni per analizzare le regole che si applicano al software sotto test e progettare i test che coprono queste regole.

Condizioni e azioni risultanti dell’oggetto di test formano le righe della tabella delle decisioni, normalmente con le condizioni nella parte superiore e le azioni nella parte inferiore. La prima colonna della tabella contiene la descrizione delle condizioni e delle azioni, rispettivamente. Le colonne seguenti, chiamate regole, contengono i valori delle condizioni e i valori delle azioni corrispondenti, rispettivamente,

La tabelle delle decisioni in cui le condizioni sono Booleani con i semplici valori “vero” (“true”) e “falso” (“false”) sono chiamate tabelle delle decisioni “limited-entry” (a valori limitati). Un esempio per una condizione di questo tipo è “Reddito dell’utente < 1000”. Le tabelle delle decisioni “extended-entry” (a valori estesi) permettono di avere condizioni con valori multipli, che possono rappresentare elementi discreti o insiemi di elementi. Ad esempio, una condizione “Reddito dell’utente” può assumere uno dei tre possibili valori: “minore di 1000”, “tra 1000 e 2000”, “maggiore di 2000”.

Le azioni semplici assumono i valori booleani “vero” (“true”) e “falso” (“false”) (ad es. l’azione “sconto ammesso = 20% assume il valore “vero” specificato con “X” se l’azione accade e il valore “falso” specificato con “-“ se non accade). Allo stesso modo delle condizioni, anche le azioni possono assumere valori multipli di altri domini. Ad esempio, un'azione "Sconto ammesso" può assumere uno dei cinque possibili valori: 0%, 10%, 20%, 35% e 50%.

Il testing della tabella delle decisioni inizia progettando le tabelle delle decisioni in base alle specifiche. Le regole che contengono combinazioni impossibili di valori delle condizioni sono escluse o marcate come “non fattibili”. Successivamente, il Test Analyst deve eseguire la review delle tabelle delle decisioni con gli altri stakeholder. Il Test Analyst dovrebbe assicurare che le regole nella tabella siano consistenti (cioè che le regole non si sovrappongono), complete (cioè contengono una regola per ogni combinazione fattibile di valori delle condizioni) e corrette (cioè modellano il comportamento atteso).

Il principio fondamentale nel testing della tabella delle decisioni è che le regole formino le condizioni di test.

Quando si progetta un test case per coprire una specifica regola, il Test Analyst dovrebbe essere consapevole che i valori di input del test case potrebbero essere diversi dai valori delle condizioni della tabella delle decisioni. Ad esempio, il valore “vero” della condizione “reddito annuale > 100,000?” può non essere applicabile direttamente ma può richiedere lavoro da parte del tester per definire l’account di un cliente con un credito superiore a 100,000 in quello specifico anno fiscale. In modo similare, i risultati attesi del test case possono essere differenti dalle azioni della tabella delle decisioni.

Quando la tabella delle decisioni è pronta, le regole devono essere implementate, i test case devono essere progettati selezionando i valori di input del test (e i risultati attesi) che soddisfano le condizioni e le azioni.

Tabelle delle decisioni collassate
Quando si tenta di testare ogni possibile combinazione di input in base alle condizioni, le tabelle delle decisioni possono diventare molto grandi. Una tabella delle decisioni limited-entry completa con N condizioni ha 2^N regole. Una tecnica per ridurre sistematicamente il numero di combinazioni è chiamata testing della tabella delle decisioni collassata [Mosley93]. Quando viene utilizzata questa tecnica, un gruppo di regole con lo stesso insieme di azioni può essere ridotta (collassata) a una regola se, all'interno di questo gruppo, alcune condizioni non sono rilevanti per l'azione, e tutte le altre condizioni rimangono immutate. Nella regola risultante, i valori delle condizioni irrilevanti sono definiti come "don't care" (irrilevanti) e sono generalmente contrassegnati con "-". Per le condizioni con valori "don't care", il Test Analyst può specificare valori validi arbitrari per l'implementazione dei test.

Un'altra situazione per collassare le regole è quella in cui nella combinazione, un valore di una condizione non è applicabile con i valori di alcune condizioni, o quando due o più condizioni hanno valori in conflitto tra loro. Ad esempio, in una tabella delle decisioni per pagamenti con carta di credito, se la condizione "carta valida" è falsa, la condizione "codice PIN corretto" non è applicabile.

Le tabelle delle decisioni collassate possono avere molte meno regole delle tabelle delle decisioni complete, con il risultato di avere un numero inferiore di test case e un effort minore. Se una specifica regola ha dei valori "don't care", e solo un test case copre questa regola, solo uno dei possibili valori della condizione verrà testato per quella regola, e quindi un difetto che coinvolge altri valori può non essere identificato. Quindi, per livelli di rischio alti, allineandosi con il Test Manager, il Test Analyst dovrebbe definire regole separate per ogni combinazione fattibile dei valori delle singoli condizioni piuttosto che collassare la tabella delle decisioni.

Applicabilità
Il testing della tabella delle decisioni è comunemente applicato ai livelli di test di integrazione, di sistema e di accettazione. Può anche essere applicabile al testing di componente quando un componente è responsabile di un insieme di logiche decisionali. Questa tecnica è particolarmente utile quando l'oggetto di test è specificato sotto forma di flowchart (diagrammi di flusso) o tabelle di regole di business.

Le tabelle delle decisioni sono anche una tecnica di definizione dei requisiti e alcune volte le specifiche dei requisiti possono essere già definite in questo formato. Anche il Test Analyst dovrebbe partecipare alla review delle tabelle delle decisioni e analizzarle prima di iniziare la progettazione dei test

Limitazioni/Difficoltà
Quando si considerano le combinazioni di condizioni, trovare tutte le condizioni di interazione può essere sfidante, in particolare quando i requisiti non sono ben definiti o non esistono. Bisogna fare attenzione quando si selezionano le condizioni considerate in una tabella delle decisioni, in modo che il numero di combinazioni di quelle condizioni rimanga gestibile. Nel caso peggiore, il numero di regole crescerà in modo esponenziale.

Copertura
La comune copertura standard per questa tecnica è coprire ogni regola della tabella delle decisioni con un test case. La copertura è misurata come percentuale del numero di regole coperte dalla test suite e il numero di regole fattibili.

L'analisi ai valori limite e il partizionamento di equivalenza possono essere combinate con la tecnica della tabella delle decisioni, specialmente nel caso di tabelle delle decisioni extended-entry. Se le condizioni contengono partizioni di equivalenza che sono totalmente ordinate, i valori limite possono essere usati come valori aggiuntivi che generano regole e test case aggiuntivi.

Tipi di Difetti

I difetti tipici includono l'elaborazione logica errata basata su particolari combinazioni di condizioni che portano a risultati non attesi. Durante la creazione delle tabelle delle decisioni, è possibile rilevare difetti nel documento di specifica. Non è insolito preparare un insieme di condizioni e determinare che il risultato atteso non è specificato per una o più regole. I tipi più comuni di difetti sono omissioni di azioni (cioè, non esiste informazione su ciò che dovrebbe effettivamente accadere in una determinata situazione) e contraddizioni.

3.2.4 Testing delle Transizioni di Stato

Il testing delle transizioni di stato viene utilizzato per testare la capacità dell'oggetto di test di entrare e uscire da stati definiti tramite transizioni valide, di cercare di entrare in stati invalidi o coprire transizioni invalide. Gli eventi causano la transizione dell'oggetto di test da uno stato all'altro e l'esecuzione di azioni. Gli eventi possono essere qualificati da condizioni (a volte chiamate “guard condition” o “transition guard”) che influenzano il tipo di transizione da eseguire. Ad esempio, un evento di login con una combinazione nome utente/password valida genererà una transizione differente rispetto a un evento di login con una password invalida. Queste informazioni sono rappresentate in uno state transition diagram o in una tabella delle transizioni di stato (che può anche includere potenziali transizioni invalide tra stati).

Applicabilità
Il testing delle transizioni di stato è applicabile a qualsiasi software che abbia definito degli stati e degli eventi che causeranno le transizioni tra questi stati (ad es. il cambio di schermate). Il testing delle transizioni di stato può essere applicato a qualsiasi livello di test. Software embedded, software Web e qualsiasi tipo di software transazionale sono ottimi candidati per questo tipo di test. Anche i sistemi di controllo, ad esempio i controller semaforici, sono buoni candidati per questo tipo di test.

Limitazioni/Difficoltà
Determinare gli stati è spesso la parte più difficile della definizione dello state transition diagram o della tabella delle transizioni di stato. Quando l'oggetto di test ha una user interface, le varie schermate visualizzate all'utente vengono spesso rappresentate da stati. Per software embedded, gli stati possono dipendere dagli stati dell'hardware.

Oltre agli stati stessi, l'unità base del testing delle transizioni di stato è la singola transizione. Il semplice testing di tutte le singole transizioni rileverà alcuni tipi di difetti di transizione di stato, ma è possibile trovarne altri verificando le sequenze di transizioni. Una singola transizione è chiamata 0-switch; una sequenza di due transizioni successive è chiamata 1-switch; una sequenza di tre transizioni successive è chiamata 2-switch e così via. In generale, N-switch rappresenta N+1 transizioni successive [Chow1978]. Con l’aumentare di N, il numero di N-switch cresce molto velocemente, rendendo difficile il raggiungimento di una copertura N-switch con un numero ragionevole e piccolo di test.

Copertura
Come con altri tipi di tecniche di test, esiste una gerarchia di livelli di copertura. Il grado minimo di copertura accettabile è quello di aver visitato ogni stato e aver percorso ogni transizione almeno una volta. La copertura delle transizioni al 100% (nota anche come copertura 0-switch) garantirà che ogni stato è stato visitato e ogni transizione è stata percorsa, a meno che la progettazione del sistema o il modello di transizione di stato (diagramma o tabella) siano difettosi. A seconda delle relazioni tra stati e transizioni, può essere necessario percorrere alcune transizioni più di una volta in modo da poter percorrere almeno una volta le altre transizioni.

Il termine "copertura N-switch" è relativa al numero di switch coperti di lunghezza N+1, ed è la percentuale del numero totale di switch di tale lunghezza. Ad esempio, per ottenere una copertura 1-switch al 100%, è necessario che ogni sequenza valida di due transizioni successive sia stata testata

almeno una volta. Questo testing può essere il trigger di alcuni tipi di failure che una copertura 0-switch al 100% non rileverebbe.

La "copertura round-trip" si applica a situazioni in cui le sequenze di transizioni formano loop (cicli). La copertura round-trip al 100% viene raggiunta quando tutti i loop da qualsiasi stato allo stesso stato sono stati testati per tutti gli stati in cui i loop iniziano e finiscono. Questo loop non può contenere più di un'occorrenza di qualsiasi stato (ad eccezione di quello iniziale/finale) [Offutt16].

Per tutti questi approcci, un grado di copertura ancora più elevato tenterà di includere tutte le transizioni invalide identificate in una tabella delle transizioni di stato. I requisiti di copertura e gli insiemi da coprire per il testing delle transizioni di stato devono specificare se sono incluse transizioni invalide.

La progettazione dei test case per ottenere la copertura desiderata è supportata dallo state transition diagram o dalla tabella delle transizioni di stato per il particolare oggetto di test. Queste informazioni possono anche essere rappresentate in una tabella che mostra le transizioni N-switch per un valore particolare di N [Black09].

È possibile applicare una procedura manuale per identificare gli elementi da coprire (ad esempio, transizioni, stati o N-switch). Un metodo suggerito è quello di stampare lo state transition diagram e la tabella delle transizioni di stato, e di utilizzare una penna o una matita per contrassegnare gli elementi coperti fino a quando viene raggiunta la copertura richiesta [Black09]. Questo approccio richiederebbe troppo tempo per state transition diagram e tabelle delle transizioni di stato più complessi. Dovrebbe quindi essere utilizzato uno strumento per supportare il testing delle transizioni di stato.

Tipi di Difetti
I difetti tipici (si veda anche [Beizer95]) includono:

- Tipi o valori di eventi errati
- Tipi o valori di azioni errati
- Stato iniziale errato
- Impossibilità di raggiungere alcuni stati finali
- Impossibilità di entrare negli stati richiesti
- Stati extra (non necessari)
- Impossibilità di eseguire correttamente alcune transizioni valide
- Possibilità di eseguire transizioni invalide
- Guard condition errate

Durante la creazione del modello di transizione di stato (state transition diagram e tabelle delle transizioni, è possibile rilevare difetti nel documento di specifica. I tipi più comuni di difetti sono le omissioni (cioè, non ci sono informazioni su quello che dovrebbe effettivamente accadere in una determinata situazione) e le contraddizioni.

3.2.5 Tecnica dell'Albero di Classificazione

Gli alberi di classificazione supportano alcune tecniche di test black-box, consentendo la creazione di una rappresentazione grafica dello spazio dati applicabile all'oggetto di test.

I dati sono organizzati in classificazioni e classi come segue:

- Classificazioni: Rappresentano parametri all'interno dello spazio dati per l'oggetto di test, come parametri di input (che possono ulteriormente contenere stati e precondizioni ambientali) e parametri di output. Ad esempio, se un'applicazione può essere configurata in molti modi differenti, le classificazioni potrebbero includere client, browser, lingua e sistema operativo.
- Classi: Ogni classificazione può avere un numero qualsiasi di classi e sottoclassi che descrivono l'occorrenza del parametro. Ogni classe, o partizione di equivalenza, è un valore

specifico all'interno di una classificazione. Nell'esempio precedente, la classificazione della lingua potrebbe includere le partizioni di equivalenza per la lingua inglese, francese e spagnola.

Gli alberi di classificazione consentono ai Test Analyst di inserire le combinazioni che ritengono opportune. Questo include, ad esempio, combinazioni pairwise (si veda il paragrafo 3.2.6), combinazioni three-wise e single-wise.

Ulteriori informazioni sull'uso della tecnica dell'albero di classificazione sono fornite in [Bath14] e [Black09].

Applicabilità
La creazione di un albero di classificazione aiuta un Test Analyst a identificare i parametri (classificazioni) e le loro partizioni di equivalenza (classi) di interesse.

Un'ulteriore analisi del diagramma dell'albero di classificazione consente di identificare possibili valori limite e determinate combinazioni di input di particolare interesse o che possono essere non considerati (ad es. perché incompatibili). L'albero di classificazione risultante può quindi essere utilizzato per il partizionamento di equivalenza, l'analisi ai valori limite o il testing pairwise (si veda il paragrafo 3.2.6).

Limitazioni/Difficoltà
All'aumentare della quantità di classificazioni e/o classi, il diagramma diventa più grande e meno facile da usare. Inoltre, la tecnica dell'albero di classificazione non crea test case completi, ma soltanto combinazioni di dati di test. I Test Analyst devono fornire i risultati per ogni combinazione di test, per creare test case completi.

Copertura
I test case possono essere progettati per raggiungere, ad esempio, una copertura minima delle classi (cioè, tutti i valori in una classificazione testati almeno una volta). Il Test Analyst può anche decidere di coprire le combinazioni pairwise o utilizzare altri tipi di testing combinatorio, ad esempio three-wise.

Tipi di Difetti
I tipi di difetti rilevati dipendono dalla tecnica supportata dagli alberi di classificazione (cioè il partizionamento di equivalenza, l'analisi ai valori limite o il testing pairwise).

3.2.6 Testing Pairwise

Il testing pairwise viene utilizzato quando il software deve essere testato combinando diversi parametri di input, ognuno con diversi valori possibili, dando origine a un numero di combinazioni maggiore di quelle che è possibile testare nel tempo consentito. I parametri di input possono essere indipendenti tra loro: qualsiasi opzione di qualsiasi fattore (cioè qualsiasi valore selezionato per ogni singolo parametro di input) può essere combinata con qualsiasi opzione di qualsiasi altro fattore. La combinazione di un parametro specifico (variabile o fattore) con un valore specifico di quel parametro viene chiamata coppia (pair) parametro-valore (ad esempio, se "colore" è un parametro con sette valori consentiti incluso il "rosso", allora "colore = rosso" potrebbe essere una coppia parametro-valore).

Il testing pairwise utilizza le tecniche combinatoriali per garantire che ogni coppia parametro-valore venga testata una sola volta rispetto ad ogni coppia parametro-valore di ogni altro parametro (cioè, vengano testate tutte le coppie "all-pair" delle coppie parametro-valore di due qualsiasi dei differenti parametri), evitando di testare tutte le combinazioni di coppie parametro-valore. Se il Test Analyst utilizza un approccio manuale, viene costruita una tabella con i test case rappresentati dalle righe e con una colonna per ogni parametro. Il Test Analyst riempirà quindi la tabella con valori tali che tutte le coppie di valori possano essere identificate nella tabella (si veda [Black09]). Qualsiasi elemento che

viene lasciato vuoto nella tabella può essere riempito dal Test Analyst con valori, identificati utilizzando le proprie conoscenze di dominio.

Esistono strumenti per aiutare un Test Analyst in questa attività (si veda www.pairwise.org per alcuni esempi). Richiedono, come input, una lista dei parametri e dei loro valori, e generano un insieme adeguato di combinazioni di valori da ogni parametro, che permette di coprire tutte le coppie parametro-valore. L'output dello strumento può essere utilizzato come input per i test case. Si noti che il Test Analyst deve fornire i risultati attesi per ogni combinazione creata dagli strumenti.

Gli alberi di classificazione (si veda il paragrafo 3.2.5) vengono spesso utilizzati in combinazione con il testing pairwise [Bath14]. La progettazione dell'albero di classificazione è supportata da strumenti e consente di visualizzare le combinazioni dei parametri e dei loro valori (alcuni strumenti offrono anche un enhancement del pairwise). Questo aiuta a identificare le seguenti informazioni:

- Gli input da utilizzare con la tecnica di test pairwise.
- Particolari combinazioni di interesse (ad es. quelle frequentemente utilizzate o una sorgente comune di difetti).
- Combinazioni particolari che sono incompatibili. Questo non assume che i fattori combinati non si influenzeranno reciprocamente; potrebbero influenzarsi reciprocamente ma dovrebbero influenzarsi in modo accettabile.
- Relazioni logiche tra variabili. Ad esempio, "se var1 = x, allora necessariamente var2 non può essere y". Gli alberi di classificazione che catturano queste relazioni sono chiamati "modelli funzionali" (feature model).

Applicabilità
Il problema di avere troppe combinazioni di valori dei parametri si manifesta in almeno due diverse situazioni relative al testing. Alcuni elementi di test coinvolgono diversi parametri, ciascuno con un numero di valori possibili, ad esempio una schermata con diversi campi di input. In questo caso, le combinazioni dei valori dei parametri costituiscono i dati di input per i test case. Inoltre, alcuni sistemi possono essere configurabili in numero di dimensioni, definendo uno spazio di configurazione potenzialmente ampio. In entrambe queste situazioni, il testing pairwise può essere utilizzato per identificare un sottoinsieme di combinazioni, che è gestibile e fattibile.

Per i parametri con molti valori, può essere applicato il partizionamento di equivalenza, o qualche altro meccanismo di selezione, a ogni singolo parametro per ridurre il numero di valori di ogni parametro, prima di utilizzare il testing pairwise per ridurre l'insieme delle combinazioni risultanti. Catturare i parametri e i loro valori in un albero di classificazione supporta questa attività.

Queste tecniche sono generalmente applicate ai livelli di test di integrazione dei componenti, test di sistema e test di integrazione dei sistemi.

Limitazioni/Difficoltà
La limitazione principale di queste tecniche è l'assunzione che i risultati di pochi test siano rappresentativi di tutti i test, e che quei pochi test rappresentino l'utilizzo atteso. Se esiste un'interazione imprevista tra determinate variabili, può non essere rilevata con questa tecnica di test se quella particolare combinazione non viene testata. Queste tecniche possono essere difficili da spiegare a un pubblico non tecnico, in quanto possono non comprendere la riduzione logica dei test. Qualsiasi spiegazione dovrebbe essere bilanciata riportando i risultati di studi empirici [Kuhn16], che hanno dimostrato che nell'area dei dispositivi medici oggetto di studio, il 66% dei failure è stato attivato da una singola variabile, e il 97% da una o due variabili che interagivano tra loro. Esiste un rischio residuo che il testing pairwise non possa rilevare failure di sistema dove interagiscono tre o più variabili.

Individuare i parametri e i rispettivi valori è talvolta difficile, quindi questa attività dovrebbe essere eseguita con il supporto degli alberi di classificazione, ove possibile (si veda il paragrafo 3.2.5). E difficile trovare manualmente un insieme minimo di combinazioni che soddisfi un certo livello di copertura.

Possono essere usati degli strumenti per trovare il più piccolo insieme possibile di combinazioni. Alcuni strumenti supportano la capacità di forzare alcune combinazioni da includere o escludere dalla selezione finale delle combinazioni. Un Test Analyst può utilizzare questa capacità per enfatizzare o de-enfatizzare i fattori in base alla conoscenza del dominio o alle informazioni sull'utilizzo del prodotto.

Copertura
La copertura pairwise al 100% richiede che ogni coppia di valori di qualsiasi coppia di parametri sia inclusa in almeno una combinazione.

Tipi di Difetti
I tipi più comuni di difetti rilevati con questo tipo di tecnica di test sono quelli relativi ai valori combinati di due parametri.

3.2.7 Testing degli Use Case

Il testing degli use case fornisce test transazionali basati su scenari (scenario-based), che dovrebbero emulare l'uso previsto del componente o del sistema specificato dallo use case. Gli use case sono definiti in termini di interazioni tra gli attori e un componente o sistema che raggiunge un determinato obiettivo. Gli attori possono essere utenti umani, hardware esterni, oppure altri componenti o sistemi.

Un comune standard per gli use case è definito in [OMG-UML].

Applicabilità
Il testing degli use case viene generalmente applicato nei livelli di test di sistema e di accettazione. Può essere anche utilizzato nel testing di integrazione se il comportamento dei componenti o dei sistemi è specificato dagli use case. Gli use case sono spesso anche la base per il performance testing, perché descrivono un utilizzo realistico del sistema. Gli scenari descritti negli use case possono essere assegnati ad utenti virtuali per creare un carico realistico sul sistema (purché in essi vengano specificati i requisiti di carico e di prestazione).

Limitazioni/Difficoltà
Per essere validi, gli use case devono specificare transazioni utente realistiche. Le specifiche degli use case (use case specification) sono una forma di progettazione del sistema. I requisiti di quello che gli utenti richiedono di realizzare dovrebbero provenire dagli utenti o dai rappresentanti degli utenti, e dovrebbero essere verificati rispetto ai requisiti dell'organizzazione prima di progettare i corrispondenti use case. Il valore di uno use case viene ridotto se non riflette i reali requisiti dell'utente e dell'organizzazione, o se ostacola piuttosto che agevolare il completamento delle attività dell'utente.

E' importante una definizione accurata dei comportamenti alternativi, di eccezione e di gestione degli errori per avere una copertura accurata. Gli use case dovrebbero essere considerati come linee guida, ma non come una definizione completa di ciò che dovrebbe essere testato, in quanto possono non fornire una definizione chiara dell'intero insieme di requisiti. Può anche essere utile creare altri modelli, come flowchart e/o tabelle delle decisioni, a partire dalla descrizione dello use case, per migliorare l'accuratezza del testing e per verificare lo use case stesso. Come con altre forme di specifica, è probabile che questo riveli anomalie logiche nelle specifiche dello use case, se presenti.

Copertura
Il livello minimo accettabile di copertura di uno use case è di avere un test case per il comportamento principale e test case aggiuntivi sufficienti a coprire ogni comportamento alternativo e di gestione degli errori. Se è richiesto una test suite minima, è possibile incorporare più comportamenti alternativi in un test case, a condizione che siano reciprocamente compatibili. Se è richiesta una migliore capacità diagnostica (ad es. per aiutare a isolare i difetti), è possibile progettare un test case aggiuntivo per ogni comportamento alternativo, sebbene i comportamenti alternativi nidificati richiederanno anche che alcuni di questi comportamenti siano combinati in un singolo test case (ad esempio, comportamenti

alternativi che terminano rispetto a quelli che non terminano all'interno di un comportamento di eccezione "riprova").

Tipi di Difetti
I difetti includono la cattiva gestione di comportamenti definiti, comportamenti alternativi mancanti, elaborazione errata delle condizioni descritte, e messaggi di errore non corretti o implementati in modo insufficiente.

3.2.8 Combinazione delle Tecniche

A volte le tecniche vengono combinate per creare test case. Ad esempio, le condizioni identificate in una tabella delle decisioni potrebbero essere soggette al partizionamento di equivalenza per scoprire molteplici modi in cui una condizione potrebbe essere soddisfatta. I test case coprirebbero quindi non solo ogni combinazione di condizioni, ma per quelle condizioni che sono state partizionate dovrebbero essere creati test case aggiuntivi per coprire le partizioni di equivalenza. Quando si seleziona la particolare tecnica da applicare, il Test Analyst dovrebbe considerare l'applicabilità della tecnica, le limitazioni e le difficoltà, e gli obiettivi del testing in termini di copertura e di difetti da rilevare. Questi aspetti sono descritti per le singole tecniche trattate in questo capitolo. Può non esistere un'unica tecnica "migliore" per una situazione. Le tecniche combinate forniranno spesso la copertura più completa, assumendo che ci siano tempo e abilità sufficienti per applicare correttamente le tecniche.

3.3 Tecniche di Test Basate sull'Esperienza

Il testing basato sull'esperienza utilizza la competenza e l'intuizione dei tester, insieme alla loro esperienza in applicazioni o tecnologie similari per indirizzare il testing, con lo scopo di aumentare il rilevamento dei difetti. Queste tecniche di test possono essere di diverse tipologie: da "test rapidi" (quick test) in cui il tester non deve eseguire attività formalmente pre-pianificate, a sessioni di test pre-pianificate utilizzando test charter, fino a sessioni di test guidate da test script. Sono quasi sempre utili, ma hanno un valore particolare quando possono essere raggiunti alcuni dei vantaggi inclusi nella seguente lista.

Il testing basato sull'esperienza presenta i seguenti vantaggi:
- Può essere una buona alternativa agli approcci più strutturati quando la documentazione del sistema è carente.
- Può essere applicato quando il tempo per il testing è molto stringente.
- Consente di applicare nel testing le competenze sul dominio e sulla tecnologia. Questo può includere coloro che non sono coinvolti nel testing, come business analyst o clienti.
- Può fornire un feedback anticipato agli sviluppatori.
- Aiuta il team a familiarizzare con il software man mano che viene prodotto.
- È efficace quando vengono analizzati failure operativi.
- Consente di applicare una varietà di tecniche di test.

Il testing basato sull'esperienza presenta i seguenti svantaggi:
- Può essere non appropriato nei sistemi che richiedono una documentazione di test dettagliata.
- È difficile raggiungere livelli elevati di ripetibilità.
- La capacità di valutare con precisione la copertura è limitata.
- I test sono meno adatti per essere automatizzati.

Quando si utilizzano approcci reattivi ed euristici, i tester normalmente utilizzano il testing basato sull'esperienza, che è più reattivo agli eventi rispetto agli approcci di test pre-pianificati. Inoltre, l'esecuzione e la valutazione sono attività concorrenti. Alcuni approcci strutturati al testing basato sull'esperienza non sono completamente dinamici, ovvero i test non vengono creati in modo completo

nello stesso momento in cui il tester esegue il test. Questo potrebbe essere il caso, ad esempio, in cui viene utilizzata la tecnica di error guessing per indirizzare particolari aspetti dell'oggetto di test prima dell'esecuzione dei test.

Si noti che sebbene per queste tecniche vengano indicate alcune idee sulla copertura, le tecniche di test basate sull'esperienza non hanno criteri di copertura formali.

3.3.1 Error Guessing

Quando si utilizza la tecnica di error guessing, un Test Analyst utilizza l'esperienza per supporre i potenziali errori che potrebbero essere stati commessi durante la progettazione e lo sviluppo del codice. Quando gli errori previsti sono stati identificati, un Test Analyst determina i metodi migliori da utilizzare per scoprire i difetti risultanti. Ad esempio, se un Test Analyst prevede che il software presenti dei failure quando viene inserita una password invalida, i test saranno eseguiti per inserire una varietà di valori diversi nel campo della password, in modo da verificare se l'errore sia stato effettivamente inserito e abbia generato un difetto che può essere visibile come failure durante l'esecuzione dei test.

Oltre ad essere utilizzata come tecnica di test, l'error guessing è utile anche durante l'analisi del rischio, per identificare potenziali failure. [Myers11]

Applicabilità
L'error guessing viene applicato principalmente durante il testing di sistema e di integrazione, ma può essere applicato a qualsiasi livello di test. Questa tecnica viene spesso utilizzata con altre tecniche, e aiuta ad ampliare l'ambito dei test case esistenti. L'error guessing può anche essere utilizzato in modo efficace durante il testing di una nuova release del software, per verificare la presenza di difetti comuni prima di iniziare il testing più rigoroso e guidato dai test script.

Limitazioni/Difficoltà
Le seguenti limitazioni e difficoltà si applicano all'error guessing:

- La copertura è difficile da valutare e varia ampiamente in base alla capacità e all'esperienza del Test Analyst.
- È utilizzato al meglio da un tester esperto, che abbia familiarità con i tipi di difetti comunemente introdotti nel tipo di codice da testare.
- È usato comunemente, ma spesso non è documentato e quindi può essere meno riproducibile rispetto ad altre forme di test.
- I test case possono essere documentati ma in un modo che solo l'autore comprende e può riprodurre.

Copertura
Quando viene utilizzata una tassonomia dei difetti, la copertura viene determinata considerando il numero di elementi della tassonomia testati diviso per il numero totale di elementi della tassonomia e indicando la copertura in percentuale. Senza una tassonomia dei difetti, la copertura è limitata all'esperienza e alla conoscenza del tester, e al tempo a disposizione. La quantità di difetti rilevati da questa tecnica varia in base alla capacità del tester di individuare le aree problematiche.

Tipi di Difetti
I difetti tipici sono generalmente quelli definiti nella particolare tassonomia dei difetti o che sono stati "indovinati" ("guessed") dal Test Analyst, che potrebbero non essere stati trovati durante il testing black-box.

3.3.2 Testing Checklist-Based

Quando si applica la tecnica di test checklist-based (basato su checklist), un Test Analyst con esperienza utilizza una lista generalizzata e di alto livello di elementi da annotare, verificare o ricordare,

oppure un insieme di regole o criteri rispetto ai quali deve essere verificato un oggetto di test. Queste checklist sono costruite sulla base di una serie di standard, esperienze e altre considerazioni. Ad esempio, una checklist standard della user interface può essere utilizzata come base per testare un'applicazione. Nello sviluppo software Agile, è possibile creare checklist in base ai criteri di accettazione di una user story.

Applicabilità
Il testing checklist-based è più efficace nei progetti con un team di test esperto che ha familiarità con il software sotto test o con l'area coperta dalla checklist (ad esempio, per applicare correttamente una checklist della user interface, il Test Analyst può avere familiarità con il testing della user interface, ma non con il sistema specifico sotto test). Poiché le checklist sono di alto livello e tendono a non riportare i passi dettagliati, che sono comunemente presenti nei test case e nelle procedure di test, la conoscenza del tester viene utilizzata per colmare le lacune. Eliminando i passi dettagliati, le checklist richiedono poca manutenzione e possono essere applicate a più rilasci similari.

Le checklist sono adatte a progetti in cui il software viene rilasciato e modificato rapidamente. Questo aiuta a ridurre sia i tempi di preparazione sia la manutenzione della documentazione di test. Possono essere applicate a qualsiasi livello di test e vengono anche utilizzate per il testing di regressione e lo smoke testing.

Limitazioni/Difficoltà
La natura di alto livello delle checklist può influire sulla riproducibilità dei risultati dei test. È possibile che diversi tester interpretino le checklist in modo diverso e che seguano approcci differenti per coprire gli elementi della checklist. Questo può causare risultati di test diversi, anche se viene utilizzata la stessa checklist. Questo può generare una copertura più ampia, ma a volte viene sacrificata la riproducibilità. Le checklist possono anche creare un'eccessiva fiducia sul livello di copertura raggiunto, poiché il testing effettivo dipende dal giudizio del tester. Le checklist possono derivare da test case o liste più dettagliate, e tendono a crescere nel tempo. E' necessario manutenere le checklist per garantire che coprano gli aspetti importanti del software sotto test.

Copertura
La copertura può essere determinata considerando il numero di elementi della checklist testati diviso il numero totale di elementi della checklist, e indicando la copertura in percentuale. La copertura è buona in base alla qualità della checklist ma, a causa della natura di alto livello della checklist, i risultati varieranno in base al Test Analyst che esegue la checklist.

Tipi di Difetti
I difetti tipici rilevati con questa tecnica generano failure durante l'esecuzione dei test dovuti a variazioni sui dati, sulla sequenza dei passi o sul workflow in generale.

3.3.3 Testing Esplorativo

Il testing esplorativo è caratterizzato dal fatto che il tester contemporaneamente apprende l'oggetto di test e i suoi difetti, pianifica il lavoro di test da eseguire, progetta ed esegue i test, esegue il reporting dei risultati. Il tester adegua dinamicamente gli obiettivi del test durante l'esecuzione e prepara solo documentazione leggera ("lightweight"). [Whittaker09]

Applicabilità
Un buon testing esplorativo è pianificato, interattivo e creativo. Richiede poca documentazione sul sistema da testare ed è spesso usato in situazioni dove la documentazione non è disponibile o non è adeguata per altre tecniche di test. Il testing esplorativo è spesso utilizzato in aggiunta ad altre tecniche di test, e come base per lo sviluppo di test case aggiuntivi. Il testing esplorativo viene frequentemente utilizzato nello sviluppo software Agile per rendere il testing delle user story più flessibile e più veloce,

con una documentazione minimale. Tuttavia, la tecnica può essere applicata anche a progetti che utilizzano un modello di sviluppo sequenziale.

Limitazioni/Difficoltà
La copertura del testing esplorativo può essere sporadica e la riproducibilità di questi test eseguiti può essere difficile. L'uso di test charter per progettare le aree da coprire in una sessione di test e il time-boxing per determinare il tempo consentito per il testing sono tecniche utilizzate per gestire il testing esplorativo. Al termine di una sessione di test o di un insieme di sessioni di test, il Test Manager può svolgere una sessione di debriefing per raccogliere i risultati dei test e determinare i test charter per le sessioni di test successive.

Un'altra difficoltà con le sessioni d testing esplorativo è quella di tracciarle accuratamente in un sistema di test management. A volte questo viene fatto creando test case che sono in realtà sessioni di testing esplorativo. Questo consente di tenere traccia del tempo allocato per il testing esplorativo e per la copertura pianificata insieme agli altri effort del test.

Poiché la riproducibilità può essere difficile da raggiungere con il testing esplorativo, questo può anche causare problemi quando è necessario ricordare i passi per riprodurre un failure. Alcune organizzazioni utilizzano la capacità di cattura/riesecuzione (capture/playback) di uno strumento di esecuzione dei test automatizzati per registrare i passi svolti da un tester esplorativo. Questo fornisce una registrazione completa di tutte le attività durante la sessione di testing esplorativo (o qualsiasi sessione di testing basato sull'esperienza). Analizzare i dettagli per trovare la causa reale di un failure può essere noioso, ma almeno esiste una registrazione di tutti i passi svolti.

Altri strumenti possono essere utilizzati per catturare sessioni di testing esplorativo, ma questi non registrano i risultati attesi, perché non catturano l'interazione con la Graphical User Interface (GUI, Interfaccia Utente). In questo caso, i risultati attesi devono essere annotati in modo che, se necessario, sia possibile effettuare un'adeguata analisi dei difetti. In generale, si raccomanda di prendere nota anche durante l'esecuzione del testing esplorativo per supportare la riproducibilità, ove richiesto.

Copertura
I test charter possono essere progettati per specifiche attività, obiettivi e rilasci. Sono quindi pianificate sessioni di testing esplorativo per raggiungere tali criteri. Il test charter può anche identificare dove focalizzare l'effort del test, cosa è in ambito (in scope) o fuori ambito (out of scope) nella sessione di test, e quali risorse dovrebbero essere utilizzate per completare i test pianificati. Una sessione di test può essere utilizzata per focalizzarsi su particolari tipi di difetti e altre aree potenzialmente problematiche, che possono essere indirizzate senza la formalità del testing guidato da test script.

Tipi di Difetti
I difetti tipici rilevati con il testing esplorativo sono i problemi scenario-based che vengono dimenticati durante il testing di adeguatezza funzionale guidato da test script, problemi che accadono tra i limiti funzionali e problemi relativi al workflow. Talvolta durante il testing esplorativo vengono rilevati anche problemi di prestazione e di sicurezza.

3.3.4 Tecnica di Test Basata sui Difetti

Una tecnica di test basata sui difetti è quella in cui il tipo di difetto ricercato viene utilizzato come base per la progettazione dei test, con i test derivati sistematicamente in base a quello che è conosciuto sul tipo di difetto. A differenza del testing black-box che deriva i test dalla base di test, il testing basato sui difetti deriva i test da liste che si focalizzano sui difetti. In generale, le liste possono essere organizzate in tipi di difetti, root cause, sintomi di failure e altri dati relativi ai difetti. Le liste standard si applicano a molti tipi di software e non sono specifiche del prodotto. L'uso di queste liste aiuta a sfruttare le conoscenze standard del settore per derivare test particolari. Aderendo a liste specifiche del settore, le metriche relative al verificarsi dei difetti possono essere tracciate tra diversi progetti e persino tra diverse

organizzazioni. Le liste di difetti più comuni sono quelle specifiche dell'organizzazione o del progetto, e fanno uso di competenze ed esperienze specifiche.

Il testing basato sui difetti può anche utilizzare liste di rischi identificati e scenari di rischio come base per indirizzare il testing. Questa tecnica di test consente a un Test Analyst di indirizzare un tipo specifico di difetto o di lavorare sistematicamente utilizzando una lista di difetti noti e comuni di un particolare tipo. Da queste informazioni, il Test Analyst crea le condizioni di test e i test case che permetteranno al difetto di manifestarsi (se esiste).

Applicabilità
Il testing basato sui difetti può essere applicato a qualsiasi livello di test, ma viene generalmente applicato durante il testing di sistema.

Limitazioni/Difficoltà
Esistono diverse tassonomie dei difetti e possono focalizzarsi su particolari tipi di test, come l'usabilità. È importante scegliere una tassonomia che sia applicabile al software sotto test (se disponibile). Ad esempio, possono non esistere tassonomie disponibili per software innovativi. Alcune organizzazioni hanno definito tassonomie custom di difetti probabili o rilevati di frequente. Qualunque sia la tassonomia dei difetti utilizzata, è importante definire la copertura attesa prima di iniziare il testing.

Copertura
La tecnica fornisce criteri di copertura che vengono utilizzati per determinare se sono stati identificati tutti i test case utili. Gli elementi di copertura possono essere elementi strutturali, elementi di specifica, scenari di utilizzo o qualsiasi combinazione di questi, a seconda della lista dei difetti. In pratica, i criteri di copertura per le tecniche di test basate sui difetti tendono ad essere meno sistematici rispetto ai criteri per le tecniche di test black-box, in quanto vengono fornite solo le regole generali per la copertura e la decisione specifica sul limite di copertura utile è opzionale. Come con altre tecniche, il raggiungimento dei criteri di copertura non significa che l'insieme di test sia completo, ma piuttosto che i difetti considerati non suggeriscono più alcun test utile basato su quella tecnica.

Tipi di Difetti
I tipi di difetti rilevati di solito dipendono dalla tassonomia dei difetti in uso. Ad esempio, se viene utilizzata una lista di difetti della user interface, la maggior parte dei difetti rilevati sarebbero probabilmente correlati alla user interface, ma altri difetti possono essere rilevati come sottoprodotto del testing specifico.

3.4 Applicazione della Tecnica più Appropriata

Le tecniche di test black-box e le tecniche di test basate sull'esperienza sono più efficaci quando utilizzate insieme. Le tecniche di test basate sull'esperienza colmano le lacune nella copertura dovute ad eventuali debolezze sistematiche nelle tecniche di test black-box.

Non esiste una tecnica perfetta per tutte le situazioni. È importante che il Test Analyst comprenda i vantaggi e gli svantaggi di ogni tecnica, e sia in grado di selezionare la migliore tecnica o insieme di tecniche per la situazione, considerando il tipo di progetto, la schedulazione, l'accesso alle informazioni, le competenze del tester e altri fattori che possono influenzare la selezione.

Nella discussione di ogni tecnica di test black-box e basata sull'esperienza (si vedano rispettivamente i paragrafi 3.2 e 3.3), le informazioni fornite nei paragrafi "Applicabilità", "Limitazioni/Difficoltà" e "Copertura" guidano un Test Analyst nella selezione delle tecniche di test più appropriate da applicare

4. Testing delle Caratteristiche di Qualità del Software - 180 min.

Parole Chiave

accessibilità, adeguatezza funzionale, apprendibilità, appropriatezza funzionale, compatibilità, completezza funzionale, correttezza funzionale, estetica della user interface, interoperabilità, operabilità, Software Usability Measurement Inventory (SUMI) , protezione da errori dell'utente, usabilità, User Experience (UX), Website Analysis and Measurement Inventory (WAMMI)

Obiettivi di Apprendimento per il Testing delle Caratteristiche di Qualità del Software

4.1 Introduzione

Nessun Obiettivo di Apprendimento

4.2 Caratteristiche di Qualità per il Testing del Dominio di Business

TA-4.2.1 (K2) Spiegare quali tecniche di test sono appropriate per testare la completezza funzionale, la correttezza funzionale e l'appropriatezza funzionale

TA-4.2.2 (K2) Definire i difetti tipici da individuare per le caratteristiche di completezza funzionale, correttezza funzionale e appropriatezza funzionale

TA-4.2.3 (K2) Definire quando dovrebbero essere testate le caratteristiche di completezza funzionale, correttezza funzionale e appropriatezza funzionale nel ciclo di vita dello sviluppo software

TA-4.2.4 (K2) Spiegare gli approcci che sarebbero adeguati per verificare e validare sia l'implementazione dei requisiti di usabilità, sia la soddisfazione delle aspettative dell'utente

TA-4.2.5 (K2) Spiegare il ruolo del Test Analyst nel testing di interoperabilità, compresa l'identificazione dei difetti da individuare

TA-4.2.6 (K2) Spiegare il ruolo del Test Analyst nel testing di portabilità, compresa l'identificazione dei difetti da individuare

TA-4.2.7 (K4) Per un determinato insieme di requisiti, determinare le condizioni di test richieste per verificare le caratteristiche di qualità funzionali e/o non funzionali nell'ambito del Test Analyst

4.1 Introduzione

Il capitolo precedente ha descritto tecniche specifiche di test disponibili per il tester; questo capitolo invece prende in considerazione l'applicazione di tali tecniche nella valutazione delle caratteristiche utilizzate per descrivere la qualità delle applicazioni o dei sistemi software.

Questo Syllabus descrive le caratteristiche di qualità che possono essere valutate da un Test Analyst. Gli attributi che devono essere valutati dal Technical Test Analyst sono considerati nel Syllabus Advanced Technical Test Analyst [CTAL-TTA].

La descrizione delle caratteristiche di qualità del prodotto fornita dallo standard ISO 25010 [ISO25010] viene utilizzata come guida per descrivere le caratteristiche. Il modello di qualità del software ISO suddivide la qualità del prodotto in diverse caratteristiche di qualità del prodotto, ognuna delle quali può avere delle sotto-caratteristiche. Queste sono elencate nella tabella seguente, insieme a un'indicazione di quali caratteristiche/sotto-caratteristiche vengono coperte dai Syllabi Test Analyst e Technical Test Analyst:

Caratteristica	Sotto-Caratteristiche	Test Analyst	Technical Test Analyst
Adeguatezza funzionale	Correttezza funzionale, appropriatezza funzionale, completezza funzionale	X	
Affidabilità	Maturità, tolleranza ai guasti, recuperabilità, disponibilità		X
Usabilità	Riconoscibilità dell'appropriatezza, apprendibilità, operabilità, estetica della user interface, protezione da errori dell'utente, accessibilità	X	
Efficienza delle prestazioni	Comportamento nel tempo, utilizzo delle risorse, capacità		X
Manutenibilità	Analizzabilità, modificabilità, testabilità, modularità, riutilizzabilità		X
Portabilità	Adattabilità, installabilità, sostituibilità	X	X
Sicurezza	Confidenzialità, integrità, non-repudiation, responsabilità, autenticità		X
Compatibilità	Coesistenza		X
	Interoperabilità	X	

Questa allocazione delle attività tra i due ruoli può variare nelle diverse organizzazioni, ma è l'unica che viene seguita nei Syllabi ISTQB® associati.

Per tutte le caratteristiche e sotto-caratteristiche di qualità discusse in questo capitolo, i rischi tipici devono essere riconosciuti in modo tale da poter definire e documentare una strategia di test appropriata. Il testing delle caratteristiche di qualità richiede un'attenzione particolare alle tempistiche del SDLC, agli strumenti richiesti, alla disponibilità del software e della documentazione, e alla competenza tecnica. Senza una strategia per affrontare ogni caratteristica e le relative esigenze specifiche del testing, il tester può non avere nella schedulazione un tempo adeguato per la pianificazione, la preparazione e l'esecuzione dei test [Bath14]. Alcuni di questi test, ad esempio il testing di usabilità, possono richiedere l'allocazione di risorse umane speciali, una pianificazione estesa, laboratori dedicati, strumenti specifici, competenze di test specializzate e, in molti casi, un periodo di tempo significativo. In alcuni casi, il testing di usabilità può essere eseguito da un gruppo separato di esperti di usabilità o di User Experience (UX).

Anche se il Test Analyst può non essere responsabile delle caratteristiche di qualità che richiedono un approccio più tecnico, è importante che il Test Analyst sia a conoscenza delle altre caratteristiche e comprenda le aree che si sovrappongono per il testing. Ad esempio, un oggetto di test che fallisce durante il performance testing può fallire anche durante il testing di usabilità se è troppo lento per consentire all'utente di utilizzarlo in modo efficace. Allo stesso modo, un oggetto di test con problemi di interoperabilità con alcuni componenti non è probabilmente pronto per il testing di portabilità, poiché tenderà a nascondere i problemi più elementari che si presentano quando l'ambiente viene modificato.

4.2 Caratteristiche di Qualità per il Testing del Dominio di Business

Il testing di adeguatezza funzionale è un obiettivo primario per il Test Analyst. Il testing di adeguatezza funzionale è focalizzato su "cosa" fa l'oggetto di test. La base di test per il testing di adeguatezza funzionale sono generalmente i requisiti, le specifiche, le competenze specifiche del dominio o le necessità implicite. I test di adeguatezza funzionale variano a seconda del livello di test in cui vengono eseguiti, e possono anche essere influenzati dal SDLC. Ad esempio, un test di adeguatezza funzionale svolto durante il testing di integrazione verificherà l'adeguatezza funzionale dei componenti di interfaccia che implementano una singola funzione definita. A livello di test di sistema, i test di adeguatezza funzionale includono il testing dell'adeguatezza funzionale del sistema nel suo insieme. Per i sistemi di sistemi, il testing di adeguatezza funzionale si focalizzerà principalmente sul testing end-to-end tra i sistemi integrati. Una vasta gamma di tecniche di test viene applicata durante il testing di adeguatezza funzionale (si veda il capitolo 3).

In uno sviluppo software Agile, il testing di adeguatezza funzionale di solito include quanto segue:

- Testing di una funzionalità specifica (ad es. user story) pianificata per l'implementazione nella particolare iterazione
- Regression Testing di tutte le funzionalità non modificate

Oltre al testing di adeguatezza funzionale trattato in questo paragrafo, esistono anche alcune caratteristiche di qualità che sono parte dell'area sotto la responsabilità del Test Analyst e che sono considerate aree di test non funzionali (focalizzate sul "come" l'oggetto di test esegue la funzionalità).

4.2.1 Testing di Correttezza Funzionale

La correttezza funzionale implica la verifica dell'aderenza dell'applicazione ai requisiti specificati o impliciti, e può anche includere l'accuratezza computazionale. Il testing di correttezza funzionale utilizza molte delle tecniche di test spiegate nel Capitolo 3, e spesso utilizza le specifiche o un sistema legacy come oracolo del test. Il testing di correttezza funzionale può essere svolto a qualsiasi livello di test e ha lo scopo di rilevare difetti relativi alla gestione errata di dati o di situazioni.

4.2.2 Testing di Appropriatezza Funzionale

Il testing di appropriatezza funzionale implica la valutazione e la validazione dell'appropriatezza di un insieme di funzioni per le relative attività specifiche previste. Questo testing può essere basato sulla progettazione funzionale (ad es. use case e/o user story). Il testing di appropriatezza funzionale viene generalmente svolto durante il testing di sistema, ma può anche essere svolto durante le fasi successive del testing di integrazione. I difetti che possono essere rilevati indicano che il sistema non sarà in grado di soddisfare le esigenze dell'utente in un modo da poter essere considerato accettabile.

4.2.3 Testing di Completezza Funzionale

Il testing di completezza funzionale viene eseguito per determinare la copertura di attività specifiche e di obiettivi utente rispetto alla funzionalità implementata. La tracciabilità tra gli elementi della specifica (ad es. requisiti, user story, use case) e la funzionalità implementata (ad es. funzione, componente, workflow) è fondamentale per consentire di determinare la completezza funzionale richiesta. La

valutazione della completezza funzionale può variare in base al particolare livello di test e/o al SDLC utilizzato. Ad esempio, la completezza funzionale di uno sviluppo software Agile può essere basata su user story e funzionalità implementate. La completezza funzionale per il testing di integrazione di sistemi può focalizzarsi sulla copertura dei processi di business di alto livello.

Se il Test Analyst mantiene la tracciabilità tra i test case e gli elementi di specifica funzionale, la valutazione della completezza funzionale è generalmente supportata da strumenti di test management. Livelli di completezza funzionale inferiori alle aspettative indicano che il sistema non è stato completamente implementato.

4.2.4 Testing di Interoperabilità

Il testing di interoperabilità verifica lo scambio di informazioni tra due o più sistemi o componenti. I test si focalizzano sulla capacità di scambiare informazioni e successivamente di utilizzare le informazioni che sono state scambiate. Il testing dovrebbe coprire tutti gli ambienti target previsti (comprese le variazioni nel software, hardware, middleware, sistema operativo, ecc.) per garantire che lo scambio dati funzionerà correttamente. In realtà, questo può essere fattibile solo per un numero relativamente piccolo di ambienti. In tal caso, il testing di interoperabilità può essere limitato a un selezionato gruppo rappresentativo degli ambienti. La specifica dei test per l'interoperabilità richiede che le combinazioni degli ambienti target previsti siano identificate, configurate e disponibili per il team di test. Questi ambienti vengono quindi testati utilizzando una selezione di test case di adeguatezza funzionale, che esercitano i vari punti di scambio dati presenti nell'ambiente.

L'interoperabilità riguarda il modo in cui diversi componenti e sistemi software interagiscono tra loro. Un software con buone caratteristiche di interoperabilità può essere integrato con una serie di altri sistemi senza richiedere grandi modifiche o un impatto significativo sul comportamento non funzionale. Il numero di modifiche e l'effort richiesto per implementare e testare tali modifiche possono essere utilizzati come misura di interoperabilità.

Il testing per l'interoperabilità del software può, ad esempio, focalizzarsi sulle seguenti caratteristiche di progettazione:

- Uso di standard di comunicazione a livello di settore, come XML
- Capacità di rilevare automaticamente le esigenze di comunicazione dei sistemi con cui interagisce e adattarsi di conseguenza

Il testing di interoperabilità può essere particolarmente significativo per:

- Strumenti e prodotti software commercial off-the-shelf
- Applicazioni basate su un sistema di sistemi
- Sistemi basati su Internet of Things
- Servizi web con connettività ad altri sistemi

Questo tipo di test viene eseguito durante il testing di integrazione dei componenti e il testing di integrazione di sistemi. A livello di integrazione di sistemi, questo tipo di test viene eseguito per determinare come bene il sistema, completamente sviluppato, interagisce con altri sistemi. Poiché i sistemi possono interagire su più livelli, il Test Analyst deve comprendere queste interazioni ed essere in grado di creare le condizioni che eserciteranno le varie interazioni. Ad esempio, se due sistemi si scambieranno dati, il Test Analyst deve essere in grado di creare i dati e le transazioni necessarie per eseguire tale scambio. È importante ricordare che non tutte le interazioni possono essere chiaramente specificate nei documenti dei requisiti. Al contrario, molte di queste interazioni saranno definite solo nei documenti di architettura e di progettazione di sistema. Il Test Analyst deve avere le capacità ed essere preparato a esaminare questi documenti per determinare i punti di scambio di informazione tra i sistemi, e tra il sistema e il suo ambiente, per garantire che siano tutti testati. Tecniche come il partizionamento di equivalenza, l'analisi ai valori limite, il testing della tabella delle decisioni, il testing delle transizioni di

stato, il testing degli use case e il testing pairwise, sono tutte applicabili al testing di interoperabilità. I difetti tipici rilevati includono lo scambio errato dei dati tra i componenti che interagiscono.

4.2.5 Valutazione dell'Usabilità

I Test Analyst si trovano spesso nella posizione di dover coordinare e supportare la valutazione dell'usabilità. Questo può richiedere di dover specificare i test di usabilità o di avere un ruolo di moderatore che collabora con gli utenti per eseguire i test. Per fare questo in modo efficace, un Test Analyst deve comprendere i principali aspetti, gli obiettivi e gli approcci coinvolti in questi tipi di test. Per ulteriori dettagli rispetto a quanto descritto in questo paragrafo, fare riferimento al Syllabus ISTQB® Foundation Level Specialist Syllabus Usability Testing [ISTQB_UT_SYL].

È importante capire perché gli utenti potrebbero avere difficoltà a utilizzare il sistema o perché non hanno una User Experience (UX) positiva (ad es. utilizzando software per intrattenimento). Per raggiungere questa comprensione è necessario innanzitutto comprendere che il termine "utente" può essere applicato a una vasta gamma di diverse tipologie di persone, che vanno dagli esperti IT ai bambini, alle persone con disabilità.

4.2.5.1 Aspetti di Usabilità

I seguenti sono i tre aspetti considerati in questo paragrafo:

- Usabilità
- User Experience (UX)
- Accessibilità

Usabilità

Il testing di usabilità si concentra sui difetti del software, che impattano la capacità di un utente di eseguire attività tramite la user interface. Tali difetti possono incidere sulla capacità dell'utente di raggiungere i propri obiettivi in modo efficace, efficiente o con soddisfazione. I problemi di usabilità possono portare a confusione, errore, ritardo o totale insuccesso nel completare alcune attività da parte dell'utente.

Di seguito sono riportate le sotto-caratteristiche di usabilità secondo lo standard ISO 25010 [ISO 25010]; per le loro definizioni si veda [ISTQB_GLOSSARY]:

- Riconoscibilità dell'appropriatezza (o comprensibilità)
- Apprendibilità
- Operabilità
- Estetica della user interface (o attrattività)
- Protezione da errori dell'utente
- Accessibilità (si veda di seguito)

User Experience (UX)

La valutazione della User Experience indirizza l'esperienza utente (user experience) completa dell'utente con l'oggetto di test, non solo l'interazione diretta. Questa è di particolare importanza per gli oggetti di test dove i fattori come il divertimento e la soddisfazione dell'utente sono critici per il successo del business.

I fattori tipici che influenzano la User Experience includono quanto segue:

- Brand del marchio (cioè la fiducia dell'utente nel produttore)
- Comportamento interattivo
- Disponibilità dell'oggetto di test, incluso il sistema di help, il supporto e la formazione

Accessibilità

È importante considerare l'accessibilità al software per coloro che hanno particolari esigenze o restrizioni per il suo utilizzo, includendo anche quelle relative alla disabilità. Il testing di accessibilità

dovrebbe prendere in considerazione gli standard rilevanti, come Web Content Accessibility Guidelines (WCAG) e le normative, come Disability Discrimination Acts (Irlanda del Nord, Australia), Equality Act 2010 (Inghilterra, Scozia, Galles) e Section 508 (USA). L'accessibilità deve essere considerata, in modo similare all'usabilità, quando si conducono attività di progettazione. Il testing si esegue spesso durante i livelli di integrazione e continua durante i livelli del test di sistema e del test di accettazione. Di solito i difetti sono rilevati quando il software non soddisfa normative o standard specifici, definiti per il software.

Le misure tipiche per migliorare l'accessibilità si focalizzano sulle opportunità offerte agli utenti con disabilità di interagire con l'applicazione. Queste includono:

- Utilizzare il riconoscimento vocale per gli input
- Garantire che il contenuto non testuale presentato all'utente abbia un'alternativa testuale equivalente
- Abilitare il ridimensionamento del testo senza perdere di contenuto o funzionalità

Le linee guida per l'accessibilità supportano il Test Analyst, fornendo una sorgente di informazioni e delle checklist che possono essere utilizzate per il testing (alcuni esempi di linee guida per l'accessibilità sono riportati in [ISTQB_UT_SYL]). Inoltre, sono disponibili strumenti e plugin del browser per aiutare i tester a identificare problemi di accessibilità, come ad esempio la scarsa scelta dei colori nelle pagine Web che violano le linee guida per il daltonismo.

4.2.5.2 Approcci per la Valutazione dell'Usabilità

L'usabilità, la User Experience e l'accessibilità possono essere testate applicando uno o più dei seguenti approcci:

- Testing di usabilità
- Review di usabilità
- Sondaggi e questionari di usabilità

Testing di Usabilità

Il testing di usabilità valuta la facilità con cui gli utenti possono utilizzare o imparare a utilizzare il sistema per raggiungere un obiettivo specifico in un contesto specifico. Il testing di usabilità è diretto a misurare quanto segue:

- Efficacia: capacità dell'oggetto di test di consentire agli utenti di raggiungere obiettivi specifici con accuratezza e completezza in uno specifico contesto di utilizzo
- Efficienza: capacità dell'oggetto di test di consentire agli utenti di utilizzare quantità appropriate di risorse in relazione all'efficacia raggiunta in uno specifico contesto di utilizzo
- Soddisfazione: capacità dell'oggetto di test di soddisfare gli utenti in un specifico contesto di utilizzo

È importante notare che la progettazione e la specifica dei test di usabilità viene spesso condotta dal Test Analyst in collaborazione con tester che possiedono abilità speciali nel testing di usabilità e con ingegneri di progettazione dell'usabilità che conoscono il processo di progettazione centrata sulle persone (human-centered design) (si veda [ISTQB_UT_SYL] per ulteriori dettagli).

Review di Usabilità

Le ispezioni e le review sono un tipo di test condotto da una prospettiva dell'usabilità, che aiutano ad aumentare il livello di coinvolgimento dell'utente. Questo può essere efficace a livello economico per rilevare problemi di usabilità nelle specifiche dei requisiti e nelle progettazioni nelle prime fasi di SDLC. La valutazione euristica (ispezione sistematica di una progettazione della user interface per l'usabilità) può essere utilizzata per trovare problemi di usabilità nella progettazione, in modo da per poter essere indirizzati come parte di un processo di progettazione iterativo. Questo comporta la presenza di un piccolo gruppo di valutatori, che esamina l'interfaccia e ne giudica la conformità rispetto ai principi di usabilità riconosciuti (le '"euristiche"). Le review sono più efficaci quando la user interface è visibile. Ad esempio, gli screenshot di esempio sono generalmente più facili da comprendere e interpretare rispetto

alla semplice descrizione della funzionalità fornita per una particolare schermata. La visualizzazione è importante per un'adeguata review di usabilità della documentazione.

Sondaggi e Questionari di Usabilità
Possono essere applicare le tecniche dei sondaggi e dei questionari per raccogliere osservazioni e feedback sul comportamento dell'utente con il sistema. Sondaggi standard e disponibili al pubblico come SUMI (Software Usability Measurement Inventory) e WAMMI (Website Analysis and Measurement Inventory) consentono di effettuare il benchmarking rispetto a un database di precedenti misure di usabilità. Inoltre, poiché SUMI fornisce misure concrete di usabilità, può fornire un insieme di criteri di completamento/accettazione.

4.2.6 Testing di Portabilità

I test di portabilità si riferiscono alla capacità di un componente o sistema software di essere trasferito nel suo ambiente target (di destinazione) previsto, come nuova installazione oppure a partire da un ambiente esistente.

La classificazione ISO 25010 delle caratteristiche di qualità del prodotto include le seguenti sotto-caratteristiche di portabilità:
- Installabilità
- Adattabilità
- Sostituibilità

L'attività di identificare i rischi e progettare i test per le caratteristiche di portabilità viene condivisa tra il Test Analyst e il Technical Test Analyst (si veda [ISTQB_ALTTA_SYL] paragrafo 4.7).

4.2.6.1 Testing di Installabilità
Il testing di installabilità viene condotto sul software; procedure scritte vengono utilizzate per installare e disinstallare il software nel suo ambiente target.

Gli obiettivi di test tipici che sono il focus del Test Analyst includono:
- Validare che differenti configurazioni del software possono essere installate con successo. Laddove è possibile configurare un numero elevato di parametri, il Test Analyst può progettare i test utilizzando la tecnica pairwise, per ridurre il numero di combinazioni di parametri testati, e focalizzarsi su particolari configurazioni di interesse (ad esempio, quelle utilizzate di frequente).
- Testare la correttezza funzionale delle procedure di installazione e disinstallazione.
- Eseguire test di adeguatezza funzionale dopo un'installazione o disinstallazione, per rilevare eventuali difetti che possono essere stati introdotti (ad es. configurazioni errate, funzioni non disponibili).
- Identificare problemi di usabilità nelle procedure di installazione e disinstallazione (ad es. per validare che gli utenti siano forniti di istruzioni comprensibili e di messaggi di feedback/errore durante l'esecuzione della procedura).

4.2.6.2 Testing di Adattabilità
Il testing di adattabilità verifica se una determinata applicazione può essere adattata in modo efficace ed efficiente per funzionare correttamente in tutti gli ambienti target previsti (hardware, software, middleware, sistema operativo, cloud, ecc.). Il Test Analyst supporta il testing di adattabilità identificando gli ambienti target previsti (ad es. versioni di diversi sistemi operativi mobile supportati, diverse versioni dei browser che possono essere utilizzati) e progettando i test che coprono le combinazioni di questi ambienti. Gli ambienti target vengono quindi testati utilizzando una selezione dei test case di adeguatezza funzionale, che esercitano i vari componenti presenti nell'ambiente.

4.2.6.3 Testing di Sostituibilità
Il testing di sostituibilità si focalizza sulla capacità di sostituire componenti software o versioni con altri, all'interno di un sistema. Questo può essere particolarmente rilevante per architetture di sistema basate

su Internet of Things, dove la sostituzione di diversi dispositivi hardware e/o installazioni software è un evento comune. Ad esempio, un dispositivo hardware utilizzato in un magazzino per registrare e controllare i livelli delle scorte può essere sostituito con un dispositivo hardware più avanzato (ad es. con uno scanner migliore), oppure il software installato può essere aggiornato con una nuova versione che consente agli ordini di sostituzione delle scorte di essere automaticamente inviati al sistema di un fornitore.

I test di sostituibilità possono essere eseguiti dal Test Analyst in parallelo ai test di integrazione funzionale, dove sono disponibili più componenti alternativi per l'integrazione nel sistema completo.

5. Review - 120 min.

Parole Chiave
review checklist-based

Obiettivi di Apprendimento per le Review

5.1 Introduzione
Nessun Obiettivo di Apprendimento

5.2 Utilizzo delle Checklist nelle Review
TA-5.2.1 (K3) Identificare i problemi in una specifica dei requisiti in base alle informazioni della checklist fornite nel Syllabus
TA-5.2.2 (K3) Identificare i problemi in una user story in base alle informazioni della checklist fornite nel Syllabus

5.1 Introduzione

I Test Analyst devono essere partecipanti attivi nel processo di review, fornendo i loro punti di vista unici. Quando eseguite in modo appropriato, le review possono essere il contributo più grande, e più conveniente a livello economico, alla qualità complessiva fornita.

5.2 Utilizzo delle Checklist nelle Review

La review checklist-based è la tecnica più comune utilizzata da un Test Analyst durante la review della base di test. Le checklist vengono utilizzate durante le review per ricordare ai partecipanti di controllare durante la review dei punti specifici. Possono anche aiutare a depersonalizzare la review (ad es.: "Questa è la stessa checklist che utilizziamo per ogni review. Non è indirizzata solo sul tuo prodotto di lavoro").

La review checklist-based può essere eseguita in modo generico per tutte le review, o può concentrarsi su particolari caratteristiche di qualità, aree o tipi di documenti. Ad esempio, una checklist generica potrebbe verificare le proprietà generali del documento, come avere un identificativo univoco, nessun riferimento contrassegnato "da determinare", una formattazione appropriata e elementi di conformità similari. Una checklist specifica per un documento dei requisiti potrebbe contenere verifiche sull'uso appropriato dei termini "deve" e "dovrebbe", verifiche della testabilità di ogni requisito dichiarato, e così via.

Il formato dei requisiti può anche indicare il tipo di checklist da utilizzare. Un documento dei requisiti in un formato testuale avrà criteri di review differenti rispetto a un documento basato su modelli visuali.

Le checklist possono anche essere orientate ad un aspetto particolare, come:

- Un insieme di abilità del programmatore/architetto o un insieme di abilità del tester - nel caso del Test Analyst, la checklist dell'insieme di abilità del tester sarebbe quella più appropriata
- Un certo livello di rischio (ad es. nei sistemi safety critical) - le checklist includeranno tipicamente informazioni specifiche richieste per il livello di rischio
- Una tecnica di test specifica - la checklist si focalizzerà sulle informazioni necessarie per una particolare tecnica (ad es. le regole da rappresentare in una tabella delle decisioni)
- Un particolare elemento di specifica, come un requisito, uno use case o una user story - questi sono trattati nei paragrafi seguenti e generalmente hanno un focus diverso rispetto a quelli utilizzati da un Technical Test Analyst per la review del codice o dell'architettura

5.2.1 Review dei Requisiti

I seguenti elementi sono un esempio di cosa potrebbe includere una checklist orientata ai requisiti:

- Sorgente del requisito (ad es. persona, dipartimento)
- Testabilità di ogni requisito
- Priorità di ogni requisito
- Criteri di accettazione per ogni requisito
- Disponibilità di uno use case che descrive la struttura, se applicabile
- Identificativo univoco di ogni requisito/use case/user story
- Versioning di ogni requisito/use case/user story
- Tracciabilità di ogni requisito verso i requisiti di business/marketing
- Tracciabilità tra requisiti e/o use case (se applicabile)
- Uso di una terminologia consistente (ad es. utilizzo di un glossario)

È importante ricordare che se un requisito non è testabile, significa che è stato definito in modo tale che il Test Analyst non è in grado di determinare come testarlo, e quindi esiste un difetto in quel requisito. Ad esempio, il requisito: "Il software deve essere molto intuitivo per l'utente" non è testabile. Come può

il Test Analyst determinare se il software è intuitivo o addirittura molto intuitivo per l'utente? Se invece il requisito viene descritto come: "Il software deve essere conforme agli standard di usabilità indicati nel documento degli standard di usabilità, versione xxx", e il documento degli standard di usabilità esiste, allora questo è un requisito testabile. Ed è anche un requisito generale, poiché si applica a tutti gli elementi nell'interfaccia. In questo caso, questo requisito potrebbe facilmente generare molti singoli test case in un'applicazione non banale. Anche la tracciabilità da questo requisito, o dal documento degli standard di usabilità, verso i test case è critica, perché se la specifica di usabilità di riferimento dovesse cambiare, tutti i test case dovranno essere rivisti e aggiornati in base alle necessità.

Inoltre, un requisito è non testabile se il tester non è in grado di determinare se il test è stato superato o è fallito, oppure se non è in grado di progettare un test che può essere superato o che può fallire. Ad esempio, "Il sistema deve essere disponibile per il 100% del tempo, 24 ore al giorno, 7 giorni alla settimana, 365 (o 366) giorni all'anno" non è testabile.

Una semplice checklist[1] per le review degli use case può includere le seguenti domande:

- Il comportamento principale (cammino) è chiaramente definito?
- Tutti i comportamenti (cammini) alternativi sono stati identificati, sono completi e hanno una gestione degli errori?
- I messaggi della user interface sono definiti?
- Esiste un comportamento principale (dovrebbe esistere, altrimenti, esistono più use case)?
- Ogni comportamento è testabile?

5.2.2 Review delle User Story

In uno sviluppo software Agile, i requisiti di solito assumono la forma di user story. Queste user story rappresentano piccole unità di funzionalità verificabili. Uno use case è una transazione utente che impatta diverse aree di funzionalità, invece una user story è una caratteristica più isolata, ed è generalmente considerata in ambito in base al tempo necessario per svilupparla. Una checklist[1] per una user story potrebbe includere quanto segue:

- La user story è appropriata per l'iterazione/sprint target?
- La user story è scritta dal punto di vista della persona che la richiede?
- I criteri di accettazione sono definiti e testabili?
- La funzionalità è chiaramente definita e specificata?
- La user story è indipendente dalle altre?
- La user story è stata prioritizzata?
- La user story segue il formato comunemente usato:
  Come <tipo utente>, voglio che <obiettivo> di modo che <motivo dell'obiettivo>
  In inglese:
  As a < type of user >, I want < some goal > so that < some reason > [Cohn04]

Se la user story definisce una nuova interfaccia, sarebbe appropriato utilizzare una checklist generica per una user story (come specificato sopra) e una checklist dettagliata per la user interface.

5.2.3 Adattare le Checklist

Una checklist può essere adattata (tailoring) in base a quanto segue:

- Organizzazione (ad es. considerando le politiche aziendali, gli standard, le convenzioni, i vincoli legali)
- Progetto/effort per lo sviluppo (ad es. il focus, gli standard tecnici, i rischi)

[1] La domanda di esame fornirà un sottoinsieme della checklist con cui rispondere alla domanda

- Tipo di prodotto di lavoro sottoposto a review (ad es. le code review potrebbero essere adattate a linguaggi di programmazione specifici)
- Livello di rischio del prodotto di lavoro sottoposto a review
- Tecniche di test da utilizzare

Buone checklist troveranno problemi e aiuteranno anche ad avviare discussioni su altri elementi che potrebbero non essere stati specificamente referenziati nella checklist. L'utilizzo di una combinazione di checklist è un modo efficace per garantire che una review permetta al prodotto di lavoro di raggiungere un livello maggiore di qualità. L'uso di review checklist-based con checklist standard come quelle specificate nel Syllabus Foundation Level, e lo sviluppo di checklist specifiche dell'organizzazione come quelle specificate sopra, aiuteranno il Test Analyst ad essere efficace nelle review.

Per ulteriori informazioni sulle review e sulle ispezioni, si veda [Gilb93] e [Wiegers03]. Ulteriori esempi di checklist possono essere ottenuti dai riferimenti riportati nel paragrafo 7.4.

6. Strumenti di Test e Test Automation - 90 min.

Parole Chiave
preparazione dei dati di test, progettazione dei test, esecuzione dei test, test script, testing keyword-driven

Obiettivi di Apprendimento per gli Strumenti di Test e Test Automation

6.1 Introduzione
Nessun Obiettivo di Apprendimento

6.2 Automazione Keyword-Driven
TA-6.2.1 (K3) Per uno specifico scenario, determinare le attività appropriate per un Test Analyst in un progetto di test keyword-driven

6.3 Tipi di Strumenti di Test
TA-6.3.1 (K2) Spiegare l'utilizzo e i tipi di strumenti di test applicati nella progettazione dei test, nella preparazione dei dati di test e nell'esecuzione dei test

6.1 Introduzione

Gli strumenti di test possono migliorare notevolmente l'efficienza e l'accuratezza del testing. Gli strumenti di test e gli approcci di automazione utilizzati da un Test Analyst sono descritti in questo capitolo. Dovrebbe essere osservato che i Test Analyst lavorano insieme agli sviluppatori, ai Test Automation Engineer e ai Technical Test Analyst per creare soluzioni di test automation. L'automazione keyword-driven, in particolare, coinvolge il Test Analyst e sfrutta la sua esperienza rispetto alle funzionalità di sistema e di business.

Ulteriori informazioni sull'argomento del test automation e sul ruolo del Test Automation Engineer sono fornite nel Syllabus ISTQB® Advanced Level Test Automation Engineer [ISTQB_TAE_SYL].

6.2 Automazione Keyword-Driven

Il testing keyword-driven è uno dei principali approcci di test automation, e coinvolge il Test Analyst nel fornire i principali input: keyword e dati.

Le keyword (alcune volte definite action word) sono principalmente, ma non esclusivamente, utilizzate per rappresentare interazioni di business di alto livello con un sistema (ad es. "annulla ordine"). Ogni keyword viene in genere utilizzata per rappresentare una serie di interazioni dettagliate tra un attore e il sistema sotto test. Le sequenze di keyword (inclusi i dati di test rilevanti) vengono utilizzate per specificare i test case.[Buwalda02]

Nel test automation, una keyword è implementata come uno o più test script eseguibili. Gli strumenti leggono i test case scritti come una sequenza di keyword, che richiamano i test script appropriati, che implementano la funzionalità della keyword. Gli script sono implementati in modo altamente modulare, per consentire un facile mapping alle specifiche keyword. Sono necessarie competenze di programmazione per implementare questi script modulari.

Di seguito sono riportati i principali vantaggi del testing keyword-driven:

- Le keyword che si riferiscono a una particolare applicazione o dominio di business possono essere definite da esperti del dominio. Questo può rendere più efficiente l'attività di specifica dei test case.
- Una persona con una grande competenza di dominio può trarre beneficio dall'esecuzione automatica dei test case (una volta che le keyword sono state implementate come script) senza dover comprendere il codice di automazione sottostante.
- L'uso di una tecnica di scrittura modulare consente una manutenzione efficiente dei test case da parte del Test Automation Engineer quando si verificano modifiche alla funzionalità e all'interfaccia nel software sotto test [Bath14].
- Le specifiche dei test case sono indipendenti dalla loro implementazione.

I Test Analyst di solito creano e manutengono i dati delle keyword/action word. Devono rendersi conto che l'attività di implementazione degli script è ancora necessaria per implementare le keyword. Dopo aver definito le keyword e i dati da utilizzare, il test automator (ad es. il Technical Test Analyst o il Test Automation Engineer) traduce le keyword dei processi di business e le azioni di livello più basso in test script automatizzati.

Mentre il testing keyword-driven viene in genere eseguito durante il testing di sistema, lo sviluppo del codice può iniziare in anticipo fin dalla progettazione dei test. In un ambiente iterativo, in particolare quando si utilizzano approcci di continuous integration/continuous deployment, lo sviluppo del test automation è un processo continuo.

Dopo aver creato le keyword e i dati di input, il Test Analyst si assume la responsabilità di eseguire i test case keyword-driven e di analizzare eventuali failure che potrebbero verificarsi.

Quando viene rilevata un'anomalia, il Test Analyst dovrebbe aiutare a investigare sulla causa del failure per determinare se il difetto è relativo alle keyword, ai dati di input, allo script di automazione stesso o all'applicazione sotto test. Di solito il primo passo per la risoluzione del problema è eseguire manualmente lo stesso test, con gli stessi dati, per vedere se il failure è nell'applicazione stessa. Se questo non mostra un failure, il Test Analyst dovrebbe rivedere la sequenza di test che ha portato al failure per determinare se il problema si è verificato in uno passo precedente (forse introducendo dati di input errati), anche se il difetto non si è evidenziato finché non è stata terminata l'elaborazione. Se il Test Analyst non è in grado di determinare la causa del failure, le informazioni sulla risoluzione del problema dovrebbero essere fornite al Technical Test Analyst o allo sviluppatore, per ulteriori analisi.

6.3 Tipi di Strumenti di Test

Gran parte del lavoro di un Test Analyst richiede l'uso efficace di strumenti. Questa efficacia è migliorata da quanto segue:

- Sapere quali strumenti usare
- Sapere che gli strumenti possono aumentare l'efficienza dell'effort del test (ad esempio, contribuendo a fornire una migliore copertura nei tempi consentiti)

6.3.1 Strumenti di Progettazione dei Test

Gli strumenti di progettazione dei test vengono utilizzati per creare test case e dati di test da applicare durante il testing. Questi strumenti possono funzionare a partire da specifici formati di documenti dei requisiti, modelli (ad es. UML, Unified Modeling Language) o da input forniti dal Test Analyst. Gli strumenti di progettazione dei test sono spesso progettati e realizzati per funzionare con formati e strumenti particolari, come specifici strumenti di requirements management.

Gli strumenti di progettazione dei test possono fornire informazioni che il Test Analyst utilizza per determinare i tipi di test necessari per ottenere il particolare livello target di copertura, di confidenza nel sistema o di azioni di mitigazione del rischio del prodotto. Ad esempio, gli strumenti dell'albero di classificazione generano (e visualizzano) l'insieme di combinazioni che sono necessarie per raggiungere la copertura completa in base a un criterio di copertura selezionato. Queste informazioni possono quindi essere utilizzate dal Test Analyst per determinare i test case che devono essere eseguiti.

6.3.2 Strumenti di Preparazione dei Dati di Test

Gli strumenti di preparazione dei dati di test possono offrire i seguenti benefici:

- Analizzare un documento, come un documento dei requisiti o anche il codice sorgente, per determinare i dati richiesti durante il testing per raggiungere un livello di copertura.
- Prendere un insieme di dati da un sistema in produzione ed eseguire uno "scrub" o anonimizzarlo per eliminare qualsiasi informazione personale, pur mantenendo l'integrità interna di tali dati. I dati sottoposti a scrub possono quindi essere utilizzati per il testing, senza il rischio di una perdita di sicurezza o di un uso improprio delle informazioni personali. Questo è particolarmente importante dove sono richiesti grandi volumi di dati realistici, e dove esistono rischi per la sicurezza e la privacy dei dati.
- Generare dati di test sintetici da determinati insiemi di parametri di input (ad es. per usarli nel testing casuale). Alcuni di questi strumenti analizzeranno la struttura del database per determinare quali input saranno richiesti dal Test Analyst.

6.3.3 Strumenti di Esecuzione dei Test Automatizzati

Gli strumenti di esecuzione dei test sono utilizzati dai Test Analyst a tutti i livelli di test per eseguire test automatizzati e verificarne il risultato ottenuto. L'obiettivo dell'utilizzo di uno strumento di esecuzione dei test è in genere uno o più dei seguenti:

- Ridurre i costi (in termini di effort e/o tempo)
- Eseguire più test
- Eseguire lo stesso test in molti ambienti
- Rendere più ripetibile l'esecuzione dei test
- Eseguire dei test che sarebbe impossibile eseguire manualmente (ad es. test di validazione di molti dati)

Questi obiettivi spesso si sovrappongono agli obiettivi principali, che mirano ad aumentare la copertura riducendo i costi.

Il ritorno sull'investimento degli strumenti di esecuzione dei test è generalmente più elevato quando si automatizzano i regression test, a causa del basso livello di manutenzione previsto e dell'esecuzione ripetuta dei test. Automatizzare gli smoke test può anche rappresentare un uso efficace dell'automazione, proprio per l'uso frequente dei test, la necessità di un risultato di test rapido e, sebbene i costi di manutenzione possono essere più elevati, la capacità di avere un modo automatizzato per valutare una nuova build in un ambiente di continuous integration.

Gli strumenti di esecuzione dei test vengono comunemente utilizzati durante il testing di sistema e di integrazione. Alcuni strumenti, in particolare gli strumenti di test delle API (Application Program Interface), possono essere utilizzati anche durante il testing di componente. Sfruttare gli strumenti dove sono più applicabili contribuirà a migliorare il ritorno sugli investimenti.

7. Riferimenti

7.1 Standard

[ISO25010] ISO/IEC 25010 (2011) Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models, Capitolo 4

[ISO29119-4] ISO/IEC/IEEE 29119-4 Software and Systems Engineering – Software Testing – Parte 4, Test Techniques, 2015

[OMG-DMN] Object Management Group: OMG® Decision Model and NotationTM, Versione 1.3, Dicembre 2019; url: www.omg.org/spec/DMN/, Capitolo 8

[OMG-UML] Object Management Group: OMG® Unified Modeling Language®, Versione 2.5.1, Dicembre 2017; url: www.omg.org/spec/UML/

[RTCA DO-178C/ED-12C]: Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12C, 2013, Capitolo 1

7.2 Documenti ISTQB® e IREB

[IREB_CPRE] IREB Certified Professional for Requirements Engineering Foundation Level Syllabus, Versione 2.2.2, 2017

[ISTQB_AL_OVIEW] ISTQB® Advanced Level Overview, Versione 2.0

[ISTQB_ALTTA_SYL] ISTQB® Advanced Level Technical Test Analyst Syllabus, Versione 2019

[ISTQB_FL_SYL] ISTQB® Foundation Level Syllabus, Versione 2018

[ISTQB_GLOSSARY] Standard glossary of terms used in Software Testing, url:https://glossary.istqb.org/

[ISTQB_TAE_SYL] ISTQB® Advanced Level Test Automation Engineer Syllabus, Versione 2017

[ISTQB_UT_SYL] ISTQB® Foundation Level Specialist Syllabus Usability Testing, Versione 2018

7.3 Libri e Articoli

[Bath14] Graham Bath, Judy McKay, “The Software Test Engineer’s Handbook (2nd Edition)”, Rocky Nook, 2014, 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: Becoming an effective and efficient test professional”, John Wiley and Sons, 2007, ISBN 978-0-470-12790-2

[Black09]: Rex Black, “Advanced Software Testing, Volume 1”, Rocky Nook, 2009, ISBN 978-1-933-952-19-2

[Buwalda02]: Hans Buwalda, “Integrated Test Design and Automation: Using the Test Frame Method”, Addison-Wesley Longman, 2002, ISBN 0-201-73725-6

[Chow1978] T.S. Chow, “Testing Software Design Modeled by Finite-State Machines,” IEEE Transactions on Software Engineering vol. SE-4, issue 3, May 1978, pp. 178-187

[Cohn04]: Mike Cohn, “User Stories Applied: For Agile Software Development”, Addison-Wesley Professional, 2004, ISBN 0-321-20568-5

[Copeland04]: Lee Copeland, ”A Practitioner's Guide to Software Test Design”, Artech House, 2004, ISBN 1-58053-791-X

[Craig02]: Rick David Craig, Stefan P. Jaskiel, “Systematic Software Testing”, Artech House, 2002, ISBN 1-580-53508-9

[Forgács19] István Forgács, Attila Kovács, “Practical Test Design”, BCS, 2019, ISBN 978-1-780-1747-23

[Gilb93]: Tom Gilb, Graham Dorothy, “Software Inspection”, Addison-Wesley, 1993, ISBN 0-201-63181-4

[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

[Kuhn16]: D. Richard Kuhn et al, “Introduction to Combinatorial Testing”, CRC Press, 2016, ISBN 978-0-429-18515-1

[Myers11]: Glenford J. Myers, The Art of Software Testing 3rd Edition, John Wiley & Sons, 2011, ISBN 978-1-118-03196-4

[Offutt16]: Jeff Offutt, Paul Ammann, “Introduction to Software Testing” – 2nd Edition, Cambridge University Press, 2016, ISBN 13: 9781107172012

[vanVeenendaal12]: Erik van Veenendaal, “Practical risk-based testing. Product Risk Management: The PRISMA Method”, 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: tips, tricks, tours, and techniques to guide test design”, Addison-Wesley, 2009, ISBN 0-321-63641-4

7.4 Altri Riferimenti

I seguenti riferimenti indicano le informazioni disponibili su Internet e altrove. Anche se questi riferimenti sono stati verificati al momento della pubblicazione di questo Syllabus Advanced Level, ISTQB® non può essere ritenuto responsabile se i riferimenti non sono più disponibili.

- Capitolo 3
  - Czerwonka, Jacek: www.pairwise.org

- Tassonomia dei difetti: www.testingeducation.org/a/bsct2.pdf
- Esempio di tassonomia dei difetti basata sul lavoro di Boris Beizer: www.ottovinter.dk/bugtaxst.doc
- Una buona panoramica di varie tassonomie: testingeducation.org/a/bugtax.pdf
- Testing euristici basati sul rischio di James Bach: https://terh.pw/raz.pdf
- “Exploring Exploratory Testing”, Cem Kaner e Andy Tinkham, www.kaner.com/pdfs/ExploringExploratoryTesting.pdf
- “An Exploratory Testing Workshop Report”, Pettichord, Bret, www.testingcraft.com/exploratorypettichord

- Capitolo 5
  - http://www.tmap.net/checklists-and-templates

8. Appendice A

La seguente tabella deriva dalla tabella completa fornita da ISO 25010. Si focalizza solo sulle caratteristiche di qualità trattate nel Syllabus Test Analyst, e confronta i termini usati in ISO 9126 (utilizzati nella versione 2012 del Syllabus) con quelli nel più recente ISO 25010 (utilizzati in questa versione).

ISO/IEC 25010	ISO/IEC 9126-1	Note
Adeguatezza funzionale	**Funzionalità**	
Completezza funzionale		
Correttezza funzionale	Accuratezza	
Appropriatezza funzionale	Adeguatezza	
	Interoperabilità	Spostato in Compatibilità
Usabilità		
Riconoscibilità dell'appropriatezza	Comprensibilità	Nuovo nome
Apprendibilità	Apprendibilità	
Operabilità	Operabilità	
Protezione da errori dell'utente		Nuova sotto-caratteristica
Estetica della user interface	Attrattività	Nuovo nome
Accessibilità		Nuova sotto-caratteristica
Compatibilità		Nuova definizione
Interoperabilità		
Coesistenza		Trattato nel Technical Test Analyst

9. Indice