

Certified Tester Test Automation Strategy Syllabus

Versione 1.0

International Software Testing Qualifications Board



Avviso di copyright

Avviso di Copyright © International Software Testing Qualifications Board (di seguito denominato ISTQB®)

ISTQB® è un marchio registrato di International Software Testing Qualifications Board.

Copyright © 2024 gli autori del syllabus Test Automation Strategy v1.0: Andrew Pollner (Chair), Péter Földházi, Patrick Quilter, Gergely Ágnecz, László Szikszai

Tutti i diritti riservati. Con la presente gli autori trasferiscono il copyright a ISTQB®. Gli autori (come attuali titolari del copyright) e ISTQB® (come futuro titolare del copyright) hanno concordato le seguenti condizioni di utilizzo:

È possibile copiare estratti di questo documento per uso non-commerciale, a condizione che venga riconosciuta la fonte. Qualsiasi Training Provider accreditato (azienda accreditata alla Formazione) può utilizzare il presente Syllabus come base per un corso di formazione, a condizione che gli autori e ISTQB® siano riconosciuti come fonte e proprietari del copyright del Syllabus, e a condizione che qualsiasi pubblicità di tale corso di formazione possa menzionare il Syllabus solo dopo l'accreditamento ufficiale dei materiali di formazione da parte di un Member Board riconosciuto da ISTQB®.

Qualsiasi individuo o gruppo di individui può utilizzare questo Syllabus come base per articoli e libri, a condizione che gli autori e ISTQB® siano riconosciuti come fonte e proprietari del copyright del Syllabus.

È proibito qualsiasi altro utilizzo di questo Syllabus senza avere prima ottenuto l'approvazione scritta di ISTQB®.

Qualsiasi Member Board riconosciuto da ISTQB® può tradurre questo Syllabus, a condizione che riproduca il suddetto avviso di copyright nella versione tradotta del Syllabus.

Storico delle review

Versione	Data	Osservazioni
Syllabus v1.0	2024/05/03	CT-TAS v1.0 GA Release

Indice dei contenuti

Avviso di copyright	1
Storia della revisione	3
Indice dei contenuti	4
Ringraziamenti	7
0 Introduzione	8
0.1 Scopo di questo Syllabus	8
0.2 Strategia di test automation nel testing del software	8
0.3 Percorso di carriera per tester e Test Automation Engineer	9
0.4 Business Outcome (Risultati di Business)	10
1 Introduzione e obiettivi della strategia di test automation - 45 minuti (K2)	15
1.1 Fattori di successo di un progetto di test automation	16
1.1.1 Definizione di obiettivi e finalità per una strategia di test automation	16
1.1.2 Identificare i fattori tecnici di successo di un progetto di automazione dei test	16
1.1.3 Riassumere i criteri di investimento appropriati nella selezione dei progetti candidati per la test automation	17
2 Risorse per il test automation - 60 minuti (K2)	18
2.1 Costi e rischi dell'implementazione di una soluzione di test automation	19
2.1.1 Confrontare le soluzioni tecniche alternative in base ai costi di gestione	19
2.1.2 Spiegare le considerazioni sul modello di licenza per gli strumenti di test automation	19
2.1.3 Fornire esempi di fattori da considerare nella definizione di una strategia di test automation.	20
2.2 Ruoli e responsabilità nell'ambito dell'automazione dei test	21
2.2.1 Riassumere i ruoli e le competenze necessarie per una soluzione di test automation di successo	21
3 Preparazione alla test automation - 225 minuti (K3)	22
3.1 Integrazione attraverso i livelli di test	23
3.1.1 Differenziare le distribuzioni dell'automazione dei test	23
3.1.2 Scegliere un approccio di test automation in base all'architettura del sistema da testare	24
3.1.3 Dimostrare come ottimizzare la distribuzione dell'automazione dei test per ottenere approcci shift-left e shift right	24
3.2 Considerazioni strategiche nei diversi modelli del ciclo di vita dello sviluppo software	25

3.2.1	Spiegare come i progetti di test automation si conformano ai modelli di ciclo di vita dello sviluppo software tradizionali	25
3.2.2	Spiegare come i progetti di automazione dei test sono conformi alle migliori pratiche di sviluppo software Agile che supportano l'automazione dei test.	26
3.2.3	Preparare i progetti di test automation in conformità con le best practice DevOps per ottenere continuous testing	26
3.3	Applicabilità e fattibilità dell'automazione dei test.....	26
3.3.1	Illustrare i criteri per determinare l'idoneità dei test per la Test Automation	26
3.3.2	Identificare le sfide che solo la test automation può affrontare	27
3.3.3	Identificare le condizioni di test difficili da automatizzare	27
4	Strategie organizzative di distribuzione e rilascio per la strategia di test automation - 135 minuti (K2)	29
4.1	Pianificazione delle soluzioni di test automation	30
4.1.1	Identificare i modi in cui la test automation supporta un time to market più breve	30
4.1.2	Identificare i modi in cui la test automation aiuta a verificare i defect reportati.....	30
4.1.3	Definire approcci che consentano lo sviluppo di scenari rilevanti dal punto di vista operativo per la test automation	31
4.2	Strategie di implementazione dell'automazione dei test.....	32
4.2.1	Definire una strategia di implementazione dell'automazione dei test	32
4.2.2	Identificare i rischi della test automation durante l'implementazione	33
4.2.3	Definire gli approcci per mitigare i rischi dell'implementazione.....	34
4.3	Dipendenze all'interno dell'ambiente di test	34
4.3.1	Definire i componenti della test automation nell'ambiente di test.	34
4.3.2	Identificare i componenti dell'infrastruttura e le dipendenze della Test Automation	35
4.3.3	Definire i dati di test automation e i requisiti di interfaccia.	36
5	Analisi degli impatti della test automation - 150 minuti (K3)	37
5.1	Investimenti per l'impostazione e la manutenibilità dei test automation	38
5.1.1	Dimostrare il ritorno dell'investimento nella creazione di una soluzione di test automation	38
5.2	Metriche per l'automazione dei test	39
5.2.1	Classificare le metriche per la test automation	39
5.3	Il valore dell'automazione dei test a livello di progetto e di organizzazione	40
5.3.1	Identificare le considerazioni organizzative per l'uso della Test Automation	40
5.3.2	Analizzare le caratteristiche del progetto che aiutano a determinare l'implementazione ottimale dei test automation Obiettivi del test.....	41
5.4	Decisioni prese in base ai report del test automation.....	43

5.4.1	Analizzare i dati dei test report per informare il processo decisionale	43
6	Strategie di implementazione e miglioramento per la test automation - 150 minuti (K3).....	44
6.1	Transizione delle attività dal test manuale al continuous testing.....	45
6.1.1	Descrivete i fattori e le attività di pianificazione nella transizione dal test manuale all'automazione dei test.	45
6.1.2	Descrivete i fattori e le attività di pianificazione nella transizione dall'automazione dei test al continuous testing.....	46
6.2	Strategia di test automation in tutta l'organizzazione	47
6.2.1	Eseguire una valutazione delle risorse e delle pratiche di test automation per identificare le aree di miglioramento.	47
7	Riferimenti	49
8	Appendice A - Obiettivi di apprendimento/Livello cognitivo di conoscenza	52
10	Appendice C - Note di rilascio	59
11	Appendice D - Termini specifici del dominio	60
12	Indice	61

Ringraziamenti

Questo documento è stato formalmente rilasciato dalla General Assembly di ISTQB® il 3 maggio 2024.

È stato prodotto dalla Test Automation Task Force dello Specialist Working Group dell'International Software Testing Qualifications Board: Graham Bath (Specialist Working Group Chair) Andrew Pollner (Specialist Working Group Vice Chair e Test Automation Task Force Chair), Péter Földházi, Patrick Quilter, Gergely Ágneecz, László Szikszai. I reviewer della Test Automation Task Force sono: Armin Beer, Armin Born, Geza Bujdosó, Renzo Cerquozzi, Jan Giesen, Arnika Hryszko, Kari Kakkonen, Gary Mogyorodi, Chris van Bael, Carsten Weise, Marc-Florian Wendland.

Reviewer tecnico: Gary Mogyorodi

Le seguenti persone hanno partecipato alla revisione, ai commenti e alla votazione di questo syllabus:

Horváth Ágota, Laura Albert, Prasunkumar Banerjee, Jürgen Beniermann, Armin Born, Piet de Roo, Nicola De Rosa, Dingguofu Ding Guofu, Elizabeta Fournieret, Jan Giesen, Erik Haartmans, Matthias Hamburg, Tobias Horn, Mattijs Kemmink, Ilia Kulakov, Ashish Kumar, Vincenzo Marrazzo, Marton Matyas, Patricia McQuaid, Smitha Mohandas, Ingvar Nordström, Sreeja Padmakumari, Nishan Portoyan, Meile Posthuma, Swapnil Shah, Péter Sótér, Szilard Szell, Richard Taylor, Giancarlo Tomasig, Chris Van Bael, Daniel Van der Zwan, Carsten Weise, Marc-Florian Wendland, Claude Zhang.

Traduzione italiana a cura di ITA-STQB (www.ita-stqb.org) – info@ita-stqb.org

Revisione della traduzione italiana a cura di: Raffaele Bozzini, Salvatore Reale

0 Introduzione

0.1 Scopo di questo Syllabus

Questo Syllabus costituisce la base per l'International Software Testing Qualification per la qualifica di Test Automation Strategy Specialist. ISTQB® fornisce questo Syllabus come segue:

1. Ai Member Board, per la traduzione nella loro lingua locale e per l'accreditamento dei Training Provider. I Member Board possono adattare il Syllabus alle loro specifiche esigenze linguistiche e modificare i riferimenti per adattarli alle loro pubblicazioni locali.
2. Agli organismi di certificazione, per derivare le domande d'esame nella loro lingua locale adattate agli obiettivi di apprendimento di questo Syllabus.
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 in modo indipendente).
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.

0.2 Test Automation Strategy nel Testing del Software

La certificazione di Test Automation Strategy Specialist è rivolta a chiunque sia coinvolto nel testing del software e nella test automation. Ciò include persone che ricoprono ruoli come tester, test analyst, test automation engineer, consulenti di testing, architetti di testing, test manager e sviluppatori di software. Questa certificazione Specialist è adatta anche a tutti coloro che desiderano una comprensione di base della test automation, come project manager, quality manager, software development manager, business analyst, IT director e consulenti di gestione.

Il Syllabus Test Automation Strategy Specialist (CT-TAS) presenta molteplici fattori che entrano in gioco nella pianificazione della test automation all'interno di un'organizzazione. Gli aspetti dell'implementazione come tecnica ingegneristica della test automation e le best practice non rientrano nell'ambito, in quanto sono trattati nel Syllabus separato ISTQB CTAL-TAE.

Test Automation Strategy affronta le esigenze di test automation al di là delle sfide di implementazione tecnica degli strumenti e di integrazione. Una visione strategica della test automation fornisce una visione dell'implementazione su tutti i progetti all'interno di un'organizzazione, in modo sistematico e coerente, che alla fine dimostra valore all'organizzazione.

0.3 Percorso di Carriera per i Tester e i Test Automation Engineer

Lo schema ISTQB® fornisce supporto ai professionisti del testing in tutte le fasi della loro carriera, offrendo una conoscenza ampia e approfondita. Chi ha conseguito la certificazione ISTQB® Test Automation Strategy Specialist può essere interessato anche alla Certificazione Test Automation Engineering (CTAL-TAE).

Chi ha conseguito la certificazione ISTQB® Certified Tester Test Automation Strategy Specialist può essere interessato anche alle altre certificazioni Core Advanced Level (Test Analyst, Technical Test Analyst e Test Management) e successivamente alle certificazioni Expert Level (Test Management o Improving the Test Process). Chiunque voglia sviluppare competenze nelle pratiche del testing in un ambiente Agile può prendere in considerazione le certificazioni Agile Technical Tester o Agile Test Leadership at Scale. Il percorso di certificazione Specialist offre un approfondimento in aree che hanno specifici approcci di test e attività di test, ad esempio Test Automation Engineering, Performance Testing, Security Testing, AI Testing e Mobile Application Testing, o dove è richiesta una conoscenza specifica del dominio (ad esempio, Automotive Software Testing o Game Testing). È possibile visitare il sito www.istqb.org per le ultime informazioni sul Certified Tester Scheme di ISTQB®.

0.4 Business Outcome

Questo paragrafo elenca i Business Outcome (Risultati di Business) previsti da una persona che ha conseguito la certificazione Test Automation Strategy Specialist.

Un Certified Tester Test Automation Strategy Specialist è in grado di...

TAS -B01	Comprendere i fattori del software e dei sistemi che influenzano il successo della test automation
TAS -B02	Identificare i costi e i rischi dell'implementazione di una test automation solution
TAS -B03	Comprendere i ruoli e le responsabilità delle persone che contribuiscono alla test automation
TAS -B04	Pianificare l'integrazione della test automation tra i livelli di test
TAS -B05	Identificare le considerazioni strategiche per l'implementazione della test automation nei diversi modelli del ciclo di vita dello sviluppo software
TAS -B06	Comprendere l'applicabilità e la fattibilità della test automation
TAS -B07	Pianificare test automation solution che soddisfino le esigenze dell'organizzazione
TAS -B08	Comprendere le strategie di deployment della test automation
TAS -B09	Comprendere le dipendenze della test automation all'interno dell'ambiente di test
TAS -B10	Comprendere i costi per la configurazione e la manutenzione della test automation
TAS -B11	Imparare quali metriche di test automation aiutano a guidare il processo decisionale
TAS -B12	Identificare i modi in cui la test automation apporta valore al progetto e all'organizzazione
TAS -B13	Identificare i requisiti di reporting della test automation per soddisfare le esigenze degli stakeholder
TAS -B14	Definire le attività di transizione dal testing manuale alla test automation
TAS -B15	Definire una strategia di test automation che garantisca che i progetti condividano risorse e metodi per assicurare un'implementazione coerente in tutta l'organizzazione

0.5 Obiettivi di Apprendimento Esaminabili e Livelli di Conoscenza

Gli Obiettivi di Apprendimento (LO, Learning Objective) supportano i Business Outcome e vengono utilizzati per creare gli esami di Certified Tester Test Automation Strategy Specialist.

In generale, tutti i contenuti di questo Syllabus sono esaminabili ai livelli K2 e K3, ad eccezione di Introduzione, Riferimenti e Appendici. Al candidato può quindi essere chiesto di riconoscere, ricordare o richiamare una parola chiave o un concetto menzionato in uno qualsiasi dei sei capitoli. I livelli specifici degli obiettivi di apprendimento sono specificati all'inizio di ogni capitolo e sono classificati come segue:

- K2: Capire
- K3: Applicare

Ulteriori dettagli ed esempi di Obiettivi di Apprendimento sono riportati nell'Appendice A.

Tutti i termini elencati come parole chiave sotto i titoli dei capitoli devono essere ricordati (K1), anche se non esplicitamente menzionati negli Obiettivi di Apprendimento.

0.6 L'Esame per il Certificato Test Automation Strategy Specialist

L'esame per il Certificato Test Automation Strategy Specialist si basa 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, delle Appendici e dei Riferimenti. Gli standard e i libri sono inclusi come riferimenti, ma il loro contenuto non è esaminabile, al di là di quanto sintetizzato in questo Syllabus relativamente a tali standard e libri.

Fare riferimento al documento *'Exam Structures and Rules v1.1 Compatible with Syllabus Foundation and Advanced Levels and Specialists Modules'* per ulteriori dettagli relativi all'esame per il Certificato Test Automation Strategy.

Il criterio di accesso per sostenere l'esame di certificazione Test Automation Strategy è che i candidati abbiano un interesse per il testing del software e la test automation. Tuttavia, è fortemente consigliato ai candidati anche di:

- Avere almeno un minimo di esperienza nello sviluppo del software e dei sistemi, e di leadership per l'implementazione della tecnologia nell'impresa e un'esperienza come senior test engineer, test lead o come sviluppatore di software.
- Seguire un corso accreditato secondo gli standard ISTQB (da uno dei Member Board riconosciuti da ISTQB).

Nota per i Requisiti di Ingresso: il certificato ISTQB® Foundation Level deve essere ottenuto prima di sostenere l'esame di certificazione Test Automation Strategy Specialist.

0.7 Accreditamento

Un Member Board ISTQB® può accreditare 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 a gestire un esame ISTQB® come parte del corso stesso.

Le linee guida per l'accreditamento di questo Syllabus seguono le *Accreditation Guidelines* pubblicate dal Working Group ISTQB® *Processes Management and Compliance*.

0.8 Gestione degli Standard

Il Syllabus Test Automation Strategy contiene riferimenti a standard (ad esempio, IEEE, ISO, ecc.). Lo scopo di questi riferimenti è fornire un framework (come nel caso dei riferimenti alla norma ISO 25010 sulle caratteristiche di qualità) o una fonte di informazioni aggiuntive, se il lettore lo desidera. Si noti che il Syllabus utilizza questi standard come riferimenti. Il contenuto degli standard non è oggetto di esame. Per ulteriori informazioni sugli standard, fare riferimento al Capitolo 7 Riferimenti.

0.9 Mantenere l'Attualità

L'industria del software cambia rapidamente. Per far fronte a questi cambiamenti e per fornire agli stakeholder l'accesso a informazioni rilevanti e aggiornate, i gruppi di lavoro ISTQB hanno creato dei link sul sito web www.istqb.org, che rimandano a documenti di supporto e a modifiche agli standard. Queste informazioni non sono oggetto d'esame nell'ambito del Test Automation Strategy Specialist Syllabus.

0.10 Livello di Dettaglio

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

- Obiettivi didattici generali che descrivono i propositi del Test Automation Strategy Specialist
- Una lista di termini (parole chiave) che gli studenti devono essere in grado di ricordare.
- Obiettivi di apprendimento per ogni area di conoscenza, che descrivono i risultati cognitivi di apprendimento che devono essere raggiunti.
- Una descrizione dei concetti chiave, che include i riferimenti a fonti riconosciute come accettate dalla letteratura o dagli standard.

Il contenuto del Syllabus non è una descrizione dell'intera area di conoscenza del testing del software, ma riflette il livello di dettaglio che deve essere coperto dai corsi di formazione Test Automation Strategy Specialist. Si focalizza sui concetti e sulle tecniche di test che possono essere applicati a tutti i progetti software, compresi quelli che seguono i metodi Agile. Questo Syllabus non contiene obiettivi di apprendimento specifici relativi al testing Agile, ma discute il modo in cui questi concetti si applicano ai progetti Agile e ad altri tipi di progetti.

0.11 Come è Organizzato questo Syllabus

Sono presenti sei capitoli con contenuti esaminabili. Nell'intestazione principale di ogni capitolo è specificato il tempo per il capitolo; la tempistica non viene fornita al di sotto del livello del capitolo. Per i corsi di formazione accreditati, il Syllabus richiede un minimo di 12,75 ore di lezione, distribuite nei sei capitoli come segue:

- Capitolo 1: 45 minuti - Introduzione e Obiettivi della Strategia di Test Automation
 - Il tester comprende i concetti di test automation e apprende i criteri di selezione dei progetti candidati
 - Il tester comprende i fattori che definiscono il successo di un'implementazione della test automation
- Capitolo 2: 60 minuti - Risorse per la Test Automation
 - Il tester apprende le diverse soluzioni che sono disponibili per la test automation e il relativo investimento per ognuna di esse
 - Vengono trattate le licenze software per gli strumenti di test automation
 - I tester comprendono le competenze necessarie per la test automation
- Capitolo 3: 225 minuti - Preparazione per la Test Automation
 - Il tester impara come viene utilizzata la test automation tra i vari livelli di test e all'interno di essi
 - Vengono trattate le strategie di test automation per distribuire adeguatamente il testing e raggiungere lo shift-left e lo shift-right
 - Il tester apprende come la test automation supporta i progetti legacy e Agile
 - Viene trattata la test automation nell'ambito di DevOps e pratiche di continuous testing
 - Il tester comprende come definire i criteri per l'uso della test automation, compresi i test più adatti per la test automation
- Capitolo 4: 135 minuti - Strategie Organizzative di Deployment e Release per la Test Automation
 - Il tester apprende come la test automation può migliorare il time to market
 - Il tester comprende come sviluppare test automatizzati rilevanti dal punto di vista operativo e il reporting dei difetti
 - Vengono trattate la strategia di deployment della test automation e la mitigazione del rischio
 - Il tester impara a conoscere l'ambiente di test automation e le sue dipendenze
 - Viene trattata l'integrazione della test automation e dei dati di test in un system under test

- Capitolo 5: 150 minuti - Analisi degli Impatti per la Test Automation
 - Vengono trattati le metriche di test automation e il reporting per aiutare a prendere decisioni
 - Un tester impara a realizzare un ritorno dell'investimento per la test automation
 - Vengono trattati gli obiettivi per un'organizzazione e per un progetto per utilizzare la test automation
 - Il tester impara ad analizzare i test report e a informare chi prende le decisioni in modo chiaro e comprensibile
- Capitolo 6: 150 minuti - Strategie di Implementazione e di Miglioramento per la Test Automation
 - Il tester impara come passare dal testing manuale alla test automation e al continuous testing
 - Viene trattata la valutazione della test automation per il miglioramento continuo

1 Introduzione e Obiettivi della Strategia di Test Automation - 45 minuti (K2)

Parole Chiave

test automation architecture, test automation framework, strategia di test automation

Obiettivi di Apprendimento per il Capitolo 1:

1.1 Fattori di Successo di un Progetto di Test Automation

CT-TAS -1.1.1 (K2) Spiegare gli obiettivi e la rilevanza della test automation

CT-TAS -1.1.2 (K2) Identificare i fattori tecnici di successo di un progetto di test automation

CT-TAS -1.1.3 (K2) Riassumere i criteri di investimento appropriati nella selezione dei progetti candidati per la test automation

1.1 Fattori di Successo di un Progetto di Test Automation

1.1.1 Definire obiettivi e traguardi di una strategia di test automation

Quando si definisce la strategia di un progetto di test automation, è necessario affrontare i seguenti aspetti:

- Definire lo scopo previsto
- Identificare i rischi
- Definire l'ambito
- Identificare gli stakeholder coinvolti
- Selezionare gli strumenti per l'automazione
- Progettare la test automation architecture (TAA)
- Identificare gli ambienti

Gli obiettivi della test automation possono includere:

- Miglioramento dell'efficienza dei test
- Fornire copertura più ampia e più profonda
- Miglioramento della qualità complessiva del SUT
- Riduzione dei costi totali e del time to market
- Esecuzione di test che i tester manuali non possono eseguire
- Riduzione dei tempi di esecuzione dei test
- Aumento della frequenza dei test

Poiché lo sviluppo software Agile fornisce cicli di deployment più rapidi e le applicazioni cloud sono più diffuse, lo sviluppo richiede un feedback più anticipato e più rapido sulla qualità del SUT. Come risultato di questo cambiamento, la test automation ha una maggiore messa a fuoco e rilevanza nei progetti moderni, data la sua natura e gli obiettivi dei test.

1.1.2 Identificare i fattori tecnici di successo di un progetto di test automation

I seguenti fattori di successo si applicano ai progetti di test automation che si focalizzano sui fattori che influiscono sul successo a lungo termine di un progetto. L'uso di un progetto pilota aiuta a determinare gli strumenti e la fattibilità delle tecnologie selezionate.

I fattori di successo per la test automation sono i seguenti:

- Testabilità del SUT
 - Consente alla test automation di accedere alle interfacce del SUT
- Definizione della strategia di test automation
 - La strategia deve essere applicabile e personalizzabile, gli obiettivi devono essere raggiungibili entro i vincoli di tempo e di costo, e deve essere mantenuta aggiornata
- TAA
 - Chiarezza su cosa e come implementare
 - Per ulteriori informazioni, vedere il Syllabus CTAL-TAE, sezione 3.1.1.

- Test automation framework (TAF)
 - TAF che sia semplice da utilizzare, ben documentato e manutenibile, supporta un approccio coerente all'automazione dei test. Per ulteriori informazioni, vedere il Syllabus CTAL-TAE, sezione 3.1.3.
 - Definizione e implementazione del reporting dei test
 - Facile risoluzione dei problemi
 - Ambiente di test appropriato
 - Test case automatizzati documentati
 - Tracciabilità dei test automatizzati rispetto alle definizioni e ai requisiti dei test case
 - Facile manutenzione
 - Test automatizzati aggiornati
 - Un piano chiaro di deployment
 - Un piano chiaro di esecuzione dei test
 - Test ritirati secondo necessità
 - Gestione efficace delle eccezioni

Prima di avviare il progetto di test automation, è importante analizzare le possibilità di successo del progetto, considerando i fattori presenti e quelli mancanti, tenendo conto dei rischi dell'approccio di test scelto e del contesto del progetto. Non tutti i fattori sono richiesti e, nella pratica, raramente tutti i fattori vengono soddisfatti.

1.1.3 Riassumere i criteri di investimento appropriati nella selezione dei progetti candidati per la test automation

La configurazione della test automation in un progetto comporta investimenti e, nella maggior parte dei casi, costi significativi. Questo aspetto dovrebbe essere preso in considerazione, insieme alla natura del progetto e se utilizzare o meno la test automation nel progetto.

Prima di introdurre la test automation, considerate i seguenti criteri di investimento:

- Costi di introduzione: Oltre al lavoro necessario per configurare la test automation, saranno richiesti costi aggiuntivi per il progetto, in quanto potrebbe essere necessario assumere nuovi test automation engineer (TAE), acquistare nuovo hardware o organizzare la formazione
- Fase attuale del ciclo di vita dello sviluppo software (SDLC) del progetto: È meglio iniziare il prima possibile, in modo che la test automation possa apportare più valore al più presto
- Durata attesa/pianificata del progetto/sviluppo software: Per un progetto più breve, potrebbero non esserci abbastanza risorse per iniziare o tempo a disposizione per far sì che la test automation apporti valore
- Costo di manutenzione: nel caso in cui si parta da zero, la configurazione di una test automation solution (TAS) richiederà tempo e dovrà essere mantenuta

Se l'investimento per l'introduzione della test automation in un progetto è accettabile, può iniziare la preparazione per la test automation. Ciò include la selezione del giusto approccio di test e degli strumenti di test automation. La responsabilità principale di questo processo è un ruolo strategico di un architetto di testing o di un test manager, che hanno le conoscenze necessarie in materia di test automation per prendere decisioni rilevanti.

2 Risorse per la Test Automation - 60 minuti (K2)

Parole Chiave

test automation engineer, test automation solution

Obiettivi di Apprendimento per il Capitolo 2:

2.1 Costi e Rischi dell'Implementazione di una Test Automation Solution

CT-TAS -2.1.1 (K2) Confrontare soluzioni tecniche alternative in termini di costi di proprietà

CT-TAS -2.1.2 (K2) Spiegare le considerazioni sul modello di licenza per gli strumenti di test automation

CT-TAS -2.1.3 (K2) Fornire esempi di fattori da considerare quando si definisce una strategia di test automation

2.2 Ruoli e Responsabilità all'interno della Test Automation

CT-TAS -2.2.1 (K2) Riassumere i ruoli e le competenze necessarie per una test automation solution di successo

2.1 Costi e Rischi dell'Implementazione di una Test Automation Solution

2.1.1 Confrontare soluzioni tecniche alternative in termini di costi di proprietà

In termini di proprietà, un approccio comune è una soluzione interna personalizzata basata su strumenti open source o commerciali. Altri approcci alternativi includono soluzioni commerciali non personalizzabili o la firma di un contratto con un'azienda esterna per far svolgere il lavoro al TAE.

La creazione interna di tutta la TAS assicura che tutti i costi, le risorse, i rischi e la gestione siano all'interno dell'organizzazione (ad esempio, membri del team/sviluppatori e risorse hardware e software necessarie per la soluzione effettiva). Un fattore chiave è che, seguendo questo approccio e allocando tempo a questa attività, la TAS può essere sviluppata senza bisogno di un contratto con un fornitore o un'azienda esterna, poiché tutti i TAE richiesti e le loro conoscenze sono disponibili all'interno dell'organizzazione. Tuttavia, per portarla a termine con successo, l'organizzazione deve avere i TAE giusti, assunti o formati, che possano guidare lo sviluppo della TAS.

Nel caso di una soluzione basata su un fornitore, la proprietà sarà condivisa tra il cliente e il fornitore, a seconda dei dettagli del contratto. Se nell'organizzazione è già presente uno strumento di test che è già stato sperimentato e soddisfa tutti i requisiti, e non sono necessarie ulteriori funzionalità nello strumento di test, questo approccio sarà più facile da adottare. I tester all'interno dell'organizzazione saranno in grado di lavorare in modo produttivo una volta ricevuta la formazione necessaria dal fornitore, e di solito all'interno dell'organizzazione ci saranno esperti in materia (SME) assegnati come product owner. Il rischio di questo approccio è che se fosse necessario eseguire del lavoro aggiuntivo (ad esempio, correggere i difetti dello strumento, e richieste di funzionalità aggiuntive) allo strumento di test, ci vorrebbe del tempo e sarebbe necessario negoziare con il fornitore per ottenerlo in tempo e nel modo giusto, il che può influenzare il lavoro dei tester che lavorano con questo strumento di test.

Se l'organizzazione non vuole assumersi la responsabilità di creare un team, l'outsourcing è una soluzione consigliata. Nel caso di una collaborazione con aziende esterne, il cliente non deve noleggiare o acquistare hardware, software o assumere dipendenti con le competenze richieste. Tutto il lavoro e i costi aggiuntivi saranno a carico dell'azienda esterna, così come la proprietà degli strumenti. Nel contratto devono essere definite aspettative e metriche misurabili per rendere trasparenti e visibili i progressi. Questo approccio è suggerito nel caso in cui l'organizzazione non intenda investire effort nell'assunzione di TAE a causa dei tempi brevi del progetto, dei costi o di altre considerazioni.

2.1.2 Spiegare le considerazioni sul modello di licenza per gli strumenti di test automation

Le licenze sono un aspetto importante da considerare quando si stabilisce la test automation. Ogni modello di licenza menzionato di seguito ha un impatto diverso sui fattori di usabilità e di costo, come ad esempio i costi di esecuzione dei test e la difficoltà di configurazione dell'ambiente di sviluppo.

I modelli di licenza includono:

- **Open-source:** In molti casi, le organizzazioni utilizzano strumenti open-source per raggiungere i loro obiettivi nella test automation. Il motivo principale è che non ci sono costi di licenza o di manutenzione per l'uso degli strumenti di test. Molti utenti e organizzazioni contribuiscono allo sviluppo di strumenti open-source, il che facilita la raccolta di informazioni e la ricezione di

supporto. La maggior parte delle licenze open-source consente inoltre alle persone di modificare gli strumenti, se necessario, o addirittura di ripubblicarli senza restrizioni significative.

- **Licenza per utente/macchina:** Gli strumenti commerciali spesso prevedono licenze per utente o per macchina. In termini di costi, gli strumenti possono essere utilizzati con efficienza quando l'organizzazione conosce il numero di TAE che lavoreranno su un determinato progetto a lungo termine. Spesso, più licenze un'organizzazione ordina, migliore è il prezzo che riceve.
- **Licenza floating:** È una licenza che può essere condivisa tra più persone nell'organizzazione per essere utilizzata in momenti diversi. Può essere molto utile quando ci sono molti TAE che lavorano su macchine diverse. Il numero di licenze viene calcolato in base all'uso contemporaneo, non al numero totale di TAE o alle macchine specifiche su cui eseguono i test.
- **Licenza runtime:** Molti strumenti commerciali hanno licenze runtime per eseguire la test automation. Questo tipo di licenza è disponibile presso i fornitori di cloud che ospitano strumenti di test automation, più sistemi operativi (OS) e versioni di browser come servizio. Esistono anche fornitori di cloud che forniscono l'accesso a complessi di dispositivi con una varietà di dispositivi mobili, piattaforme, versioni e reti cellulari. A coloro che utilizzano questi servizi viene addebitato solo il tempo impiegato per eseguire i test e quindi, viene addebitato un costo in base alla licenza runtime.

2.1.3 Fornire esempi di fattori da considerare quando si definisce una strategia di test automation

Molti fattori possono influenzare le decisioni sull'implementazione della test automation e sulla sua strategia.

I fattori più importanti sono:

- Vincoli di tempo
- Livello di competenza necessario e numero di TAE per sviluppare la TAS
- Hardware di test
- Licenze per strumenti di test
- Adattabilità
- Manutenzione
- Supporto per diverse piattaforme (ad esempio, Web, desktop e/o mobile)
- Supporto per la continuous integration/continuous delivery (CI/CD)
- Integrazione del test management e reporting dei test

Il primo e più importante fattore di costo è la tempistica del lavoro da svolgere e della test automation stessa.

Da un punto di vista temporale, se la scadenza è breve, un approccio spesso utilizzato è quello di assumere solo pochi TAE qualificati per creare la TAS. In termini di roadmap e schedulazione più lunghe, l'organizzazione può decidere il numero di TAE richiesti con più contesto. Quando il SUT cresce, migliora e diventa più complesso, si può decidere di assumere più TAE per implementare e mantenere la TAS e i test.

Altri fattori di costo importanti sono gli strumenti e le relative licenze. Il budget dovrebbe contenere gli strumenti di test richiesti, l'hardware e la formazione aggiuntiva per i TAE. Questi fattori sono strettamente

legati all'integrazione con altri strumenti e sistemi per svolgere qualsiasi funzione non-testing aggiuntiva, come il caricamento dei risultati dei test nel sistema di test management o l'attivazione di una build nel sistema di configuration management. Per queste funzioni aggiuntive esistono strumenti, gratuiti o a pagamento, e librerie da utilizzare, ma questi devono essere considerati durante la creazione della strategia di test automation e dovrebbero essere inclusi nel budget.

Il numero di ambienti di test e di agenti di un team di sviluppo dipende sempre dal contesto. Più il SUT e la schedulazione della release sono grandi e complessi, più risorse saranno richieste dall'organizzazione, con un conseguente aumento dei costi. Un approccio recente per limitare i costi delle risorse prevede l'utilizzo di fornitori di cloud e il pagamento di risorse hardware di test e di licenze runtime su richiesta, tuttavia l'approccio deve essere utilizzato con cautela poiché può essere controproducente. A volte le macchine possono essere lasciate in funzione e diventare più costose che con il proprio hardware. Questo approccio può ridurre gli elevati costi di manutenzione e operatività per il team TAE.

2.2 Ruoli e Responsabilità all'interno della Test Automation

2.2.1 Riassumere i ruoli e le competenze necessarie per una test automation solution di successo

Per una TAS di successo, le organizzazioni necessitano di TAE qualificati che abbiano una forte conoscenza della programmazione e dell'architettura tecnica.

A seconda delle dimensioni e della maturità del progetto, dovrebbe esserci anche almeno un forte SME (ad esempio, un test lead, un architetto del testing e un business analyst) che comprenda l'effettivo dominio di business e gli obiettivi di test. Il ruolo di questa persona è quello di aiutare a guidare e creare il concetto, basato sugli obiettivi di test, e su una roadmap per la TAS effettiva. Il team management e le competenze trasversali saranno necessarie per formare, motivare e costruire il team per il lavoro effettivo. [CTEL-TM-MTT]

Un TAE dovrebbe possedere forti competenze tecniche e conoscenze sui diversi SDLC, sull'architettura del SUT e sul suo ambiente di sviluppo. Oltre agli aspetti tecnici, un TAE dovrebbe possedere la capacità di collaborare con i test analyst e gli altri stakeholder agli obiettivi della test automation. Poiché il testing esaustivo non è realizzabile, lo stesso vale per la test automation: il 100% di copertura della test automation non è raggiungibile. Siccome il tempo e l'effort sono limitati, i TAE devono essere in grado di prioritizzare le condizioni di test più impattanti da coprire dal punto di vista del business e degli investimenti, contribuendo alle valutazioni del rischio.

3 Preparazione per la Test Automation - 225 minuti (K3)

Parole Chiave

testing delle API, testing di componente, testing di contratti, shift-left, shift right, system under test, approccio di test automation, condizione di test, test doppio, livello di test, piramide di test

Obiettivi di Apprendimento per il Capitolo 3:

3.1 Integrazione tra i Livelli di Test

CT-TAS -3.1.1 (K2) Differenziare tra le distribuzioni della test automation

CT-TAS -3.1.2 (K2) Selezionare un approccio di test automation basato sull'architettura del system under test

CT-TAS -3.1.3 (K3) Dimostrare come ottimizzare la distribuzione della test automation per ottenere approcci shift-left e shift right

3.2 Considerazioni Strategiche nei Diversi Modelli del Ciclo di Vita dello Sviluppo Software

CT-TAS -3.2.1 (K2) Spiegare come i progetti di test automation siano conformi ai modelli del ciclo di vita dello sviluppo software legacy

CT-TAS -3.2.2 (K2) Spiegare come i progetti di test automation siano conformi alle best practice di sviluppo software Agile che supportano la test automation

CT-TAS -3.2.3 (K3) Preparare i progetti di test automation per conformarsi alle best practice di DevOps che supportano la test automation nel continuous testing

3.3 Applicazione e Fattibilità della Test Automation

CT-TAS -3.3.1 (K2) Spiegare i criteri per determinare l'idoneità dei test per la test automation

CT-TAS -3.3.2 (K2) Identificare le sfide che solo la test automation può affrontare

CT-TAS -3.3.3 (K2) Identificare le condizioni di test difficili da automatizzare

3.1 Integrazione tra i Livelli di Test

3.1.1 Differenziare tra le distribuzioni della test automation

Mike Cohn ha ideato il concetto originale di piramide di test automation che descrive tre livelli: unit (ISTQB lo chiama componente), service e UI. Da allora sono apparse molte varianti, tutte accomunate dagli stessi principi di base che descrivono il ruolo dei livelli di test e la distribuzione dei test case tra questi livelli (vedere il Syllabus ISTQB CTFL, sezione 5.1.6 Piramide di test).

La piramide di test automation originale di Mike Cohn si focalizza sui tipi di test eseguiti dalla TAS. Il livello service può essere suddiviso in tre tipi di test: testing di integrazione dei componenti, testing di contratti e testing delle API.

Lo unit testing (ISTQB lo chiama testing di componente) serve a validare i singoli componenti, focalizzandosi sulla qualità del codice. Il testing di integrazione dei componenti consente la validazione dell'interfaccia utente (UI) e delle API utilizzando i test doppi, come mock e stub. Il testing di contratti consente la validazione dei contratti tra i servizi. Il testing delle API si focalizza sulla validazione funzionale di determinati servizi con dati reali attraverso connessioni di servizi reali. Il testing della UI è il testing end-to-end di un sistema che interagisce con la propria GUI.

È utile tracciare lo stato attuale del testing come una base di partenza e lo stato target. Questo fornisce una chiara comprensione di ciò che manca o quale sia un livello accettabile di testing. Ciò indicherà la quantità di testing effettuato per ogni livello di test e se si tratta di testing manuale o automatizzato. Lo stato target dipende anche dalla tempistica e da ciò che è raggiungibile in quel lasso di tempo. Se è possibile raggiungerlo, la forma a piramide dovrebbe essere la forma target.

Esempi di distribuzioni dei test:

- **Piramide:** Si tratta di una distribuzione equilibrata del testing, con meno testing eseguito ai livelli di test più alti e più testing eseguito ai livelli di test più bassi, con test stabili e più rapidi. Se le risorse disponibili e la tempistica lo consentono, questa è spesso la piramide dello stato target.
- **Cono gelato:** Questa è la versione rovesciata della piramide. C'è una quantità equilibrata di testing di servizi, ma il testing dipende fortemente dalla scoperta della maggior parte dei difetti nel livello di test UI, che di solito è più costoso da automatizzare a causa della sua complessità. A causa della mancanza di testing di componente, i difetti vengono scoperti più tardi nel SDLC.
- **Clessidra:** Il testing si concentra sui livelli di test più alti e più bassi, mentre il testing del livello service è per lo più assente, con il risultato di difetti di integrazione. Se la logica di business è fornita dalle API (cioè dai servizi), molti dei test della UI possono essere facilmente spostati a livelli di test inferiori.
- **Ombrello:** Il testing dipende completamente dai costosi test della UI. Questo comporta una lenta risoluzione dei difetti, una manutenzione costosa dei test case e della TAS. Se non è tecnicamente possibile implementare test case di basso livello, l'abbandono della forma a ombrello potrebbe non essere realizzabile e l'attenzione dovrebbe concentrarsi sull'ottimizzazione della test automation suite della UI, sulla riduzione dei tempi di esecuzione dei test e sul miglioramento della stabilità.

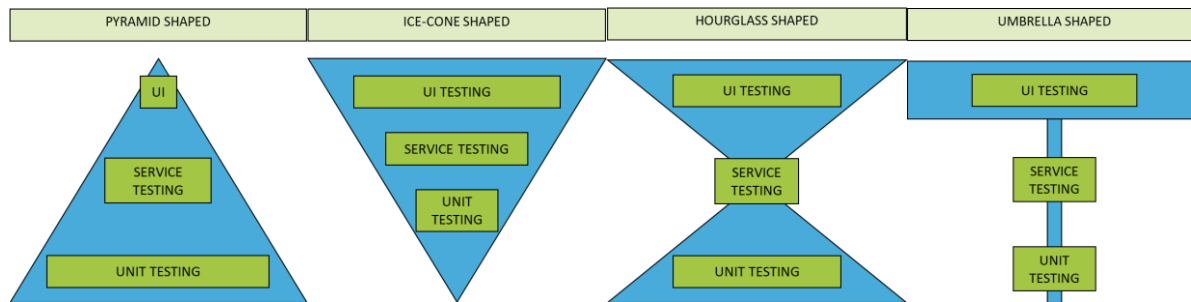


Figura 1: Esempi di distribuzioni dei test

3.1.2 Selezionare un approccio di test automation basato sull'architettura del system under test

Esistono molti modi per definire i livelli di test in una strategia di test. Quale scegliere dipende fortemente da diversi fattori, come la cultura dell'organizzazione, la maturità generale dell'ingegneria del software, il SDLC seguito e se il SUT ha un'architettura mainframe o a microservizi.

Sebbene la forma a piramide sia considerata la distribuzione dei test ideale, non sempre è realizzabile o richiede un tempo maggiore per la transizione a questo stato finale. È buona norma avere una forma realistica come stato target, come ad esempio la trasformazione dalla forma a ombrello a quella a clessidra. Una volta raggiunto questo obiettivo, è possibile determinare un nuovo stato target.

La scelta della distribuzione di test automation appropriata per i sistemi mainframe è diversa da quella dello sviluppo software moderno e dipende da molti fattori, poiché alcuni livelli potrebbero non essere affatto automatizzabili. L'accessibilità ai mainframe avviene tradizionalmente tramite emulatori di terminale (cioè green screen). La validazione dei processi batch è possibile ma limitata. Il testing di componente è più difficile da realizzare. Poiché i microservizi sono poco o per nulla presenti, non è sempre possibile validare la comunicazione dei dati tra le interfacce con il testing delle API. Il testing attraverso lavori batch, database e una GUI è più diffuso.

Le organizzazioni che stanno modernizzando le loro soluzioni legacy, passando lentamente a un'architettura a microservizi, possono introdurre il testing delle API e il testing di contratti nelle parti modernizzate del sistema.

3.1.3 Dimostrare come ottimizzare la distribuzione della test automation per ottenere approcci shift-left e shift right

Una volta identificata la distribuzione dello stato attuale e selezionata la distribuzione dello stato target, è possibile pianificare una roadmap di miglioramenti. Questo determina cosa testare con l'automazione (cioè l'ambito della test automation) e come testare (cioè la strategia di test automation). È necessario determinare un backlog prioritizzato di elementi da implementare e i criteri di ingresso per la test automation.

In caso di distribuzione sbilanciata della test automation, si consiglia di adottare un approccio bottom-up. Se la copertura del codice è bassa, è segno di una mancanza di test case di componente e occorre scrivere test case di componente aggiuntivi. Come secondo passo, è necessario eseguire la review della qualità dei test di componente e, se necessario, migliorarla. Suggestire le best practice, seguire una

tecnica di Test-Driven Development e organizzare workshop sulle tecniche di test può migliorare la qualità della test automation.

Introdurre il testing di componente per i test della UI e delle API, utilizzando i test doppi (ad esempio, mock, stub) aiuta a ottenere uno shift-left da test costosi, lenti e inaffidabili. Eliminare la dipendenza da servizi e dati reali migliora la coerenza dell'esecuzione dei test e fornisce un feedback anticipato facile da integrare nelle pipeline di CI/CD.

Negli ultimi anni, il testing di contratti ha assunto un ruolo sempre più importante. Validando il contratto tra un fornitore e un consumatore, è più facile scoprire prima le root cause dei difetti. I team non si affidano ai test delle API o ai test della UI per rilevare i difetti dei servizi sottostanti e possono invece ridurre il numero di tali test case. La creazione di un grafico delle chiamate per le API è un modo intelligente per tenere traccia di quali API sono collegate tra loro e di quali sono i consumatori o i produttori di una determinata API. Ciò consente di identificare meglio i potenziali punti critici di un sistema.

Mentre shift-left è un approccio che prevede lo spostamento dei test all'inizio del SDLC, shift right sposta i test più tardi, quando il SUT è stata rilasciato, come mezzo per valutare le prestazioni in un ambiente di test di pre-produzione o in produzione. Applicando lo shift right, i tester possono monitorare le prestazioni delle applicazioni e delle API mentre ricevono feedback dagli utenti effettivi. Sebbene la test automation sia più importante in un approccio shift-left, se è combinata con l'osservabilità, la test automation può aiutare a prendere decisioni più rapidamente sul fatto che il rilascio completo possa procedere o se il release candidate debba essere ritirato.

Il testing shift right ha lo scopo di:

- Comprendere le preferenze degli utenti
- Supportare i canary release e i dark launch per disturbare al minimo le funzionalità degli utenti
- Identificare precocemente i difetti nella produzione
- Contribuire ad ampliare l'ambito e l'uso della test automation
- Aumentare la copertura

3.2 Considerazioni Strategiche nei Diversi Modelli del Ciclo di Vita dello Sviluppo Software

3.2.1 Spiegare come i progetti di test automation siano conformi ai modelli del ciclo di vita dello sviluppo software legacy

Nel modello waterfall, la fase di testing viene dopo le fasi di analisi dei requisiti, progettazione del sistema e implementazione. A causa di questo ordine rigoroso, le attività di test automation iniziano più tardi. Cicli più lunghi tra i testing significano meno opportunità di utilizzare la test automation e il feedback della test automation di solito arriva troppo tardi, con il risultato di un minore ritorno dell'investimento (ROI).

Nel V-model, tutta la pianificazione dei test e la preparazione della documentazione avvengono nelle prime fasi. Ad esempio, la fase di progettazione dell'architettura produce la progettazione dei test di integrazione, che configura il testing in anticipo rispetto al modello waterfall. Purtroppo, l'implementazione attuale di una TAS avviene comunque più tardi nel SDLC e, sebbene il ROI della test automation sia più

elevato rispetto al modello waterfall, è ancora inferiore rispetto alle moderne pratiche di sviluppo software Agile.

3.2.2 Spiegare come i progetti di test automation siano conformi alle best practice di sviluppo software Agile che supportano la test automation

Uno degli ideali del seguire lo sviluppo software Agile è quello di ottenere la test automation in-sprint. Ciò significa determinare tutte le attività di test automation necessarie come parte dei criteri di uscita per ognuna delle user story. Include le definizioni dei test case, l'implementazione dei test case automatizzati, gli aggiornamenti del TAF e, in alcuni casi, l'integrazione in una pipeline di CI/CD. Le organizzazioni che non applicano correttamente le pratiche Agile non includono una stima dell'effort di test e lo gestiscono in un ticket separato o non lo monitorano affatto. Raggiungendo la test automation in-sprint, i team Agile si assicurano di essere pronti a distribuire l'ambito concordato entro o per la fine di ogni sprint. Se un team non è maturo da un punto di vista Agile, il primo obiettivo dovrebbe essere quello di raggiungere il testing in-sprint, mentre l'automazione sarebbe in ritardo di uno sprint. Da quel punto, un team può lavorare per raggiungere la test automation in-sprint.

3.2.3 Preparare i progetti di test automation per conformarsi alle best practice di DevOps per ottenere il continuous testing

Lo sviluppo software Agile si focalizza su come è organizzato il lavoro, mentre DevOps è responsabile della consegna end-to-end del software. Questo si ottiene automatizzando le attività di creazione, integrazione, test, deployment e produzione. Questo facilita il continuous testing attraverso cicli di feedback che garantiscono un miglioramento continuo.

L'enfasi è maggiore sull'implementazione di test case automatizzati di livello inferiore, tra cui test di componente, di integrazione dei componenti, e di contratti. Riducendo la dipendenza da dati e servizi reali, i test saranno eseguiti in tempi più brevi. Se possibile, la test automation dovrebbe essere eseguita nella stessa pipeline in cui viene costruito il SUT. Le diverse test suite vengono attivate dopo ogni fase di sviluppo e costruzione (ad esempio, local, pull request, merge e deploy).

Se i tester hanno la capacità di costruire test suite della UI e delle API più grandi, queste vengono eseguite separatamente per fornire valore aggiuntivo. È suggerito focalizzare l'effort del testing manuale sul testing esplorativo e sul feedback degli utenti finali, come attività complementare alla test automation.

3.3 Applicabilità e Fattibilità della Test Automation

3.3.1 Spiegare i criteri per determinare l'idoneità dei test per la test automation

La selezione dei test case per la test automation viene solitamente effettuata da un test analyst che comprende quali test case possono essere automatizzati, oppure da un TAE che ha la conoscenza del dominio necessaria per prendere tali decisioni.

I seguenti aspetti vengono considerati durante la selezione e la prioritizzazione dei test case per la test automation:

- È tecnicamente possibile implementare i test case in modo automatizzato?
- Ci sono sfide tecniche che influiscono sulla consegna dei test case automatizzati? Il team è preparato e ben formato per svolgere il lavoro di implementazione?
- L'effort di codifica fornisce un ROI adeguato? (vedere sezione 5.1.1)
- C'è valore nell'eseguire frequentemente i test case?
- Si tratta di un test funzionale o non-funzionale? Fa parte della smoke test suite, della regression test suite o della confirmation test suite?
- Il test case è ripetibile?
- Il test case è facile da mantenere quando il SUT cambia a causa di aggiornamenti?
- Il test case copre i flussi di lavoro di business utilizzati frequentemente?
- Esiste una sovrapposizione funzionale tra i test che consenta la riutilizzabilità dei passi del test e dei dati di test?

3.3.2 Identificare le sfide che solo la test automation può affrontare

Ci sono alcuni test che possono essere eseguiti solo con la test automation. Questi includono categorie quando:

- 1 L'esecuzione dei test manuali richiede tempi più lunghi di quelli adeguati
- 2 L'esecuzione dei test case deve essere sincronizzata
- 3 I risultati del test devono essere disponibili in una pipeline
- 4 I file di log di grandi dimensioni devono essere analizzati per individuare i difetti
- 5 È richiesta la precisione nei tempi nei test
- 6 Sono richieste permutazioni di test su più OS, browser, dispositivi, luoghi o configurazioni
- 7 Per massimizzare la copertura è richiesto un grande volume di esecuzioni dei test e/o di dati di input
- 8 Qualsiasi testing non-funzionale che richieda monitoraggio e analisi automatizzati o input da parte di un'enorme quantità di utenti, come stress testing o testing di affidabilità

3.3.3 Identificare le condizioni di test difficili da automatizzare

Ci sono alcune condizioni di test che rendono l'automazione dei test case una sfida difficile o addirittura un compito impossibile. Esistono molti esempi di vita reale. Alcuni di questi sono elencati di seguito:

- Validazione dei requisiti di progettazione, compresa la coerenza della UI su diverse piattaforme. L'aspetto generale e la sensazione del software sono soggettivi e richiedono la review di un essere umano.
- Qualsiasi test case che preveda un'interazione umana eccessiva. Nel settore finanziario, un buon esempio potrebbe essere la richiesta di un prestito. In questo processo possono verificarsi casi di determinate normative o condizioni (ad esempio, reddito insufficiente e dati personali errati). In queste situazioni, gli agenti devono ricontrollare l'effettiva richiesta di prestito e prendere decisioni manuali o contattare il cliente.
- Difficoltà tecniche che bloccano le attività di test automation, come le restrizioni a livello di OS. Un esempio è il software OS nativo che invia messaggi di testo agli utenti. Le interazioni o le validazioni di questi messaggi di testo non sono possibili con gli strumenti di test automation della UI, poiché l'OS non consente interazioni al di fuori del SUT target.

- Condizioni di test con una lunga dipendenza temporale renderebbero la test automation inefficiente e spesso è difficile da configurare correttamente rispetto all'esecuzione dei test manuali. Ad esempio, il tester deve effettuare il login al SUT, aggiornare il contenuto della home page e attendere due ore prima che il SUT si disconnetta automaticamente a causa di un timeout della sessione. Se il tempo trascorso non può essere modificato manualmente (ad esempio, mock, risposte lato server e modifiche dell'orario a livello di OS), si sconsiglia l'implementazione di tali test case in modo automatizzato.

4 Strategie Organizzative di Deployment e Release per la Test Automation - 135 minuti (K2)

Parole Chiave

componente, testing confermativo, quality gate

Obiettivi di Apprendimento per il Capitolo 4:

4.1 Pianificazione della Test Automation Solution

CT-TAS -4.1.1 (K2) Identificare i modi in cui la test automation supporta un time to market più breve

CT-TAS -4.1.2 (K2) Identificare i modi in cui la test automation aiuta a verificare i difetti segnalati in base ai requisiti

CT-TAS -4.1.3 (K2) Definire approcci che consentano lo sviluppo di scenari rilevanti dal punto di vista operativo per la test automation

4.2 Strategie di Deployment

CT-TAS -4.2.1 (K2) Definire una strategia di deployment della test automation

CT-TAS -4.2.2 (K2) Identificare i rischi della test automation nel deployment

CT-TAS -4.2.3 (K2) Definire approcci per mitigare i rischi del deployment dei test

4.3 Dipendenze all'interno dell'Ambiente di Test

CT-TAS -4.3.1 (K2) Definire i componenti della test automation nell'ambiente di test

CT-TAS -4.3.2 (K2) Identificare i componenti e le dipendenze dell'infrastruttura della test automation

CT-TAS -4.3.3 (K2) Definire i requisiti dei dati e dell'interfaccia della test automation per l'integrazione all'interno del system under test

4.1 Pianificazione della Test Automation Solution

4.1.1 Identificare i modi in cui la test automation supporta un time to market più breve

La test automation aiuta a portare il software sul mercato più velocemente, perché consente di ridurre il tempo del ciclo di test. Il testing che precede la release di un software richiede una quantità approfondita di verifica e validazione. Ciò può essere particolarmente significativo durante il regression testing, poiché questo tipo di testing favorisce il riutilizzo, riduce i costi e fornisce coerenza tra le esecuzioni dei test.

La test automation aiuta a ridurre l'effort dei test manuali e fornisce un feedback rapido agli sviluppatori, coprendo lo stesso ambito del testing. Inoltre, consente di eseguire il testing nelle prime fasi del ciclo di vita dello sviluppo software se vengono automatizzati i test di componente e i test di integrazione dei componenti, che in genere non vengono eseguiti manualmente.

Un quality gate è una misurazione forzata incorporata nel vostro processo che il software deve soddisfare prima di poter passare alla fase successiva. La configurazione di quality gate basati sulla test automation consente un processo di deployment accelerato ad un ambiente di pre-produzione o di produzione.

Utilizzando questi quality gate, i difetti possono essere scoperti prima, con conseguente riduzione del time to market. I tempi di esecuzione dei test possono essere ridotti utilizzando l'esecuzione parallela dei test e seguendo un approccio shift-left. Un approccio shift-left promuove una cultura di consapevolezza della qualità e incoraggia l'esecuzione del testing nelle prime fasi del ciclo di vita dello sviluppo software. Questo può includere test case multipli indipendenti o testing cross-browser.

Per ulteriori informazioni, vedere le sezioni 3.1.3, 6.1.2 e 6.2.1.

4.1.2 Identificare i modi in cui la test automation aiuta a verificare i difetti segnalati

Il testing confermativo eseguito a seguito di una correzione del codice può risolvere un difetto segnalato. Il tester segue in genere i passi del test necessari per replicare il difetto e verificare che non esista più.

I difetti hanno un modo per reintrodursi nelle release successive (ad esempio, questo può indicare un problema di configuration management o di gestione del repository del codice) e quindi i test confermativi sono candidati adatti alla test automation e possono essere aggiunti alla regression test suite esistente.

Un test confermativo automatizzato ha in genere un ambito ristretto di funzionalità. L'implementazione può avvenire in qualsiasi momento, una volta segnalato un difetto e compresi i passi del test necessari per replicarlo.

La tracciabilità dei test confermativi automatizzati consente il reporting del tempo e del numero di cicli di test impiegati per la risoluzione dei difetti.

Verificando le correzioni su più piattaforme, dispositivi, browser e versioni OS con la test automation, la quantità di tempo speso sul testing è notevolmente ridotto.

4.1.3 Definire approcci che consentano lo sviluppo di scenari rilevanti dal punto di vista operativo per la test automation

L'operational acceptance testing assicura la prontezza del software per i sistemi di produzione e viene tipicamente condotto appena prima di una release. Questo effort ha lo scopo di effettuare una validazione finale dei sistemi, dei componenti e delle altre infrastrutture del SUT e di testarne la prontezza per la produzione. Questo è necessario perché, nonostante i migliori sforzi di un TAE, non esiste garanzia che il SUT si comporti allo stesso modo al di fuori dell'ambiente di test e in produzione.

Un buon test plan operativo include testing di affidabilità, di tolleranza ai guasti, di integrità e di manutenibilità. Di seguito sono riportati gli approcci di test automation che possono essere utilizzati:

- Analisi statica del codice - Strumenti automatizzati che analizzano il codice per la sicurezza e le vulnerabilità. I risultati dei test vengono memorizzati per l'analisi delle tendenze nel tempo.
- Testing end-to-end - Test script automatizzati che conducono scenari utente end-to-end ed esercitano tutti gli aspetti dell'ambiente di test per scoprire eventuali difetti.
- Failover testing - Test script automatizzati che testano in modo specifico cosa succede quando l'hardware di un'applicazione si disconnette. Alcuni esempi includono il modo in cui l'applicazione si riprende quando i server fisici, i server cloud, le reti, i dischi dei computer e altri componenti hardware funzionano male. I test script sono progettati per misurare il successo o il fallimento della capacità del SUT di ripristinarsi automaticamente e questo è particolarmente importante per le organizzazioni che implementano il chaos engineering.
- Backup and restore testing - Test script automatizzati che testano il successo dell'esecuzione di un backup della versione attuale e del successivo rollback a un punto precedente (cioè una versione precedente del software).
- Testing di efficienza delle prestazioni (ad esempio, load testing) - Test script automatizzati che possono essere utilizzati per simulare l'utilizzo del SUT da parte di molti utenti contemporaneamente. I risultati dei test vengono memorizzati per il confronto e l'analisi delle tendenze nel tempo.
- Review della documentazione operativa - Per questa attività si possono utilizzare test script automatizzati per confrontare le versioni della documentazione del SUT e segnalare ai team di sviluppo la necessità di un aggiornamento in base alle nuove funzionalità aggiunte a una particolare release.
- Testing di sicurezza - È possibile creare test script automatizzati in combinazione con strumenti di test di sicurezza standard del settore per valutare il SUT, in modo statico e dinamico. I risultati dei test vengono memorizzati nel tempo per il confronto e l'analisi delle tendenze.
- Monitoraggio basato sul service level agreement di un'organizzazione - I test script automatizzati utilizzati durante il SDLC possono essere riutilizzati per monitorare le operazioni di produzione e inviare avvisi se un test automatizzato fallisce. Questa è una misura proattiva per individuare interruzioni della produzione prima che gli utenti reali le sperimentino.

La test automation per la validazione del software operativo può fornire immensi vantaggi, soprattutto se i test automatizzati possono essere eseguiti ripetutamente e in più ambienti. È possibile creare test suite automatizzate dedicate, in modo che i risultati dei test possano essere generati e confrontati con le esecuzioni dei test precedenti. Questo garantisce la coerenza quando vengono rilasciate nuove versioni del SUT. Una test suite automatizzata affidabile di condizioni di test specifiche operative può ridurre i costi nel tempo.

4.2 Strategie di Deployment della Test Automation

4.2.1 Definire una strategia di deployment della test automation

Una buona strategia di deployment della test automation terrà conto, ma non solo, dell'ambiente di test complessivo, degli strumenti di test disponibili, dell'accessibilità al SUT, della memorizzazione dei test script e di altre dipendenze, nonché del provisioning dei dati di test. Con queste considerazioni di alto livello in mente, un TAE può iniziare a pensare a una strategia per lo sviluppo e il deployment della TAS.

Ambiente di test - I TAE dovrebbero considerare come accederanno al SUT nei diversi ambienti di test. Quando iniziano a sviluppare i loro testware di test automation, sia che utilizzino un approccio keyword-driven o un'altra soluzione, devono considerare come questa soluzione verrà eseguita in più ambienti di test. Ad esempio, un test script dovrebbe essere sviluppato in modo tale da poter essere eseguito in un ambiente di test e in un ambiente di pre-produzione con modifiche minime. In genere, si tratta di modificare uno uniform resource locator (URL) per un'applicazione Web-based e in seguito il test script viene eseguito allo stesso modo indipendentemente dall'ambiente di test in cui si trova.

Strumenti - I TAE dovranno anche considerare quali strumenti stanno usando per costruire la TAS. Se si tratta di uno strumento commerciale, è probabile che dovranno capire come viene concesso in licenza. Il TAE potrebbe scoprire che il suo ambiente di test deve avere accesso al server di licenza dello strumento. Questo aspetto deve essere preso in considerazione quando il test script è destinato a essere utilizzato in più ambienti di test. Il fatto che il server delle licenze sia disponibile nell'ambiente di test non significa necessariamente che lo sarà anche nell'ambiente di pre-produzione.

Accesso al software - È necessario comprendere delle importanti considerazioni su come accedere al SUT. I test script possono essere progettati per accettare parametri in modo che utenti specifici o credenziali con accesso speciale possano essere aggiornati rapidamente quando cambiano gli end point del SUT. Anche questo aspetto diventa molto importante quando si passa a diversi ambienti di test e l'URL deve cambiare. Inoltre, potrebbe essere necessario utilizzare credenziali diverse (ad esempio, tramite account utente e di test, biometria e smart card) a seconda dell'ambiente di pre-produzione. Una buona progettazione di test script automatizzati includerà una flessibilità tale da consentire la configurazione di parametri semplici e l'esecuzione dei test script come previsto, indipendentemente dall'ambiente di test.

Archiviazione dei test script - Il TAE dovrà decidere una posizione centrale per archiviare e gestire i test script automatizzati. Una buona strategia includerebbe un repository di codice sorgente che utilizzi il configuration management. In questo modo, i test script possono essere accessibili da più ambienti di test, purché abbiano accesso al repository di codice sorgente, e le versioni possono essere create per la versione specifica del SUT. Tutte le configurazioni e le dipendenze possono essere salvate nello stesso repository dei test script, e forniscono una portabilità ottimale. In breve, la TAS, il TAF e tutti i test case possono essere archiviati e gestiti all'interno di repository.

Provisioning dei dati - Sarà importante capire se un test script automatizzato dipende da alcuni dati di test già esistenti nell'ambiente di test in cui viene testato il SUT. Una buona progettazione dei test script eviterà il più possibile i dati statici (cioè i set di dati fissi). Tuttavia, ci saranno situazioni in cui ciò non è possibile. In questi casi, il TAE dovrà determinare una soluzione per avere i dati di test precaricati o utilizzare la test automation per creare test script di setup che generano i dati di test necessari prima che i test script reali vengano eseguiti e testino il SUT. Entrambi gli approcci presentano pro e contro. Da un lato è conveniente che un amministratore precarichi i dati di test. Tuttavia, il TAE si affida poi ad un'altra risorsa per avere la disponibilità di supportarli. Essere autosufficienti con un test script di setup elimina la necessità di affidarsi a un'altra risorsa. Tuttavia, la creazione di questi test script richiede tempo e diventa un'altra parte della soluzione che deve essere mantenuta.

Tenendo conto di tutti gli elementi sopra menzionati, la strategia di test automation sarà in grado di fornire una pianificazione e un controllo adeguati delle attività di test automation.

4.2.2 Identificare i rischi della test automation nel deployment

I problemi tecnici possono portare a rischi di prodotto e rischi di progetto (per ulteriori informazioni, vedere il Syllabus CTFL, sezione 5.2.2). I problemi tecnici tipici includono:

- Troppe astrazioni possono portare a difficoltà nella comprensione di ciò che il codice di test automation sta realmente facendo (ad esempio, con le keyword nell'approccio keyword-driven)
- Le tabelle dei dati di test possono diventare troppo grandi/complesse/difficili da migrare in altri ambienti di test, risultando in esiti di stato incoerenti
- Dipendenza dalla TAS per l'utilizzo di alcune librerie OS o di altri componenti che potrebbero non essere disponibili in tutti gli ambienti di test del SUT

I rischi di progetto di deployment tipici includono:

- Problemi di personale: Trovare le persone giuste per mantenere la test automation può essere difficile
- Manutenzione non pianificata della TAS dovuta ad aggiornamenti del SUT che causano un funzionamento errato della TAS
- Ritardi nell'introduzione della test automation
- Ritardi nell'aggiornamento della TAS in base alle modifiche apportate al SUT
- La TAS non può catturare oggetti UI non standard
- Permettere ai test case obsoleti di rimanere nelle test suite, sprecando tempo per l'esecuzione dei test

I potenziali punti di failure del progetto della TAS includono:

- Migrazione a un altro ambiente di test
- Deployment in un ambiente di produzione
- Dimenticare che l'automazione è un software che dovrebbe anche essere testato

È importante rendersi conto che vari problemi tecnici, rischi di progetto e potenziali punti di failure possono compromettere il successo dei progetti di test automation. I problemi tecnici più comuni includono le complessità derivanti da un'eccessiva astrazione, le difficoltà nella gestione dei dati di test e le dipendenze da componenti specifici. I rischi di progetto devono anche tenere conto delle difficoltà di personale, dei problemi di manutenzione dovuti agli aggiornamenti del sistema, dei ritardi nel deployment e della presenza di test case obsoleti. Inoltre, i potenziali punti di failure, come la migrazione in ambienti diversi e l'aver trascurato la necessità di testare il software di automazione stesso, dovrebbero essere

affrontati con una pianificazione adeguata. Nel complesso, il monitoraggio e le misure proattive garantiranno il successo dei progetti di test automation.

4.2.3 Definire approcci per mitigare i rischi del deployment

Esistono molte strategie di mitigazione del rischio che possono essere impiegate per affrontare queste aree di rischio. Alcune di queste sono discusse di seguito.

La TAS ha un proprio SDLC, sia che si tratti di una soluzione sviluppata internamente sia che si tratti di una soluzione acquisita. Una cosa da ricordare è che la TAS, come qualsiasi altro software, deve essere sottoposta a configuration management e le sue funzionalità devono essere documentate. In caso contrario, diventa estremamente difficile distribuire diverse parti di essa e farle funzionare insieme o in determinati ambienti di test.

Inoltre, deve esistere una procedura di deployment documentata, chiara e facile da seguire. Questa procedura dipende dalla versione; pertanto, anch'essa deve essere inclusa nel configuration management.

Esistono due casi distinti quando si effettua il deployment di una TAS:

1. Primo deployment
2. Deployment di manutenzione - la TAS esiste già e deve essere distribuito un aggiornamento

I rischi legati al primo deployment includono:

- Il tempo totale di esecuzione dei test della test suite può essere superiore al tempo di esecuzione dei test pianificato per il ciclo di test. In questo caso, è importante assicurarsi di pianificare un tempo sufficiente per l'esecuzione completa della test suite prima che inizi il prossimo ciclo di test schedato.
- Esistono problemi di installazione e configurazione degli ambienti di test (ad esempio, configurazione del database e caricamento iniziale, avvio/arresto dei servizi). La TAS ha bisogno di una test fixture (cioè un set di dati predefinito) per creare le precondizioni necessarie affinché i test case automatizzati possano essere eseguiti all'interno dell'ambiente di test.

Per i deployment di manutenzione, ci sono ulteriori considerazioni da fare. La TAS stessa deve evolversi e i suoi aggiornamenti devono essere distribuiti in produzione. Prima di eseguire il deployment di una versione aggiornata della TAS in produzione, è necessario testarla. È quindi necessario controllare le nuove funzionalità, verificare che una test suite possa essere eseguita sulla TAS aggiornata, che i test report possano essere inviati e che non ci siano difetti delle prestazioni o altri problemi di qualità. In alcuni casi, potrebbe essere necessario modificare l'intera test suite per adattarla all'ultima versione della TAS.

4.3 Dipendenze all'interno dell'Ambiente di Test

4.3.1 Definire i componenti della test automation nell'ambiente di test

I componenti della test automation sono tipicamente costituiti da strumenti, macchine virtuali, automation test script, container e configurazioni. L'ambiente di test comprende anche il SUT. I componenti della test automation nell'ambiente di test possono essere definiti come segue:

SUT - È una parte ovvia dell'ambiente di test. Il SUT può essere testato nel suo complesso o suddiviso in sottocomponenti (ad esempio, API, interfaccia Web-based e database).

Piattaforma - La piattaforma descrive dove sono ospitati i componenti della test automation. Include l'infrastruttura del cloud, la rete, le macchine virtuali e i container che possono essere utilizzati per rendere efficiente e portatile la test automation.

Test case e test suite - Descrivono i singoli test case che sono costituiti da passi del test che indirizzano azioni sia automatizzate che manuali. Le test suite descrivono un raggruppamento logico di test case in modo che possano essere eseguiti insieme con efficienza.

Strumenti - I diversi strumenti di test automation vengono utilizzati per diverse ragioni. Una raccolta di strumenti comprenderebbe quelli per la test automation della UI, per il testing degli endpoint delle API, per la generazione di dati, per il monitoraggio, per il requirements management, per il defect management, per gli strumenti di logging e di reporting e per la generazione di tendenze basate sulle metriche.

TAF - Include tutti gli elementi che compongono la progettazione del TAF. I TAF includono driver script, librerie comuni, template per test case automatizzati, script di caricamento/provisioning dei dati, documentazione su come usare i componenti del TAF e tutorial che aiutano i TAE a utilizzare il TAF. Per ulteriori informazioni, consultare il Syllabus CTAL-TAE, sezione 3.1.

La manutenzione dei componenti di test è un aspetto importante, perché un'eccessiva complicazione dell'ambiente di test può comportare ore di tempo indesiderato per correggere i difetti nel TAF invece di beneficiare della soluzione. È importante scoprire la giusta combinazione di strumenti, configurazione e portabilità della piattaforma per rendere i componenti il più possibile riutilizzabili.

4.3.2 Identificare i componenti e le dipendenze dell'infrastruttura della test automation

Ci sono diversi componenti e dipendenze dell'infrastruttura da tenere presenti quando si assembla la test automation. Nel complesso, essi coprono tutti i prerequisiti necessari per eseguire una TAS. I componenti e le dipendenze principali includono:

Macchine host - Possono essere macchine virtuali, server fisici, laptop e dispositivi (ad esempio, tablet e mobile). Su di esse è installato il software di test automation ed è qui che vengono creati ed eseguiti i test script.

Rete - È ciò che dà alla TAS l'accessibilità al SUT. Può anche includere il collegamento in rete di più macchine host per fornire l'esecuzione dei test automatizzati in parallelo. In genere, le macchine host devono essere sulla stessa rete e devono essere configurate correttamente per comunicare tra loro.

Piattaforma - La test automation, come qualsiasi altro software, può essere eseguita su piattaforme cloud o progettata per essere eseguita in container. Finché la piattaforma fornisce le autorizzazioni e l'accessibilità all'OS sottostante, è possibile installare tutti gli strumenti e le dipendenze necessarie.

Dipendenze del software - Per far funzionare correttamente gli strumenti di test automation, è necessario comprendere tutte le altre dipendenze che la test automation ha. Ad esempio, un particolare strumento di test automation potrebbe richiedere che l'ultima versione di un linguaggio di programmazione venga installata prima sulla macchina host. I TAE devono tenere conto di queste dipendenze prima e dopo la selezione di uno strumento.

SUT - Una volta considerati e configurati correttamente i componenti, le dipendenze e l'infrastruttura complessiva, l'ultimo passo è garantire l'accessibilità al SUT. Oltre alla rete, è anche necessario

considerare come interfacciarsi con il SUT. Ad esempio, in un'applicazione Web-based, è necessario installare un browser sulla macchina host. È necessario considerare il tipo e la versione del browser.

4.3.3 Definire i requisiti dei dati e dell'interfaccia della test automation

Due considerazioni che i TAE devono fare prima di sviluppare test automation script sono come pianificano di interfacciarsi con il SUT e quali sono le dipendenze di dati dei test case. Di seguito sono riportati alcuni esempi:

API - Se il SUT utilizza un'API, questa può essere testata a livello di end point invece di richiedere ad una UI di interfacciarsi a livello di applicazione. Ciò potrebbe richiedere una comprensione più approfondita degli end point e delle comunicazioni di dati che sono nascoste da una UI. Il TAE dovrebbe capire in quale ordine chiamare le API per creare un processo di business e come correlare i dati per mantenere intatto il test case. Le API esposte a Internet sono chiamate anche API web o servizi web.

Interfaccia con il database - Alcuni strumenti di test automation sono in grado di interfacciarsi direttamente con il database sottostante del SUT. È possibile scrivere test script per verificare i dati all'interno di colonne e righe, per garantire che le stored procedure e altre regole del database siano configurate correttamente. Per il testing di affidabilità, può essere richiesto che nel database siano già presenti valori di dati specifici.

Compatibilità dell'interfaccia - L'uso del testing di contratti garantisce che due sistemi separati (ad esempio, due microservizi) siano compatibili e in grado di comunicare tra loro. Il testing di contratti può essere consumer-driven, producer-driven e bidirezionale. Ulteriori dettagli possono essere trovati nel Syllabus CTAL-TAE , sezione 5.1.3.

Inoltre, i TAE devono considerare attentamente come si interfacceranno con il SUT e comprendere le dipendenze dei dati all'interno dei test case. Che si tratti di testing tramite API, interfacce con il database o garantire la compatibilità dell'interfaccia tra i sistemi, l'attenzione a queste considerazioni è essenziale per una test automation robusta ed efficace. Affrontando questi fattori in modo ponderato, i TAE possono migliorare l'affidabilità e l'efficienza dei loro effort di automazione, contribuendo in ultima analisi alla qualità e al successo del SUT.

5 Analisi degli Impatti per la Test Automation - 150 minuti (K3)

Parole Chiave

copertura, test report

Obiettivi di Apprendimento per il Capitolo 5:

5.1 Investimenti per la Configurazione e il Mantenimento della Test Automation

CT-TAS -5.1.1 (K3) Mostrare il ritorno dell'investimento per la costruzione di una test automation solution

5.2 Metriche della Test Automation

CT-TAS -5.2.1 (K2) Classificare le metriche per la test automation

5.3 Il Valore della Test Automation a Livello di Progetto e di Organizzazione

CT-TAS -5.3.1 (K3) Identificare le considerazioni organizzative per l'uso della test automation

CT-TAS -5.3.2 (K3) Analizzare le caratteristiche del progetto che aiutano a determinare l'implementazione ottimale degli obiettivi della test automation

5.4 Decisioni Prese dai Report della Test Automation

CT-TAS -5.4.1 (K2) Analizzare i dati dei test report per informare il processo decisionale

5.1 Investimenti per la Configurazione e il Mantenimento della Test Automation

5.1.1 Mostrare il ritorno dell'investimento per la costruzione di una test automation solution

È importante stimare i costi di configurazione e mantenimento della test automation prima di procedere con l'avvio dell'implementazione di un progetto. Oltre ai valori che l'automazione può apportare a un progetto, è utile comprendere il calcolo del ROI del progetto.

Il calcolo del ROI può fornire un feedback significativo in qualsiasi momento del ciclo di vita di un progetto, dimostrando il ritorno di un'attività per l'effort investito. Il calcolo del ROI può essere ulteriormente adattato includendo sia i costi dei tester manuali che quelli dei TAE, semplicemente moltiplicando il tempo speso per i rispettivi costi del lavoro.

Per calcolare il ROI, è necessario determinare l'investimento nella test automation (cioè tempo e costi) e i risparmi ottenuti con essa:

ROI = Risparmi / Investimento

È importante notare che i risparmi e l'investimento possono essere calcolati considerando metriche e dati diversi, in diverse misure e unità. Nell'ambito di questo Syllabus, per dimostrare l'approccio viene utilizzato un modello semplice, con unità di tempo e non di costo. Se alcune attività sono misurate solo in termini di costo, tale importo può essere convertito in tempo utilizzando un tasso specifico per il progetto.

In generale, i risparmi ottenuti con la test automation sono dovuti al fatto che gli stessi test possono essere eseguiti in tempi significativamente più brevi rispetto all'esecuzione manuale. Ciò significa anche che possono essere eseguiti più spesso. Di conseguenza, è possibile aumentare il numero di test eseguiti.

Per calcolare i risparmi, è necessario considerare le seguenti metriche:

- Tempo per eseguire un test case manualmente
- Tempo per eseguire un test case automatizzato
- Numero di test case
- Numero di test run

Per calcolare l'investimento, è necessario considerare le seguenti metriche:

- Tempo per configurare la test automation
- Tempo medio per lo sviluppo di test script automatizzati
- Numero di test script automatizzati implementati
- Tempo medio di manutenzione di un test script automatizzato
- Tempo per eseguire un test script automatizzato
- Percentuale di test script automatizzati falliti
- Numero di test case definiti
- Numero di test run

Adattando il modello delineato allo sviluppo software Agile, è possibile prevedere gli sprint di un progetto come illustrato nel grafico seguente. Utilizzando il grafico, si può determinare lo sprint a partire dal quale la test automation ha reso il suo investimento.

ROI

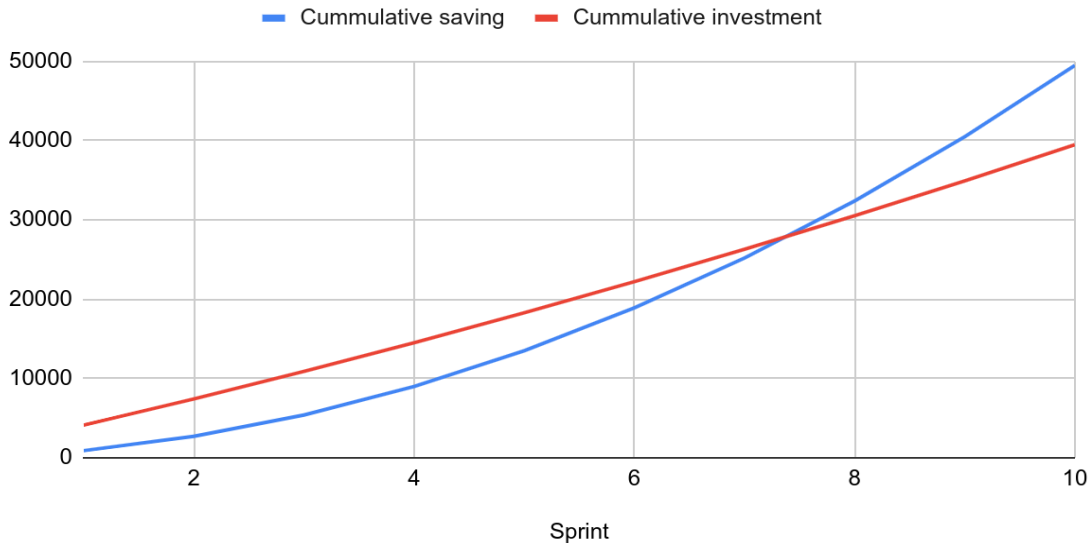


Figura 2: Esempio di calcolo del ROI che mostra il punto di Ritorno dell'investimento

Tenete presente alcune connessioni tra le metriche misurate per calcolare il ROI, tra cui:

- Se la durata pianificata del progetto è inferiore al punto di svolta del ROI, non vale la pena introdurre la test automation. In questo caso, l'esecuzione manuale dei test consente di risparmiare tempo e effort.
- Poiché l'investimento dipende in modo significativo dal tempo di esecuzione dei test automatizzati, l'applicazione della piramide di test e l'implementazione dei test case al giusto livello di test possono ridurre questo tempo di esecuzione e migliorare il ROI.

5.2 Metriche della Test Automation

5.2.1 Classificare le metriche per la test automation

L'analisi delle tendenze basata su dati concreti aiuta il processo decisionale. Raccogliendo le metriche di una TAS, i tester sono in grado di valutare la TAS e di prendere decisioni su:

- Idoneità della TAS al progetto
- Adattabilità della TAS per ampliare le funzionalità per nuove condizioni di test
 - Una modifica in un flusso utente nel SUT
 - Una modifica in come viene eseguito il testing
- Manutenibilità della test automation a causa dei difetti scoperti nella TAS

Il costo della misurazione dovrebbe essere il più basso possibile, e questo spesso si può ottenere automatizzando la raccolta e il reporting delle metriche. Alcuni esempi sono riportati di seguito.

Rapporto di superamento-fallimento

Si tratta di una metrica comune che tiene conto del rapporto tra i test automatizzati superati e i test automatizzati che hanno fallito nel raggiungere il risultato atteso. Il rapporto di superamento-fallimento = il numero di test superati / il numero di test falliti.

Rapporto tra failure e difetti

Un problema comune dei test automatizzati è che molti di essi possono fallire per lo stesso motivo, cioè per un singolo difetto nel SUT. La misurazione del numero di test automatizzati che falliscono per un determinato difetto può aiutare a indicare dove questo può essere un problema. Inoltre, si può raccogliere il numero di failure per difetto e tracciare il motivo del failure: difetto nel sottosistema, nell'intero sistema, problema con i dati di test o con l'infrastruttura.

Tempo di esecuzione della test automation

Una delle metriche più semplici da determinare è il tempo necessario per eseguire i test automatizzati. All'inizio della TAS questo potrebbe non essere importante, ma con l'aumento del numero di test case automatizzati, questa metrica può diventare molto importante. Questa metrica include il tempo di costruzione della TAS.

Numero di test case automatizzati

Questa metrica può essere utilizzata per mostrare i progressi fatti dal progetto di test automation. Ma bisogna tenere conto del fatto che il numero di test case automatizzati non rivela molte informazioni; ad esempio, non indica il livello di copertura.

Copertura funzionale della test automation

Questa copertura indica la percentuale di requisiti funzionali coperti dai test case automatizzati.

Copertura del codice

La copertura del codice tiene traccia di quante linee di codice vengono esercitate dai test di livello inferiore (cioè di componente).

Non esiste una percentuale assoluta che indichi una copertura adeguata, e il 100% di copertura del codice è spesso difficile da raggiungere in qualsiasi software che non sia il più semplice. Tuttavia, è generalmente concordato che una maggiore copertura sia migliore, in quanto aumenta la fiducia nel SUT. Con l'aggiunta di altri test automatizzati di basso livello, ci si aspetta che la copertura del codice aumenti.

5.3 Il Valore della Test Automation a Livello di Progetto e di Organizzazione

5.3.1 Identificare le considerazioni organizzative per l'uso della test automation

Prima di iniziare la test automation su qualsiasi progetto, è necessario identificare quanto segue all'interno dell'organizzazione:

Politiche e pratiche per lo sviluppo software

È bene verificare come lavorano i team di sviluppo e che tipo di documentazione è presente. Qualsiasi documentazione disponibile può essere utile per identificare come la test automation possa essere collegata ai processi dei team di sviluppo. La documentazione può includere le specifiche tecniche del

SUT, gli strumenti software e di sviluppo utilizzati, o qualsiasi politica disponibile sullo sviluppo, come le linee guida per la review del codice, gli standard di codifica definiti e i processi di merge.

Progetti di test automation attivi esistenti e loro stato

Nel caso di un progetto di sviluppo in corso, possono esserci uno o più progetti di sviluppo della TAS in corso da parte di team diversi. Una raccomandazione generale per chi prende le decisioni è di controllare questi progetti esistenti e il loro stato, per vedere se qualcuno di essi rientra negli obiettivi di test definiti per la nuova TAS. Analizzando la soluzione, si può determinare se riutilizzare una delle TAS esistenti o crearne una nuova in base alle esigenze attuali.

Gli esperti in materia dell'organizzazione per la test automation

Si raccomanda di cercare SME che possano assistere nel roll out di una nuova TAS all'interno dell'organizzazione. I SME possono condividere storie e lezioni apprese sulla loro TAS e sul loro roll out. Da ciò si possono identificare i diversi rischi che devono essere considerati o evitati per avere un roll out della TAS di successo.

Disponibilità degli ambienti di test

Nel caso di grandi organizzazioni, una raccomandazione generale è di raccogliere informazioni sugli ambienti di test esistenti, sul loro utilizzo e sulla loro disponibilità. Queste informazioni sono fondamentali per evitare di rovinare il lavoro di altri team implementando una nuova TAS e utilizzando il sistema senza alcuna notifica o accordo. Spesso, le esigenze di un progetto richiedono un nuovo ambiente di test, quindi quelli esistenti possono essere usati come base per crearne uno nuovo e il lavoro può essere svolto più facilmente se i team e le persone responsabili sono identificati e disponibili.

Strumenti di test e licenze

Vale sempre la pena informarsi su quali strumenti e licenze possiede attualmente l'organizzazione. Identificando questi, i costi e le tempistiche, è possibile ridurre la pianificazione di una TAS. Inoltre, gli strumenti per un nuovo progetto potrebbero essere già disponibili. Ad esempio, se il testing su cloud è configurato tramite un fornitore definito e le licenze sono disponibili per il nuovo progetto, non ha senso utilizzare un altro fornitore di cloud. È consigliabile utilizzare gli stessi strumenti e le stesse licenze per ridurre i costi del progetto.

5.3.2 Analizzare le caratteristiche del progetto che aiutano a determinare l'implementazione ottimale degli obiettivi della test automation

Esistono diverse caratteristiche principali del progetto che possono definire un modo ottimale di lavorare e aiutare a definire obiettivi di test automation di successo.

Dominio

È importante capire che ogni dominio si differenzia per normative o standard. Ad esempio, nel dominio del turismo ci sono regolamenti e rischi diversi rispetto a quello della finanza o della sanità. Si raccomanda sempre di verificare i diversi standard e le restrizioni del dominio per assicurarsi che gli obiettivi della test automation pianificati siano conformi.

Piattaforme

In termini di obiettivi della test automation, si sottolinea anche la necessità di valutare quali piattaforme copre il progetto e dove sarebbe vantaggioso eseguire la test automation. La pianificazione della test automation per più piattaforme può essere più difficile, poiché potrebbero essere necessarie più soluzioni

per coprire le diverse piattaforme. In molti casi, ad esempio per il mobile e il web, si possono usare gli stessi strumenti. Ma la pianificazione determinerà la riutilizzabilità della TAS.

Linguaggio di programmazione e stack tecnologico

I dettagli dell'implementazione di un progetto, come il suo linguaggio di programmazione e lo stack tecnologico, determinano anche il tipo di obiettivi della test automation che un'organizzazione dovrebbe definire. È consigliabile utilizzare gli stessi linguaggi di programmazione che gli sviluppatori utilizzano per un determinato progetto. In questo modo, la collaborazione può essere molto più facile e l'apprendimento trasversale, grazie alle review del codice eseguite insieme, migliorerà ulteriormente la qualità del SUT e la conoscenza dei TAE.

Maturità del progetto

Analizzando la maturità del progetto, si possono ricavare informazioni per progettare l'obiettivo ideale della test automation. Identificando i diversi fattori, come il numero di test case, le pipeline di CI/CD esistenti, il numero di tester e di TAE, si possono creare diversi obiettivi di test e una roadmap. Ad esempio, se ci sono molti test case manuali ad alta priorità e il tempo di esecuzione dei test manuali è lungo, è importante puntare prima all'automazione di questi per risparmiare tempo ed effort. Oltre a questi fattori, anche la tempistica del progetto è molto importante. Se il progetto è breve o finirà a breve e non c'è molto tempo per portare a termine le cose, non vale la pena pianificare una TAS grande e robusta. Tuttavia, se si tratta di un progetto greenfield, vale la pena di elaborare una roadmap incrementale passo dopo passo basata sul prodotto minimo realizzabile.

Accettazione dagli stakeholder

Molte volte la test automation non viene utilizzata perché i principali stakeholder non la accettano. Ciò può essere dovuto al timore che la velocità scenda al di sotto del loro target, alle scadenze strette, al budget o ad altre preoccupazioni. Dopo l'analisi della maturità del progetto, se c'è spazio per iniziare l'implementazione di una TAS, uno stakeholder strategico deve identificare i rischi di prodotto, elencare i benefici dell'introduzione della test automation e proporre un piano appropriato. Questo piano deve evidenziare le milestone da raggiungere con e grazie alla test automation e indicare come deve essere migliorata la testabilità del SUT per introdurre con successo una TAS che affronti i rischi identificati. Infine, questo piano deve essere presentato agli stakeholder.

Conoscenza del team ed esperienza rilevante

I fattori più importanti e rilevanti per il successo della test automation sono le competenze e l'esperienza dei tester. È utile lavorare con i tester e utilizzare le loro conoscenze per definire gli obiettivi di test con cui loro e i business analyst si trovano a proprio agio. Spesso i test manager o i test leader creano una matrice di competenze e abilità per identificare le conoscenze disponibili all'interno dei team e, allo stesso tempo, per individuare le lacune. Queste lacune possono essere identificate con obiettivi diversi, come la formazione e l'assegnazione di compiti di implementazione.

Supporto del test management e budgeting

A seconda delle dimensioni e della maturità del progetto, nel definire gli obiettivi di test per la test automation si dovrebbe tenere conto del budget disponibile e del supporto del test management. È importante definire obiettivi di test che possano essere raggiunti, il che a sua volta consentirà il supporto del test management.

Quando si propone al management una strategia di test automation, la documentazione deve essere concisa, definendo chiaramente le lacune attualmente identificate o i costi futuri per la corretta implementazione della test automation. Un elenco di raccomandazioni e dei relativi benefici, che evidenzia il valore di business e la riduzione dei costi, incoraggerà il test management ad approvare lo sviluppo o il miglioramento di una TAS.

Caratteristiche di qualità

La norma ISO/IEC 25010:2011 definisce le caratteristiche di qualità elencate nel Syllabus ISTQB CTFL, sezione 2.2.2. Tipi di test. Queste caratteristiche di qualità possono essere utilizzate per valutare l'attuale TAS o per determinare le metriche che saranno raccolte dalla TAS.

5.4 Decisioni Prese dai Report della Test Automation

5.4.1 Analizzare i dati dei test report per informare il processo decisionale

Il formato e il contenuto di un test automation report possono variare a seconda degli stakeholder che lo ricevono. Può essere creato per gli stakeholder del management, operativi o tecnici. Ulteriori informazioni possono essere trovate nel Syllabus CTAL-TAE, sezione 6.1.3.

Quando si riceve un test automation report, questo può essere incorporato in un test report più ampio, oppure può essere consolidato e poi scalato all'interno della struttura organizzativa.

I diversi stakeholder scoprono valori diversi in un test automation report di questo tipo. Un approccio strategico consiste nell'identificare le metriche chiave che sono importanti per gli stakeholder coinvolti, enfatizzando queste metriche importanti.

I dati raccolti con l'automazione possono aiutare a:

- Identificare le tendenze ed eseguire la root cause analysis
- Spostare gli effort della test automation alla manutenzione
- Spostare gli effort della test automation verso i miglioramenti e l'ulteriore sviluppo di un TAF
- Aggiungere capacità alla TAS complessiva
- Aumentare gli approcci shift-left e shift right al testing
- Espandere la copertura funzionale della test automation negli sprint futuri
- Focalizzarsi maggiormente sui cluster di difetti
- Consigliare agli sviluppatori le aree in cui migliorare il codice
- Consigliare sui processi SDLC complessivi
- Modificare il contenuto e il formato dei futuri test automation report

Sulla base delle informazioni sopra descritte, i TAE, in collaborazione con gli altri stakeholder, possono identificare le lacune e alcuni punti di miglioramento nell'attuale copertura della test automation e nei risultati dei test.

6 Strategie di Implementazione e di Miglioramento per la Test Automation - 150 minuti (K3)

Parole Chiave

copertura, preconditione, test suite

Obiettivi di Apprendimento per il Capitolo 6:

6.1 Transizione delle Attività dal Testing Manuale al Continuous Testing

CT-TAS -6.1.1 (K2) Descrivere i fattori e le attività di pianificazione nella transizione dal testing manuale alla test automation

CT-TAS -6.1.2 (K2) Descrivere i fattori e le attività di pianificazione nella transizione dalla test automation al continuous testing

6.2 Comprensione dei Fattori e delle Attività di Pianificazione nella Transizione dalla Test Automation al Continuous Testing

CT-TAS -6.2.1 (K3) Condurre una valutazione degli strumenti e delle pratiche della test automation per identificare le aree di miglioramento

6.1 Transizione delle Attività dal Testing Manuale al Continuous Testing

6.1.1 Descrivere i fattori e le attività di pianificazione nella transizione dal testing manuale alla test automation

L'opportunità più semplice per passare dal testing manuale alla test automation è di puntare sul regression testing. Una regression test suite cresce man mano che i test funzionali e non funzionali di oggi diventano i regression test di domani. È solo questione di tempo prima che il numero di regression test diventi maggiore del tempo e delle risorse disponibili di un team di test manuale tradizionale.

Costi di transizione

Durante la transizione dal testing manuale al testing automatizzato, il progetto dovrebbe aspettarsi un aumento dei costi, poiché sia il testing manuale che il testing automatizzato si svolgono contemporaneamente. Una volta che i test automatizzati sono considerati in grado di sostituire o sopravanzare adeguatamente le attività di esecuzione dei test manuali, è possibile spostare più effort verso il testing esplorativo e la definizione di ulteriori test case per la test automation, che hanno un costo diverso rispetto al regression testing manuale.

Sovrapposizione funzionale

La sovrapposizione funzionale si verifica quando gli sviluppatori di test script includono gli stessi identici passi del test automation in diversi test case. Ad esempio, la maggior parte dei test case inizia con una sequenza di login di passi del test. Questa può includere l'inserimento di un nome utente, di una password e la selezione di un pulsante di login. Questa aggiunta in ogni test case aumenta le attività di manutenzione. Se viene aggiunto un ulteriore passo del test al processo di login, la stessa modifica dovrà essere aggiornata in ogni test case. Un approccio migliore consiste nel creare un componente di test automation ripetibile dal processo di login e fare in modo che tutti i test case facciano riferimento a tale funzionalità. Vedere il Syllabus CTAL-TAE, sezione 3.1.5 per i dettagli sul modello flow model.

Condivisione dei dati

I test spesso condividono i dati di test. Questo può accadere quando i test utilizzano lo stesso record di dati di test per eseguire diverse funzionalità del SUT. Un esempio potrebbe essere il test case "A", che verifica le ferie disponibili di un dipendente, mentre il test case "B" verifica quali corsi il dipendente ha seguito come parte dei suoi obiettivi di sviluppo professionale. Ogni test case utilizza lo stesso dipendente, ma verifica parametri diversi. In un ambiente di test manuale, i dati di test del dipendente verrebbero in genere duplicati molte volte in ogni test case manuale che verifica i dati di test del dipendente in questione. Tuttavia, in un test automatizzato, i dati di test condivisi dovrebbero, ove possibile e fattibile, essere archiviati e accessibili da un'unica fonte per evitare la duplicazione o l'introduzione di errori.

Interdipendenza dei test

Durante l'esecuzione di regression test complessi, un test può avere una dipendenza da uno o più altri test. Questa eventualità può essere piuttosto comune. Ad esempio, un nuovo "Ordine ID" viene creato come risultato di un passo del test. I test successivi potrebbero voler verificare che: a) il nuovo ordine venga visualizzato correttamente nel sistema, b) sia possibile modificare l'ordine o c) l'eliminazione dell'ordine abbia successo. In ogni caso, il valore "Ordine ID" creato dinamicamente nel primo test deve essere catturato per essere riutilizzato dai test successivi. A seconda della progettazione della TAS, è possibile risolvere questo problema. Se "l'Ordine ID" non può essere trovato dai test case successivi, questi falliranno.

Precondizioni per l'esecuzione dei test

Troppo spesso i TAE iniziano immediatamente lo sviluppo di test script senza prima comprendere le precondizioni necessarie per garantire che un test case possa essere eseguito in modo affidabile. Questo può diventare estremamente impegnativo quando si cerca di eseguire un test case sullo stesso SUT in più ambienti di test. Esempi di precondizioni sono i nomi utente, i ruoli degli account, i valori delle tabelle di dati e altre voci di dati specifiche che rendono il test case ripetibile. Una buona strategia includerà un'analisi preliminare per capire quali informazioni devono esistere nel SUT prima di automatizzare un particolare test case. Inoltre, la strategia può essere migliorata automatizzando la creazione di precondizioni prima dell'esecuzione reale dei test case, per risparmiare tempo. Ciò può includere l'esecuzione di test script di precondizioni che popolano il SUT da UI o processi batch automatizzati che caricano i dati di test a un database.

Copertura funzionale

Identificate le lacune funzionali nel testing che possono essere candidate alla test automation, come spiegato nel Capitolo 3. Il 100% dei test case manuali che sono automatizzati non rappresenta il 100% di tutti i possibili test case che possono essere automatizzati con gli strumenti di test. Man mano che un maggior numero di test viene automatizzato, i tester recuperano tempo di esecuzione dei test, che possono utilizzare per identificare ulteriori test del SUT per aumentare la copertura.

Test eseguibili

Prima di convertire un regression test manuale in un regression test automatizzato, è importante verificare che il regression test manuale funzioni correttamente. In questo modo si ottiene il punto di partenza corretto per garantire il successo della conversione in un regression test automatizzato. Se il regression test manuale non viene eseguito correttamente, può essere perché è stato scritto male, perché utilizza dati di test non validi, perché non è aggiornato o non è sincronizzato con il SUT attuale o perché c'è un difetto nel SUT. Automatizzare il test prima di aver compreso e/o risolto la root cause del failure creerà un regression test automatizzato non-funzionale, che è dispendioso e improduttivo. È importante dimostrare la funzionalità equivalente che i nuovi test automatizzati apportano, per trasmettere fiducia nei test automatizzati che sostituiranno i vecchi test manuali.

6.1.2 Descrivere i fattori e le attività di pianificazione nella transizione dalla test automation al continuous testing

Il continuous testing prevede l'utilizzo delle risorse di test molto più frequentemente rispetto ai SDLC tradizionali. Ciò avviene iniziando a eseguire costantemente test suite subito dopo che le modifiche al codice sono state apportate e rese disponibili negli ambienti di test. Questo aiuta ad ottenere un feedback immediato e a ridurre i costi per i difetti, individuandoli in una fase più precoce del processo. Questo può essere gestito con strumenti di sviluppo sofisticati e con la volontà di spostare il testing nelle prime fasi del processo di costruzione.

Adattare la TAS per il continuous testing richiede quanto segue:

- Le test suite devono essere modificate per essere eseguite sulla TAS aggiornata: apportare le modifiche necessarie alle test suite e testarle prima di distribuirle sulla TAS
- Gli stub, i driver e le interfacce utilizzate nel testing devono essere modificati per adattarsi alla TAS aggiornata: apportare le modifiche necessarie alla test harness e testarla prima di distribuirle sulla TAS
- L'infrastruttura deve essere modificata per accogliere la TAS aggiornata: fare una valutazione dei componenti dell'infrastruttura che devono essere modificati
- La TAS aggiornata presenta difetti aggiuntivi o difetti delle prestazioni: eseguire un'analisi dei rischi rispetto ai benefici. Se i difetti scoperti rendono impossibile aggiornare la TAS, potrebbe essere meglio non procedere con l'aggiornamento o aspettare una versione successiva della TAS. Se i difetti sono trascurabili rispetto ai benefici derivanti dalla loro correzione, la TAS può comunque essere aggiornata
- Assicuratevi di creare release note dei difetti conosciuti per informare i TAE e gli altri stakeholder e cercate di ottenere una stima di quando i difetti saranno risolti

Tutti i punti citati in questa sezione diventano particolarmente importanti quando si utilizza il CI/CD. Mettere insieme le pipeline e automatizzare il processo di costruzione sono un adattamento naturale alla TAS in fase di sviluppo. Se lo strumento di orchestrazione della build si trova nell'ambiente di test corretto, la pipeline può essere estesa per includere test automatizzati per verificare il SUT subito dopo il deployment. Ciò richiede che lo strumento di test automation sia configurato correttamente e possa accedere al SUT, oltre a raggiungere il repository del codice e ad accedere agli script di provisioning dei dati.

6.2 Strategia di Test Automation in Tutta l'Organizzazione

6.2.1 Condurre una valutazione delle risorse e delle pratiche della test automation per identificare le aree di miglioramento

Come per qualsiasi altra attività di sviluppo, è importante avere una strategia per mettere in pausa lo sviluppo della TAS e cercare opportunità per rifattorizzare la soluzione. Le aree da monitorare sono l'implementazione iniziale, la manutenzione e la capacità di valutare la soluzione dal punto di vista della ripetibilità. Alcune buone categorie da tenere sotto controllo sono: quante ore vengono spese per sviluppare la TAS, quante ore vengono spese per correggere la TAS e quanto tempo risparmiano i tester rispetto al testing manuale. Un indicatore che qualcosa non va è se il tempo di manutenzione della TAS è maggiore del tempo del testing manuale.

Prima di iniziare il primo deployment di una TAS, è importante assicurarsi che possa essere eseguita nel proprio ambiente, che sia isolata da modifiche casuali e che i test case possano essere aggiornati e gestiti. Sia la TAS che la sua infrastruttura devono essere mantenute. Nel caso del primo deployment, sono necessari i seguenti passi fondamentali:

- Gli strumenti di copertura del codice indicano quanta parte del codice viene esercitata dalla test suite di componente e dove esistono lacune che possono essere coperte da test di componente aggiuntivi
- La copertura funzionale del SUT può essere determinata creando una matrice di tracciabilità dei requisiti, che scopre quali funzionalità non sono ancora coperte da alcun test case
- Definire un utilizzo coerente dell'infrastruttura della TAS tra i progetti o l'intera organizzazione
- Sviluppare una strategia coerente di configuration management per le test suite
- Creare linee guida comuni per lo sviluppo della TAS utilizzando le best practice descritte nel syllabus CTAL-TAE, sezione 3.1.4
- Implementare le precondizioni. Spesso un test non può essere eseguito prima di aver stabilito delle precondizioni. Queste precondizioni possono includere la selezione del database corretto o dei dati di test da cui effettuare il test, oppure l'impostazione di valori o parametri iniziali. Molti di questi passi di inizializzazione necessari per stabilire le precondizioni di un test possono essere automatizzati. Questo consente di ottenere una soluzione più sicura e affidabile quando questi passi non possono essere saltati prima dell'esecuzione dei test. Man mano che i regression test vengono convertiti nella test automation, queste precondizioni devono far parte del processo di test automation.

Quando si effettuano aggiornamenti incrementali della TAS per nuove funzionalità o per manutenzione, si dovrebbe tenere conto di quanto segue:

- Valutare gli aggiornamenti degli strumenti di test o altri strumenti di test più recenti che possano fornire capacità estese per la TAS
- Valutare i modi per ottimizzare ulteriormente le funzionalità e le prestazioni della TAS
- Identificare le opportunità di scomporre e modularizzare ulteriormente i test script per migliorare la riutilizzabilità
- Garantire la conoscenza e la consapevolezza dei componenti riutilizzabili e del loro utilizzo coerente
- Raccogliere prove su potenziali aree di miglioramento e fornire raccomandazioni e i relativi benefici
- Valutare e correggere le aree di sovrapposizione funzionale. Quando si automatizzano i regression test esistenti, è buona norma identificare qualsiasi sovrapposizione funzionale esistente tra i test case e, ove possibile, riutilizzare i componenti della test automation sviluppati in precedenza.
- Valutare i test case manuali aggiuntivi per individuare l'opportunità di automatizzarli e creare elementi di backlog per l'implementazione
- Evidenziare le opportunità di miglioramento della progettazione dei test e del management dei dati di test
- Assicurarsi che tutte le test suite esistenti siano adattate all'ultima versione della TAS
- Se la test automation causa code nella pipeline, ridurre l'ambito dei test integrati a quelli più critici, cioè creare una smoke test suite. La regression test suite più ampia può essere attivata separatamente o eseguita su richiesta.

7 Riferimenti

Standard

Gli standard per la test automation includono, ma non si limitano a:

L'Automatic Test Markup Language (ATML) dell'IEEE (Institute of Electrical and Electronics Engineers) è costituito da:

- IEEE Std 1671.1: Test Description
- IEEE Std 1671.2: Instrument Description
- IEEE Std 1671.3: UUT Description
- IEEE Std 1671.4: Test Configuration Description
- IEEE Std 1671.5: Test Adaptor Description
- IEEE Std 1671.6: Test Station Description
- IEEE Std 1641: Signal and Test Definition
- IEEE Std 1636.1: Test Results

ISO/IEC 30130:2016 (E) Software engineering – Capabilities of software testing tools

The Testing and Test Control Notation (TTCN-3) di ETSI (European Telecommunication Standards Institute) e ITU (International Telecommunication Union) è costituito da:

- ES 201 873-1: TTCN-3 Core Language
- ES 201 873-2: TTCN-3 Tabular Presentation Format (TFT)
- ES 201 873-3: TTCN-3 Graphical Presentation Format (GFT)
- ES 201 873-4: TTCN-3 Operational Semantics
- ES 201 873-5: TTCN-3 Runtime Interface (TRI)
- ES 201 873-6: TTCN-3 Control Interface (TCI)
- ES 201 873-7: Using ASN.1 with TTCN-3
- ES 201 873-8: Using IDL with TTCN-3
- ES 201 873-9: Using XML with TTCN-3
- ES 201 873-10: TTCN-3 Documentation
- ES 202 781: Extensions: Configuration and Deployment Support
- ES 202 782: Extensions: TTCN-3 Performance and Real-Time Testing
- ES 202 784: Extensions: Advanced Parameterization
- ES 202 785: Extensions: Behavior Types
- ES 202 786: Extensions: Support of interfaces with continuous signals
- ES 202 789: Extensions: Extended TRI

L'UML Testing Profile (UTP) di OMG (Object Management Group) specifica i concetti di specifica dei test per:

- Test Architecture
- Test Data
- Test Behavior
- Test Logging
- Test Management

Documenti ISTQB®

Identificatore	Riferimento
ISTQB-AL-TM	ISTQB Certified Tester, Advanced Level Syllabus, Test Manager, Version 2.0, October 2012, disponibile su [ISTQB-Web]
ISTQB-EL-TM-MTT	ISTQB Certified Tester, Expert Level Test Management Managing the Test Team, Version 2.0, November 2011, disponibile su [ISTQB-Web]
ISTQB-FL	ISTQB Certified Tester, Foundation Level Syllabus, Version 4.0, April 2023, disponibile su [ISTQB-Web]
ISTQB-PT	ISTQB Certified Tester, Performance Testing Syllabus, December 2018, disponibile su [ISTQB-Web]
ISTQB-TAE	ISTQB Certified Tester, Test Automation Engineering Syllabus, February 2024, disponibile su [ISTQB-Web]
ISTQB-Glossary	ISTQB Glossary of Terms, disponibile online su [ISTQB-Web]

Libri

Paul Baker, Zhen Ru Dai, Jens Grabowski e Ina Schieferdecker, "Model-Driven Testing: Using the UML Testing Profile", edizione Springer 2008, ISBN-10: 3540725628, ISBN-13: 978-3540725626
Efriede Dustin, Thom Garrett, Bernie Gauf, "Implementing Automated Software Testing: how to save time and lower costs while raising quality", Addison-Wesley, 2009, ISBN 0-321-58051-6
Efriede Dustin, Jeff Rashka, John Paul, "Automated Software Testing: introduction, management, and performance", Addison-Wesley, 1999, ISBN-10: 0201432870, ISBN-13: 9780201432879
Mark Fewster, Dorothy Graham, "Experiences of Test Automation: Case Studies of Software Test Automation", Addison-Wesley, 2012
Mark Fewster, Dorothy Graham, "Software Test Automation: Effective use of test execution tools", ACM Press Books, 1999, ISBN-10: 0201331403, ISBN-13: 9780201331400
Boby Jose, "Test Automation, a manager's guide", September 2021, ISBN: 9781780175478
James D. McCaffrey, ".NET Test Automation Recipes: A Problem-Solution Approach", APRESS, 2006 ISBN-13:978-1-59059-663-3, ISBN-10:1-59059-663-3
Daniel J. Mosley, Bruce A. Posey, "Just Enough Software Test Automation", Prentice Hall, 2002, ISBN-10: 0130084689, ISBN-13: 9780130084682
Casey Rosenthal, "Chaos Engineering: System Resiliency in Practice by Casey Rosenthal", April 2020, ISBN-13: 1492043869
Colin Willcock, Thomas Deiß, Stephan Tobies e Stefan Keil, "An Introduction to TTCN-3" Wiley, 2a edizione 2011, ISBN-10: 0470663065, ISBN-13: 978-0470663066

Articoli

Robert V. Binder, Suzanne Miller, "Five Keys to Effective Agile Test Automation for Government Programs" August 24, 2017, Software Engineering Institute, Carnegie Mellon University, https://resources.sei.cmu.edu/asset_files/Webinar/2017_018_101_503516.pdf
DoD CIO, Modern Software Practices "DevSecOps Fundamentals Guidebook: Activities & Tools", Version 2.2, May 2023, https://dodcio.defense.gov/Portals/0/Documents/Library/DevSecOpsActivitesToolsGuidebookTables.pdf?ver=_Sylg1WJB9K0Jxb2XTvzDQ%3d%3d
Naveen Jayachandran, "Understanding roi metrics for software test automation", 2005, https://digitalcommons.usf.edu/cgi/viewcontent.cgi?article=3937&context=etd
Thomas Pestak, William Rowell, PhD, "Automated Software Testing Practices and Pitfalls", September 2018, https://www.afit.edu/stat/statcoe_files/Automated%20Software%20Testing%20Practices%20and%20Pitfalls%20Rev%201.pdf
Andrew Pollner, Jim Simpson, Jim Wisnowski, "Automated Software Testing Implementation Guide for Managers and Practitioners", October 2018, https://www.afit.edu/stat/statcoe_files/0214simp%20%20AST%20IG%20for%20Managers%20and%20Practitioners.pdf

8 Appendice A - Obiettivi di Apprendimento/Livello Cognitivo di Conoscenza

I seguenti obiettivi di apprendimento sono definiti come applicabili a questo Syllabus. Ogni argomento del Syllabus sarà esaminato in base al relativo obiettivo di apprendimento.

Gli obiettivi di apprendimento iniziano con un verbo di azione corrispondente al livello cognitivo di conoscenza, come elencato di seguito.

Livello 2: Comprendere (K2)

Il candidato è in grado di selezionare le ragioni o le spiegazioni delle affermazioni relative all'argomento ed è in grado di riassumere, confrontare, classificare e fornire esempi per il concetto del testing.

Verbi di azione: Classificare, confrontare, differenziare, distinguere, spiegare, fornire esempi, interpretare, riassumere.

Esempi	Note
Classificare gli strumenti di test in base al loro scopo e alle attività di test che supportano.	
Confrontare i diversi livelli di test.	Può essere utilizzato per cercare somiglianze, differenze o entrambe le cose.
Differenziare il testing dal debugging.	Cerca le differenze tra i concetti.
Distinguere tra rischi di progetto e rischi di prodotto.	Consente di classificare separatamente due (o più) concetti.
Spiegare l'impatto del contesto sul processo di test.	
Fornire esempi del perché il testing è necessario.	
Ricavare la root cause dei difetti da un determinato profilo di failure.	
Riassumere le attività del processo di review del prodotto di lavoro.	

Livello 3: Applicare (K3)

Il candidato è in grado di eseguire una procedura quando si trova di fronte a un compito che gli è familiare, oppure è in grado di selezionare la procedura corretta e applicarla a un determinato contesto.

Verbi di azione: Applicare, implementare, preparare, utilizzare

Esempi	Note
Applicare l'analisi ai valori limite per ricavare i test case da determinati requisiti.	Dovrebbe riferirsi a una procedura / tecnica / processo ecc.
Implementare i metodi di raccolta delle metriche per supportare i requisiti tecnici e di management.	
Preparare l'ambiente di test automation.	
Utilizzare la tracciabilità per monitorare l'avanzamento del test, in modo da poter verificare la completezza e la consistenza con gli obiettivi del test, la strategia di test e il test plan.	Potrebbe essere utilizzato in un LO che richiede che il candidato sia in grado di utilizzare una tecnica o una procedura. Simile ad "applicare".

Riferimento

(Per i livelli cognitivi degli obiettivi di apprendimento)

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon

9 Appendice B - Matrice di Tracciabilità dei Business Outcome agli Obiettivi di Apprendimento

Questo capitolo elenca la tracciabilità tra i Business Outcome del Test Automation Strategy Specialist e gli Obiettivi di Apprendimento del Test Automation Strategy Specialist.

Business Outcome Test Automation Strategy Specialist			B 0 1	B 0 2	B 0 3	B 0 4	B 0 5	B 0 6	B 0 7	B 0 8	B 0 9	B 1 0	B 1 1	B 1 2	B 1 3	B 1 4	B 1 5
Capitolo 1	Introduzione e Obiettivi della Strategia di Test Automation																
1.1	Fattori di Successo di un Progetto di Test Automation																
1.1.1	Spiegare gli obiettivi e la rilevanza della test automation	2	X														
1.1.2	Identificare i fattori tecnici di successo di un progetto di test automation	2	X														
1.1.3	Riassumere i criteri di investimento appropriati nella selezione dei progetti candidati per la test automation	2	X														
Capitolo 2	Risorse per la Test Automation																
2.1	Costi e Rischi dell'Implementazione di una Test Automation Solution																
2.1.1	Confrontare soluzioni tecniche alternative in termini di costi di proprietà	2		X													
2.1.2	Spiegare le considerazioni sul modello di licenza per gli strumenti di test automation	2		X													
2.1.3	Fornire esempi di fattori da considerare quando si definisce di una strategia di test automation	2		X													
2.2	Ruoli e Responsabilità all'interno della Test Automation																
2.2.1	Riassumere i ruoli e le competenze necessarie per una test automation solution di successo	2			X												

Business Outcome Test Automation Strategy Specialist			B 0 1	B 0 2	B 0 3	B 0 4	B 0 5	B 0 6	B 0 7	B 0 8	B 0 9	B 1 0	B 1 1	B 1 2	B 1 3	B 1 4	B 1 5
Capitolo 3	Preparazione per la Test Automation																
3.1	Integrazione tra i Livelli di Test																
3.1.1	Differenziare tra le distribuzioni della test automation	2				X											
3.1.2	Selezionare un approccio di test automation basato sull'architettura del system under test	3				X											
3.1.3	Dimostrare come ottimizzare la distribuzione della test automation per ottenere approcci shift-left e shift right	2				X											
3.2	Considerazioni Strategiche nei Diversi Modelli del Ciclo di Vita dello Sviluppo Software																
3.2.1	Spiegare come i progetti di test automation siano conformi ai modelli del ciclo di vita dello sviluppo software legacy	2					X										
3.2.2	Spiegare come i progetti di test automation siano conformi alle best practice di sviluppo software Agile che supportano la test automation	2					X										
3.2.3	Preparare i progetti di test automation per conformarsi alle best practice di DevOps che supportano la test automation nel continuous testing	3					X										
3.3	Applicabilità e Fattibilità della Test Automation																
3.3.1	Spiegare i criteri per determinare l'idoneità dei test per la test automation	2						X									
3.3.2	Identificare le sfide che solo la test automation può affrontare	2						X									
3.3.3	Identificare le condizioni di test difficili da automatizzare	2						X									

Business Outcome Test Automation Strategy Specialist			B 0 1	B 0 2	B 0 3	B 0 4	B 0 5	B 0 6	B 0 7	B 0 8	B 0 9	B 1 0	B 1 1	B 1 2	B 1 3	B 1 4	B 1 5
Capitolo 4	Strategie Organizzative di Deployment e Release per la Test Automation																
4.1	Pianificazione della Test Automation Solution																
4.1.1	Identificare i modi in cui la test automation supporta un time to market più breve	2							X								
4.1.2	Identificare i modi in cui la test automation aiuta a verificare i difetti segnalati in base ai requisiti	2							X								
4.1.3	Definire approcci che consentano lo sviluppo di scenari rilevanti dal punto di vista operativo per la test automation	2							X								
4.2	Strategie di Deployment																
4.2.1	Definire una strategia di deployment della test automation	2								X							
4.2.2	Identificare i rischi della test automation nel deployment	2								X							
4.2.3	Definire approcci per mitigare i rischi del deployment dei test	2								X							
4.3	Dipendenze all'interno dell'Ambiente di Test																
4.3.1	Definire i componenti della test automation nell'ambiente di test	2									X						
4.3.2	Identificare i componenti e le dipendenze dell'infrastruttura della test automation	2									X						
4.3.3	Definire i requisiti dei dati e dell'interfaccia della test automation per l'integrazione all'interno del system under test	2									X						
Capitolo 5	Analisi degli Impatti per la Test Automation																

Business Outcome Test Automation Strategy Specialist		B 0 1	B 0 2	B 0 3	B 0 4	B 0 5	B 0 6	B 0 7	B 0 8	B 0 9	B 1 0	B 1 1	B 1 2	B 1 3	B 1 4	B 1 5
5.1	Investimenti per la Configurazione e il Mantenimento della Test Automation															
5.1.1	Mostrare il return on investment per la costruzione di una test automation solution	3									X					
5.2	Metriche della Test Automation															
5.2.1	Classificare le metriche per la test automation	2										X				
5.3	Il Valore della Test Automation a Livello di Progetto e di Organizzazione															
5.3.1	Identificare le considerazioni organizzative per l'uso della test automation	3											X			
5.3.2	Analizzare le caratteristiche del progetto che aiutano a determinare l'implementazione ottimale degli obiettivi della test automation	3											X			
5.4	Decisioni Prese dai Report della Test Automation															
5.4.1	Analizzare i dati dei test report per informare il processo decisionale	2												X		
Capitolo 6	Strategie di Implementazione e di Miglioramento per la Test Automation															
6.1	Transizione delle Attività dal Testing Manuale al Continuous Testing															
6.1.1	Descrivere i fattori e le attività di pianificazione nella transizione dal testing manuale alla test automation	2													X	
6.1.2	Descrivere i fattori e le attività di pianificazione nella transizione dalla test automation al continuous testing	2													X	
6.2	Strategia di Test Automation in Tutta l'Organizzazione															

Business Outcome Test Automation Strategy Specialist			B 0 1	B 0 2	B 0 3	B 0 4	B 0 5	B 0 6	B 0 7	B 0 8	B 0 9	B 1 0	B 1 1	B 1 2	B 1 3	B 1 4	B 1 5
6.2.1	Condurre una valutazione delle risorse e delle pratiche della test automation per identificare le aree di miglioramento	3															X

10 Appendice C – Release Note

L'ISTQB® Test Automation Strategy Syllabus 2024 è un nuovo Syllabus ISTQB che combina gli aspetti strategici della precedente versione del Syllabus ISTQB Test Automation Engineer del 2016 con ulteriori aggiornamenti e le attuali best practice per l'implementazione e la misurazione del successo della test automation. Per questo motivo, non sono presenti release note dettagliate per capitolo e per paragrafo.

11 Appendice D - Termini Specifici del Dominio

Termine	Definizione
canary release	Una strategia di deployment e di testing volta a ridurre il rischio e a verificare il nuovo software rilasciandolo solo a pochi utenti.
container	Un'unità di software che impacchetta il codice e le sue dipendenze, in modo che il software venga eseguito in modo rapido e affidabile in tutti gli ambienti.
DevOps	Una metodologia che integra e automatizza il lavoro dello sviluppo software e delle operazioni IT per migliorare e abbreviare il ciclo di vita dello sviluppo software.
modello flow model	Una visione di alto livello del dominio di lavoro, dei suoi componenti e delle interconnessioni tra di essi.

12 Indice

Tutti i termini sono definiti nel Glossario ISTQB® (<http://glossary.istqb.org/>).

approccio di test automation; 23	system under test; 13; 23; 30
canary release; 63	TAE; 8; 9; 16; 17; 19; 22; 27; 32; 33; 34; 36; 37; 39; 46; 48; 51; 53
componente; 23; 24; 25; 26; 27; 30; 31; 41; 48; 51	TAF; 17; 27; 33; 36; 46
condizione di test; 23	TAS; 3; 8; 10; 15; 17; 18; 19; 20; 22; 23; 24; 26; 30; 33; 34; 35; 36; 38; 40; 41; 43; 44; 45; 46; 47; 48; 50; 51
container; 63	test automation solution; 10; 17; 18; 38
copertura; 16; 22; 25; 26; 28; 38; 41; 46; 47; 49; 51	test doppio; 23
DevOps; 13; 23; 27; 58; 63	test report; 38; 46
livello di test; 23; 24; 40	test suite; 28; 31; 32; 35; 47; 48; 51
modello flow model; 48; 63	testing confermativo; 30
piramide di test; 23; 40	testing delle API; 23; 24; 25
precondizione; 47; 49	testing di componente; 23; 24; 26
quality gate; 30; 31	testing di contratti; 23; 24; 25; 26; 37
shift left; 13; 23; 26; 31; 46	
shift right; 13; 23; 26; 46	