International Software Testing Qualifications Board

Certified Tester

Foundation Level Specialist Syllabus Performance Testing

Versione 2018

Realizzato da

American Software Testing Qualifications Board

e

German Testing Board

Performance Testing Working Group:

Graham Bath
Rex Black
Alexander Podelko
Andrew Pollner
Randy Rice

Cronologia delle revisioni

Versione	Data	Osservazioni
Alpha V04	13 dicembre 2016	Versione per la riunione di New York
Alpha V05	18 dicembre 2016	Dopo la riunione di New York
Alpha V06	23 dicembre 2016	Rielaborazione C. 4 Rinumerazione e perfezionamento dei LO, come concordato a New York
Alpha V07	31 dicembre 2016	Aggiunta commenti dell'autore
Alpha V08	domenica 12 febbraio 2017	Versione pre-Alpha
Alpha V09	16 aprile 2017	Versione pre-Alpha
Revisione Alpha V10	28 giugno 2017	Per la revisione Alpha
V2017v1	27 novembre 2017	Per la revisione Alpha
V2017v2	15 dicembre 2017	Aggiornamenti Alpha
V2017v3	15 gennaio 2018	Modifica tecnica
V2017v4	23 gennaio 2018	Revisione del glossario
V2018 b1	1 marzo 2018	Candidato beta per l'ISTQB
V2018 b2	17 maggio 2018	Pubblicazione beta per l'ISTQB
V2018 b3	25 Agosto 2018	Commenti di revisione beta integrati per la versione pubblicata
Versione 2018	9 dicembre 2018	Pubblicazione GA ISTQB

Sommario

Ringraziamenti

Questo documento è stato redatto dall'American Software Testing Qualifiche Board (ASTQB) e dal German Testing Board (GTB):

Graham Bath (GTB, Co-Presidente del gruppo di lavoro)
Rex Black
Alexander Podelko (CMG)
Andrew Pollner (ASTQB, Co-Presidente del gruppo di lavoro)
Randy Rice

Il team principale ringrazia il team di revisione per i loro suggerimenti e spunti. L'ASTQB desidera ringraziare il Computer Measurement Group (CMG) per i loro contributi allo sviluppo di questo syllabus.

Le seguenti persone hanno partecipato alla revisione, all'osservazione o al voto di questo syllabus o delle versioni precedenti:

Dani Almog
Sebastian Chece
Todd DeCapua
Wim Decoutere
Frans Dijkman
Jiangru Fu
Matthias Hamburg
Ágota Horváth
Mieke Jungebload
Beata Karpinska
Gábor Ladányi
Kai Lepler
Ine Lutterman
Marek Majernik
Stefan Massonet
Judy McKay
Gary Mogyorodi
Joern Muenzel
Petr Neugebauer
Ingvar Nordström
Meile Posthuma
Michaël Pilaeten
Filip Rechtoris
Adam Roman
Dirk Schweier
Marcus Seyfert
Péter Sótér
Michael Stahl
Jan Stiller
Federico Toledo
Andrea Szabó
Yaron Tsubery
Stephanie Ulrich
Mohit Verma
Armin Wachter
Huaiwen Yang
Ting Yang

Questo documento è stato pubblicato ufficialmente dall'ISTQB il 2 novembre 2018.

0. Introduzione al syllabus

0.1 Scopo del presente documento

Questo syllabus costituisce la base per la certificazione sul Performance Testing di livello Foundation. L'ASTQB® e il GTB® forniscono questo syllabus come segue:

1. Ai Board Nazionali, da tradurre nella loro lingua e per accreditare i fornitori di formazione. I Board Nazionali hanno facoltà di adattare il syllabus alle loro esigenze linguistiche particolari e modificare i riferimenti per adattarli alle pubblicazioni locali.
2. Agli Exam Board, per elaborare domande d'esame nella loro lingua locale adattate agli obiettivi di apprendimento per ciascun syllabus.
3. Agli enti formativi, per produrre materiale didattico e determinare metodi di insegnamento appropriati.
4. Ai candidati alla certificazione, per prepararsi all'esame (come parte di un corso di formazione o in modo indipendente).
5. Alla comunità internazionale di ingegneria del software e dei sistemi, per far avanzare la professione di tester di software e sistemi e come base per libri e articoli.

L'ASTQB e il GTB possono consentire ad altri enti di utilizzare questo syllabus per altri scopi, a condizione che chiedano e ottengano un'autorizzazione scritta preventiva.

0.2 La certificazione specialistica sul Performance Testing di livello Foundation

La certificazione specialistica di livello Foundation è rivolta a chiunque, coinvolto nei test software, desideri ampliare la propria conoscenza del Performance Testing o a chiunque desideri iniziare una carriera specialistica nel settore del Performance Testing. La certificazione si rivolge anche a chiunque sia coinvolto nell'ingegneria prestazionale e desideri acquisire una migliore comprensione del Performance Testing.

Il syllabus considera i seguenti aspetti principali del Performance Testing:

- Aspetti tecnici
- Aspetti basati sul metodo
- Aspetti organizzativi

Le informazioni sul Performance Testing descritte nel syllabus per Technical Test Analyst di livello Advanced di ISTQB® [ISTQB_ALTTA_SYL] sono coerenti e vengono sviluppate da questo syllabus.

0.3 Risultati aziendali

Questa sezione elenca i risultati aziendali attesi da un candidato che ha conseguito la certificazione Performance Testing di livello Foundation.

PTFL-1 Comprendere i concetti di base del Performance Testing e di efficienza di performance
PTFL-2 Definire rischi, obiettivi e requisiti di performance per soddisfare le esigenze e le aspettative dei soggetti interessati (stakeholder)
PTFL-3 Comprendere le metriche di performance e come raccogliere i dati
PTFL-4 Sviluppare un piano di Performance Testing per raggiungere obiettivi e requisiti dichiarati
PTFL-5 Progettare, realizzare ed eseguire il Performance Testing di base
PTFL-6 Analizzare i risultati di un Performance Testing e indicarne le implicazioni per i diversi stakeholder
PTFL-7 Spiegare ai diversi stakeholder il processo, la logica, i risultati e le implicazioni del Performance Testing
PTFL-8 Comprendere le categorie e gli utilizzi per la scelta di strumenti e i criteri di performance
PTFL-9 Determinare come le attività di Performance Testing siano in linea con il ciclo di vita del software

0.4 Obiettivi di apprendimento esaminabili

Gli obiettivi di apprendimento sono alla base dei risultati aziendali e vengono utilizzati per creare l'esame per il conseguimento della Certificazione di Performance Testing di livello Foundation. Gli obiettivi di apprendimento sono assegnati a un livello cognitivo di conoscenza (livello K).

Un livello K, o livello cognitivo, viene utilizzato per classificare gli obiettivi di apprendimento in base alla tassonomia rivista di Bloom [Anderson01]. L'ISTQB® utilizza questa tassonomia per progettare i suoi esami di studio.

Questo syllabus considera quattro diversi livelli K (da K1 a K4):

Livello K	**Parola chiave**	**Descrizione**
1	Ricordare	Il candidato deve ricordare o riconoscere un termine o un concetto.
2	Comprendere	Il candidato deve selezionare una spiegazione per una dichiarazione relativa allo specifico argomento.
3	Applicare	Il candidato deve selezionare la corretta applicazione di un concetto o di una tecnica e applicarla a un determinato contesto.

4	Analizzare	Il candidato è in grado di separare le informazioni relative a una procedura o tecnica nelle sue parti costitutive per una migliore comprensione e distinguere tra fatti e inferenze.

In generale, tutte le parti di questo syllabus sono esaminabili a livello K1. Quindi il candidato riconoscerà, ricorderà e richiamerà un termine o un concetto. Gli obiettivi di apprendimento ai livelli K2, K3 e K4 sono indicati all'inizio del relativo capitolo.

0.5 Tempi di formazione consigliati

È stato definito un tempo minimo di formazione per ciascun obiettivo di apprendimento di questo syllabus. Il tempo totale per ciascun capitolo è indicato nell'intestazione.

Gli enti di formazione tengano presente che altri syllabi ISTQB applicano un approccio a "tempo standard" che assegna tempi fissi in base al livello K. Il syllabus di Performance Testing non applica rigorosamente questo schema. Di conseguenza, gli enti di formazione ricevono un'indicazione più flessibile e realistica dei tempi minimi di formazione per ciascun obiettivo di apprendimento.

0.6 Requisiti d'ingresso

È necessario ottenere il certificato principale di livello Foundation prima di sostenere l'esame di certificazione di Performance Testing di livello Foundation.

0.7 Fonti di informazione

I termini utilizzati nel syllabus sono definiti nel Glossario dei termini di ISTQB utilizzato nel test del software [ISTQB_GLOSSARY].

La sezione 6 contiene un elenco di libri e articoli consigliati sul Performance Testing.

1. Concetti di base - 60 minuti.

Parole chiave

Capacity testing, efficienza, generazione del carico, performance testing, stress testing, test di carico, test di concorrenza, Spike testing, test di resistenza, test di scalabilità

Obiettivi di apprendimento per concetti di base

1.1 Principi e concetti
PTFL-1.1.1 (K2) Comprendere i principi del Performance Testing

1.2 Tipologie di Performance Testing
PTFL-1.2.1 (K2) Comprendere i diversi tipi di Performance Testing

1.3 Tipologie di test nel Performance Testing
PTFL-1.3.1 (K1) Richiamare le tipologie di Performance Testing

1.4 Il concetto di generazione di carico
PTFL-1.4.1 (K2) Comprendere il concetto di generazione del carico

1.5 Errori comuni nel Performance Testing e loro cause
PTFL-1.5.1 (K2) Fornire esempi di modalità di errore comuni del Performance Testing e relative cause

1.1 Principi del Performance Testing

L'efficienza delle prestazioni (o semplicemente "prestazioni") è una parte essenziale per offrire una "buona esperienza" agli utenti quando usano le loro applicazioni su molte piatteforme fisse e mobili. Il Performance Testing svolge un ruolo fondamentale nello stabilire livelli di qualità accettabili per l'utente finale ed è spesso strettamente integrato con altre discipline, come l'ingegneria dell'usabilità e delle prestazioni.

Inoltre, la valutazione di idoneità funzionale, fruibilità e altre caratteristiche di qualità in condizioni di carico, ad esempio durante l'esecuzione di un Performance Testing, può rivelare problemi specifici del carico, che incidono su tali caratteristiche.

I Performance Testing non si limitano al dominio basato sul web, che si concentra sull'utente finale. È inoltre rilevante per diversi domini applicativi, con una varietà di architetture di sistema, come il classico client-server, distribuito e integrato.
Tecnicamente, l'efficienza delle prestazioni è classificata nel Modello di qualità del prodotto ISO 25010 [ISO25000] come caratteristica di qualità non funzionale con le tre

sotto-caratteristiche descritte di seguito. La corretta messa a fuoco e la definizione delle priorità dipendono dai rischi valutati e dalle esigenze dei diversi stakeholder. L'analisi dei risultati dei test può identificare altri ambiti di rischio che devono essere affrontati.

Comportamento temporale: In generale la valutazione del comportamento temporale è l'obiettivo più comune del Performance Testing. Questo aspetto del Performance Testing esamina la capacità di un componente o di un sistema di rispondere agli input dell'utente o del sistema entro un determinato periodo di tempo e in specifiche condizioni. Le misurazioni del comportamento temporale possono variare dal tempo "end-to-end" impiegato dal sistema per rispondere all'input dell'utente, al numero di cicli CPU richiesti da un componente software per eseguire un determinato compito.

Utilizzo delle risorse: Se la disponibilità delle risorse di sistema viene identificata come un rischio, l'utilizzo di tali risorse (ad es. allocazione di RAM limitata) può essere studiato conducendo specifici test di Performance.

Capacità: Se vengono identificati come rischi i problemi di comportamento ai limiti di capacità richiesti per il sistema (ad es. numero di utenti o volumi di dati), è possibile condurre test di Performance per valutare l'idoneità dell'architettura del sistema.

Il Performance Testing spesso assume la forma di una sperimentazione, che consente la misurazione e l'analisi di specifici parametri di sistema. Questi possono essere condotti in modo iterativo a supporto dell'analisi, della progettazione e dell'implementazione del sistema per consentire di prendere decisioni sull'architettura e contribuire a modellare le aspettative degli stakeholder.

I seguenti principi di Performance Testing sono particolarmente rilevanti.

- I test devono essere allineati alle aspettative definite per diversi gruppi, in particolare utenti, progettisti di sistemi e personale operativo.
- I test devono essere riproducibili. I risultati statisticamente identici (entro una tolleranza definita) devono essere ottenuti ripetendo i test su un sistema che resta invariato.
- I test devono produrre risultati comprensibili e facilmente confrontabili con le aspettative degli stakeholder.
- I test possono essere condotti, laddove le risorse lo consentano, su sistemi completi o parziali o ambienti di test rappresentativi del sistema di produzione.
- I test devono essere accessibili ed eseguibili in termini pratici entro i tempi stabiliti dal progetto.

I libri di [Molyneaux09] e [Microsoft07] forniscono una solida base ai principi e agli aspetti pratici del Performance Testing.

Tutte e tre le suddette sotto-caratteristiche di qualità influiranno sulla capacità del sistema sotto test (SUT) in termini di scalabilità.

1.2 Tipologie di Performance Testing

È possibile definire diversi tipi di Performance Testing. Ognuno di questi può essere applicato a un determinato progetto, a seconda degli obiettivi del test.

Test di performance

"Test di performance" è un termine generico che comprende qualsiasi tipo di test incentrato sulle prestazioni (reattività) del sistema o del componente sottoposto a diversi volumi di carico.

Test di carico

Il test del carico si concentra sulla capacità di un sistema di gestire livelli crescenti di carichi realistici previsti derivanti da richieste di transazione generate da un numero controllato di utenti o processi concomitanti.

Stress test

Lo stress test si concentra sulla capacità di un sistema o un componente di gestire i picchi di carico che sono pari o superiori ai limiti previsti o specificati in termini di carico di lavoro. Lo stress test viene anche utilizzato per valutare la capacità di un sistema di gestire una disponibilità ridotta di risorse, come capacità di elaborazione accessibile, larghezza di banda disponibile e memoria.

Test di scalabilità

I test di scalabilità sono incentrati sulla capacità di un sistema di soddisfare i futuri requisiti di efficienza che potrebbero eccedere quelli attualmente richiesti. L'obiettivo di questi test è determinare la capacità del sistema di crescere (ad esempio, con più utenti, maggiori quantità di dati archiviati) senza violare i requisiti di performance attualmente specificati o senza anomalie. Una volta noti i limiti di scalabilità, è possibile determinare e monitorare i valori di soglia nella produzione, così da lanciare l'allarme su eventuali problemi prima che si verifichino. Inoltre, l'ambiente di produzione può essere adattato con quantità adeguate di hardware.

Spike testing

Lo spike testing si concentra sulla capacità di un sistema di rispondere correttamente a improvvisi picchi di carico e rientrare successivamente in una condizione stabile.

Test di resistenza

I test di resistenza si concentrano sulla stabilità del sistema in un arco di tempo specifico per il contesto operativo del sistema. Questo tipo di test verifica che non vi siano problemi di capacità delle risorse (ad esempio, perdite di memoria, connessioni al database, thread pool) che possano finire per degradare le prestazioni e/o causare guasti nei punti di rottura.

Test di concorrenza

Il test di concorrenza si concentra sull'effetto di situazioni in cui si verificano simultaneamente azioni specifiche (ad esempio, quando un numero elevato di utenti accede contemporaneamente). I problemi di concorrenza sono notoriamente difficili da trovare e riprodurre, in particolare quando il problema si verifica in un ambiente in cui i test hanno un controllo scarso o nullo, come la produzione.

Test di capacità

Il test di capacità determina quanti utenti e/o transazioni potranno essere supportati da un determinato sistema, senza che questo manchi gli obiettivi di performance dichiarati. Questi obiettivi possono anche essere dichiarati rispetto ai volumi di dati risultanti dalle transazioni.

1.3 Tipologie di test per il Performance Testing

Le principali tipologie utilizzate nel Performance Testing includono test statici e test dinamici.

1.3.1 Test statici

Le attività di test statici sono spesso più importanti per il Performance Testing che per i test di idoneità funzionale. Questo perché nell'architettura e nella progettazione del sistema vengono introdotti moltissimi difetti essenziali di prestazione, che possono essere dovuti a incomprensioni o mancanza di conoscenza da parte di progettisti e architetti. Questi difetti possono sorgere perché i requisiti non acquisiscono adeguatamente i tempi di risposta, la velocità effettiva o gli obiettivi di utilizzo delle risorse, il carico e l'utilizzo del sistema previsti o i vincoli.

Le attività di test statici per le prestazioni possono includere:

- Revisioni dei requisiti con particolare attenzione agli aspetti di performance e ai rischi
- Revisioni di schemi di database, diagrammi entità-relazione, metadati, procedure e ricerche salvate
- Revisioni dell’architettura di sistema e di rete
- Revisioni di segmenti critici del codice di sistema (ad es. algoritmi complessi)

1.3.2 Test dinamici

Man mano che il sistema viene costruito, il Performance Testing dinamico dovrebbe essere introdotto il prima possibile. Le opportunità per il Performance Testing dinamico includono:

- Durante Unit Test, compreso l'uso di informazioni di profilazione per determinare potenziali colli di bottiglia (bottleneck) e analisi dinamica per valutare l'utilizzo delle risorse
- Durante i test di integrazione dei componenti, tra gli use case e i flussi di lavoro principali, in particolare quando si integrano diverse funzionalità degli use case o con la struttura "backbone" di un flusso di lavoro
- Durante i test di sistema di comportamenti generali end-to-end in varie condizioni di carico
- Durante i test di integrazione del sistema, soprattutto per i flussi di dati e di lavoro attraverso le principali interfacce tra sistemi. Nel test di integrazione del sistema non è raro che l'"utente" sia un altro sistema o una macchina (ad es. input da sensori e altri sistemi)
- Durante i test di accettazione, per costruire la fiducia dell'utente, del cliente e dell'operatore nelle prestazioni adeguate del sistema e per ottimizzare il sistema in condizioni reali (ma generalmente senza trovare difetti di prestazione nel sistema)

In livelli di test più elevati, come test di sistema e test di integrazione di sistema, l'uso di ambienti, dati e carichi realistici è fondamentale per ottenere risultati accurati (si veda il Capitolo 4). In Agile e altri cicli di vita iterativi-incrementali, i team devono includere Performance Testing statici e dinamici nelle iterazioni iniziali, invece di attendere le iterazioni finali per affrontare i rischi sulle performance.

Se il sistema comprende hardware personalizzato o nuovo, è possibile utilizzare simulatori per eseguire i primi Performance Testing dinamici. Tuttavia, è buona norma iniziare i test sull'hardware effettivo non appena possibile, poiché i simulatori spesso non acquisiscono adeguatamente i vincoli delle risorse e i comportamenti relativi alle prestazioni.

1.4 Il concetto di generazione del carico

Per eseguire i vari tipi di Performance Testing descritti nella Sezione 1.2, i carichi rappresentativi del sistema devono essere definiti, generati e sottoposti al sistema in prova. I carichi sono paragonabili agli input di dati utilizzati per i casi di test funzionali, ma differiscono nelle seguenti modalità principali:

- Un carico di Performance Testing deve rappresentare molti input dell'utente, non solo uno
- Un carico di Performance Testing può necessitare di hardware e strumenti di generazione dedicati

- La generazione di un carico di Performance Testing dipende dall'assenza di difetti funzionali nel sistema in prova, che potrebbero influire sull'esecuzione del test

La generazione efficiente e affidabile di un dato carico è un fattore chiave per l'esecuzione di Performance Testing. Esistono diverse opzioni per la generazione del carico.

Creazione del carico tramite l'interfaccia utente
Questo approccio può risultare adeguato se è necessario rappresentare solo un numero limitato di utenti e se è disponibile il corretto numero di client software da cui immettere gli input richiesti. Questo approccio può essere utilizzato anche in combinazione con strumenti di esecuzione di test funzionali, ma può rapidamente diventare poco pratico all'aumentare del numero di utenti da simulare. La stabilità dell'interfaccia utente (UI) rappresenta un altro elemento critico. Le frequenti modifiche possono influire sulla ripetibilità del Performance Testing e possono influire in modo significativo sui costi di manutenzione. I test eseguiti con l'interfaccia utente possono costituire l'approccio più rappresentativo per i test end-to-end.

Creazione del carico tramite Crowd
Questo approccio dipende dalla disponibilità di un gran numero di tester che rappresenteranno utenti reali. Nei crowd test, i tester sono organizzati in modo tale da generare il carico desiderato. Questo può essere un metodo adatto per testare le applicazioni che sono raggiungibili da qualsiasi parte del mondo (come quelle basate sul web) e che possono coinvolgere gli utenti che generano un carico da dispositivi di svariate tipologie e configurazioni. Sebbene questo approccio possa consentire l'utilizzo da un numero molto elevato di utenti, il carico generato non sarà altrettanto riproducibile e preciso come in altre opzioni ed è più complesso da organizzare.

Generazione del carico tramite Application Programming Interface (API)
Questo approccio è simile all'utilizzo dell'interfaccia utente per l'immissione dei dati, ma utilizza l'API dell'applicazione anziché l'interfaccia utente per simulare l'interazione dell'utente con il sistema in prova. L'approccio è quindi meno sensibile alle modifiche (ad esempio, ritardi) nell'interfaccia utente e consente di elaborare le transazioni come se fossero inserite direttamente da un utente tramite la sua interfaccia. È possibile creare script dedicati che richiamano ripetutamente routine API specifiche e consentono di simulare un numero maggiore di utenti rispetto all'uso degli input dell'interfaccia utente.

Generazione del carico mediante Captured Communication Protocols
Questo approccio prevede l'acquisizione dell'interazione dell'utente con il sistema in prova a livello di protocollo di comunicazione e quindi la riproduzione di questi script per simulare un numero potenzialmente molto elevato di utenti in modo ripetibile e affidabile. Questo approccio basato su strumenti viene descritto nelle Sezioni 4.2.6 e 4.2.7.

1.5 Modalità di errori comuni di efficienza delle prestazioni e relative cause

Mentre ci sono certamente molte diverse modalità di errore delle prestazioni che possono essere rilevate durante i test dinamici, le modalità seguenti sono tra le più comuni (inclusi crash di sistema), insieme alle cause tipiche:

Risposta lenta a tutti i livelli di carico
In alcuni casi, la risposta è inaccettabile indipendentemente dal carico. Ciò può essere causato da problemi di prestazioni alla base, tra cui, ma non solo, cattiva progettazione o implementazione del database, latenza di rete e altri carichi in background. È possibile identificare questi problemi durante i test funzionali e di usabilità, non solo durante quelli di performance, quindi sta agli analisti dei test prestare attenzione e segnalarli.

Risposta lenta con livelli di carico da moderati a importanti
In alcuni casi, la risposta si degrada in modo inaccettabile con un carico da moderato a importante, anche quando tali carichi rientrano completamente negli intervalli normali, previsti e consentiti. I difetti alla base includono la saturazione di una o più risorse e la variazione dei carichi in background.

Risposta degradata nel tempo
In alcuni casi, la risposta si degrada progressivamente o gravemente nel tempo. Le cause alla base includono perdite di memoria, frammentazione del disco, aumento del carico di rete nel tempo, crescita del repository di file e una crescita imprevista del database.

Gestione degli errori inadeguata in caso di carico importante o eccessivo
In alcuni casi, il tempo di risposta è accettabile ma la gestione degli errori peggiora a livelli di carico elevati e superiori al limite. I difetti di base includono pool di risorse insufficienti, code e stack sottodimensionati e impostazioni di timeout troppo rapido.

Esempi specifici delle tipologie generali di errore sopra menzionate comprendono:

- Un'applicazione basata sul Web che fornisce informazioni sui servizi di un'azienda non risponde alle richieste degli utenti entro sette secondi (una regola generale del settore). Non è possibile ottenere l'efficienza di performance del sistema in condizioni di carico specifiche.
- Un sistema si arresta in modo anomalo o non è in grado di rispondere agli input dell'utente se sottoposto a un numero improvvisamente elevato di richieste di utenti (ad esempio, vendita di biglietti per un grande evento sportivo). La capacità del sistema di gestire questo numero di utenti è inadeguata.
- La risposta del sistema è notevolmente degradata quando gli utenti inviano richieste di grandi quantità di dati (ad esempio, un report di grande peso e

importanza viene messo a disposizione sul Web per il download). La capacità del sistema di gestire i volumi di dati generati è insufficiente.

- L'elaborazione batch non può essere completata prima che sia necessaria l'elaborazione online. Il tempo di esecuzione delle elaborazioni batch è insufficiente per il tempo consentito.
- Un sistema in tempo reale esaurisce la RAM quando i processi paralleli generano grandi richieste di memoria dinamica che non può essere liberata in tempo. La RAM non è correttamente dimensionata o le richieste di RAM non seguono le priorità adeguate.
- Un componente di sistema A in tempo reale che fornisce input al componente di sistema B in tempo reale non è in grado di calcolare gli aggiornamenti alla velocità richiesta. L'intero sistema non risponde in tempo e potrebbe non funzionare. I moduli di codice nel componente A devono essere valutati e modificati ("profilazione delle prestazioni") per garantire il raggiungimento delle velocità di aggiornamento richieste.

2. Fondamenti di misura delle prestazioni (Performance Measurement) - 55 min.

Parole chiave

misurazione, parametri di misura (metriche)

Obiettivi di apprendimento per i fondamenti di misurazione delle prestazioni

2.1 Parametri di misura tipici raccolti nel Performance Testing
PTFL-2.1.1 (K2) Comprendere i parametri di misura tipici raccolti nel Performance Testing

2.2 Aggregazione dei risultati del Performance Testing
PTFL-2.2.1 (K2) Spiegare perché i risultati del Performance Testing vengono aggregati

2.3 Origini principali delle metriche di performance
PTFL-2.3.1 (K2) Comprendere le origini fondamentali delle metriche di performance

2.4 Risultati tipici di un Performance Testing
PTFL-2.4.1 (K1) Richiamare i risultati tipici di un Performance Testing

2.1 Parametri di misura tipici raccolti nel Performance Testing

2.1.1 Perché sono necessari parametri di misura di performance

Misurazioni precise e i parametri derivati da tali misurazioni sono essenziali per definire gli obiettivi del Performance Testing e per valutarne i risultati. Il Performance Testing non deve essere eseguito senza prima comprendere quali siano le misurazioni e i parametri di misura necessari. Se questa raccomandazione viene ignorata, si presentano i seguenti rischi:

- Non è noto se i livelli di prestazione siano accettabili per raggiungere gli obiettivi operativi
- I requisiti di performance non sono definiti in termini misurabili
- Potrebbe non essere possibile identificare tendenze che preludano a livelli di prestazione inferiori
- I risultati effettivi di un Performance Testing non possono essere valutati confrontandoli con una serie basilare di misurazioni delle prestazioni che definiscono prestazioni accettabili e/o inaccettabili
- I risultati del Performance Testing vengono valutati in base all'opinione soggettiva di una o più persone

- I risultati forniti da uno strumento di Performance Testing non vengono compresi

2.1.2 Raccolta di misurazioni e di metriche di performance

Come con qualsiasi forma di misurazione, è possibile ottenere ed esprimere con precisione i parametri. Pertanto, uno qualsiasi dei parametri e delle misure descritti in questa sezione può e deve essere definito come significativo in un particolare contesto. È importante eseguire test iniziali e apprendere quali parametri devono essere ulteriormente perfezionati e quali devono essere aggiunti.

Ad esempio, la misura del tempo di risposta è probabile che rientri in tutte le serie di parametri di misura di performance. Tuttavia, per essere significativa e attuabile, la metrica del tempo di risposta dovrà essere ulteriormente definita in termini di ora del giorno, numero di utenti presenti contemporaneamente, quantità di dati in elaborazione e così via.

Le misurazioni raccolte in uno specifico Performance Testing varieranno in base al

- contesto aziendale (processi aziendali, comportamento del cliente e dell'utente e aspettative degli stakeholder),
- contesto operativo (la tecnologia e come viene utilizzata)
- obiettivi del test

Ad esempio, i parametri scelti per il Performance Testing di un sito Internet internazionale di e-commerce saranno diversi da quelli scelti per il Performance Testing di un sistema embedded utilizzato per controllare la funzionalità di dispositivi medici.

Un modo comune per classificare le misurazioni e le metriche di performance è quello di considerare l'ambiente tecnico, quello aziendale od operativo, in cui è necessario valutare le prestazioni.

Le categorie di misurazioni e metriche elencate in seguito sono quelle comunemente ottenute dal Performance Testing.

Ambiente tecnico
Le metriche di performance variano in base al tipo di ambiente tecnico, come mostrato nel seguente elenco:

- Su base Web
- Mobile
- Internet-of-Things (IoT)
- Dispositivi client su desktop
- Elaborazione lato server
- Mainframe
- Banche dati
- Reti

- La natura del software in esecuzione nell'ambiente (ad esempio, embedded)

Le metriche includono:

- Tempo di risposta (ad es. per transazione, per utenti contemporanei, tempi di caricamento della pagina)
- Utilizzo delle risorse (ad es. CPU, memoria, larghezza di banda di rete, latenza di rete, spazio disponibile su disco, velocità I/O, thread inattivi e occupati)
- Velocità effettiva della transazione chiave (ovvero il numero di transazioni che possono essere elaborate in un determinato periodo di tempo)
- Tempo di elaborazione batch (ad es. tempi di attesa, tempi di elaborazione, tempi di risposta della base di dati, tempi di completamento)
- Numero di errori che incidono sulle prestazioni
- Tempo di completamento (ad es. per creare, leggere, aggiornare ed eliminare i dati)
- Carico in background su risorse condivise (specialmente in ambienti virtualizzati)
- Parametri del software (ad es. complessità del codice)

Ambiente di business

Dal punto di vista di business o funzionale, le metriche di performance possono includere:

- Efficienza dei processi di business (ad esempio, la velocità di esecuzione di un processo di business generale, inclusi flussi di use case normali, alternativi ed eccezionali)
- Velocità di elaborazione di dati, transazioni e altre unità di lavoro eseguite (ad es. ordini elaborati all'ora, righe di dati aggiunte al minuto)
- Costi di violazione o conformità di uno SLA (Service Level Agreement) (ad es. violazioni del SLA per unità di tempo)
- Ambito di utilizzo (ad esempio, percentuale di utenti mondiali o nazionali che svolgono attività in un determinato momento)
- Concorrenza di utilizzo (ad esempio, il numero di utenti che eseguono contemporaneamente un'attività)
- Tempi di utilizzo (ad esempio, il numero di ordini elaborati durante i picchi di carico)

Ambiente operativo

L'aspetto operativo del Performance Testing si concentra su attività che in genere non sono considerate come rivolte agli utenti per loro natura. Queste includono:

- Processi operativi (ad esempio, il tempo necessario per l'avvio dell'ambiente, i backup, i tempi di spegnimento e di ripresa)
- Ripristino del sistema (ad esempio, il tempo necessario per ripristinare i dati da un backup)
- Avvisi e avvertenze (ad esempio, il tempo necessario al sistema per inviare un avviso o un'avvertenza)

2.1.3 Selezione dei parametri di misura di performance

Si voglia notare che la raccolta di un numero maggiore di metriche rispetto al necessario non è necessariamente una buona prassi. Ogni parametro scelto necessita di un mezzo per la raccolta e la rendicontazione coerenti. È importante definire un insieme realizzabile di metriche a supporto degli obiettivi del Performance Testing.

Ad esempio, l'approccio Goal-Question-Metric (GQM) è un modo utile per allineare i parametri agli obiettivi di performance. L'idea è di stabilire prima gli obiettivi, quindi porre quesiti per sapere quando gli obiettivi sono stati raggiunti. A ciascuna domanda vengono associati parametri di misura per garantire che la risposta alla domanda sia misurabile. (Si veda la Sezione 4.3 del Syllabus di livello Expert - Miglioramento del processo di test [ISTQB_ELTM_ITP_SYL] per una descrizione più completa dell'approccio GQM.) Si voglia notare che l'approccio GQM non si adatta sempre al processo di Performance Testing. Ad esempio, alcuni parametri rappresentano lo stato di un sistema e non sono direttamente collegati agli obiettivi.

È importante comprendere che, dopo la definizione e l'acquisizione delle misurazioni iniziali, potrebbero essere necessarie ulteriori misurazioni e parametri per comprendere i reali livelli di performance e determinare dove potrebbero essere necessarie azioni correttive.

2.2 Aggregazione dei risultati del Performance Testing

Lo scopo di aggregare i parametri di misura di performance è quello di comprenderli ed esprimerli in un modo che trasmetta accuratamente il quadro complessivo delle prestazioni del sistema. Quando i parametri di misura di performance sono visualizzati esclusivamente a livello dettagliato, può risultare difficile trarre le giuste conclusioni, specialmente per i business stakeholder.

Per molti stakeholder, la preoccupazione principale è che il tempo di risposta di un sistema, sito Web o altro oggetto di test rientri nei limiti accettabili.

Una volta raggiunta una più profonda comprensione dei parametri di performance, i parametri possono essere aggregati in modo che:

- Gli stakeholder di business e di progetto possono vedere il "quadro generale" delle prestazioni del sistema
- È possibile identificare le tendenze delle prestazioni
- Le metriche di performance possono essere riportate in maniera comprensibile

2.3 Origini principali delle metriche di performance

Il lavoro di raccolta delle metriche deve avere un'influenza minima sulle prestazioni del sistema (l'"effetto sonda"). Inoltre, il volume, la precisione e la velocità di raccolta delle

metriche di performance rendono necessario l'utilizzo di uno strumento. Sebbene l'uso combinato di strumenti non sia raro, questo può portare a una ridondanza nell'impiego di strumenti di test e altri problemi (si veda la Sezione 4.4).

Esistono tre fonti principali di metriche di performance:

Strumenti di Performance Testing
Tutti gli strumenti di Performance Testing forniscono misure e parametri a seguito di un test. Gli strumenti possono variare nel numero di parametri mostrati, nella maniera e nella possibilità per l'utente di personalizzare i parametri in base a una situazione particolare (si veda anche la Sezione 5.1).

Alcuni strumenti raccolgono e visualizzano i parametri di performance in formato testo, mentre strumenti più potenti raccolgono e visualizzano i parametri di performance graficamente, come un pannello di controllo. Molti strumenti offrono la possibilità di esportare le metriche per facilitare la valutazione e il reporting dei test.

Strumenti di monitoraggio delle prestazioni
Gli strumenti di monitoraggio delle prestazioni vengono spesso impiegati per integrare le capacità di reporting degli strumenti di Performance Testing (si veda anche la Sezione 5.1). Inoltre, gli strumenti di monitoraggio possono essere utilizzati per monitorare le prestazioni del sistema su base continuativa e per avvisare gli amministratori di sistema in caso di livelli di performance ridotti e livelli più elevati di errori e avvisi di sistema. Questi strumenti possono anche essere utilizzati per rilevare e inviare una notifica in caso di comportamenti sospetti (come attacchi “denial of service” e attacchi “denial of service” distribuiti).

Strumenti di analisi dei log
Esistono strumenti che scansionano i log del server e ne compilano le metriche. Alcuni di questi strumenti possono creare grafici per fornire una rappresentazione grafica dei dati.

Nei registri del server vengono normalmente registrati errori, avvisi e avvertenze. Questi includono:

- Elevato utilizzo delle risorse, come elevato utilizzo della CPU, elevati livelli di consumo della memoria su disco e larghezza di banda insufficiente
- Avvisi ed errori di memoria, come esaurimento della memoria
- Deadlock e problemi di multi-threading, in particolare durante l'esecuzione di operazioni su database
- Errori del database, come eccezioni e timeout SQL

2.4 Risultati tipici di un Performance Testing

Nei test funzionali, in particolare durante la verifica dei requisiti funzionali specifici o degli elementi funzionali delle user stories, normalmente è possibile definire con chiarezza i risultati attesi ed interpretare i risultati dei test per determinare se il test è stato superato o meno. Ad esempio, un rapporto mensile sulle vendite mostra un totale corretto o errato.

Mentre i test che verificano l'idoneità funzionale spesso beneficiano di oracoli di test ben definiti, il Performance Testing spesso non dispone di questa fonte di informazioni. Non solo gli stakeholder notoriamente non sono abili nell'elaborare i requisiti di performance, ma molti analisti di business e product owner non sono in grado di estrapolare tali requisiti. I tester ricevono spesso linee guida limitate per definire i risultati dei test previsti.

Quando si valutano i risultati del Performance Testing, è importante esaminarli attentamente. I risultati iniziali non elaborati possono essere fuorvianti, con errori di prestazioni nascosti da risultati complessivi apparentemente buoni. Ad esempio, l'utilizzo delle risorse potrebbe essere ben al di sotto del 75% per tutte le potenziali risorse che sono dei bottleneck fondamentali, ma il tempo di elaborazione o di risposta delle transazioni chiave o dei casi d'uso potrebbe risultare ordini di grandezza troppo lento.

I risultati specifici da valutare variano a seconda dei test eseguiti e spesso includono quelli discussi nella Sezione 2.1.

3. Performance Testing nel ciclo di vita del software - 195 min.

Parole chiave

metrica, rischio, ciclo di vita dello sviluppo software, log dei test

Obiettivi di apprendimento

3.1 Principali attività di Performance Testing
PTFL-3.1.1 (K2) Comprendere le principali attività di Performance Testing

3.2 Rischi di performance per architetture diverse
PTFL-3.2.1 (K2) Spiegare le categorie tipiche dei rischi sulle performance per diverse architetture

3.3 Rischi sulle performance lungo il ciclo di sviluppo del software
PTFL-3.3.1 (K4) Analizzare i rischi sulle performance per un determinato prodotto lungo il ciclo di sviluppo del software

3.4 Attività di Performance Testing
PTFL-3.4.1 (K4) Analizzare un determinato progetto per determinare le attività di Performance Testing appropriate per ogni fase del ciclo di sviluppo del software

3.1 Principali attività di Performance Testing

Il Performance Testing è di natura iterativa. Ogni test fornisce informazioni preziose sulle prestazioni dell'applicazione e del sistema. Le informazioni raccolte da un test vengono utilizzate per correggere od ottimizzare i parametri dell'applicazione e del sistema. La successiva iterazione del test mostrerà quindi i risultati delle modifiche, fino al raggiungimento degli obiettivi del test.

Le attività di Performance Testing si allineano al processo di test ISTQB [ISTQB_FL_SYL].

Pianificazione dei test
La pianificazione dei test è particolarmente importante per il Performance Testing a causa della necessità di assegnazione degli ambienti di test, dati di test, strumenti e risorse umane. Inoltre, questa è l'attività in cui viene stabilita la portata del Performance Testing.

Durante la pianificazione dei test, le attività di identificazione e analisi dei rischi vengono completate e le informazioni pertinenti vengono aggiornate in tutta la documentazione di pianificazione dei test (ad es. test plan, level test plan). Proprio

come la pianificazione dei test, che viene rivista e modificata in base alle esigenze, così anche i rischi, i livelli e lo stato di rischio vengono modificati per riflettere i cambiamenti nelle condizioni di rischio.

Monitoraggio e controllo dei test
Le misure di controllo sono definite per fornire piani d'azione in caso di problemi che potrebbero influire sull'efficienza delle performance, ad esempio

- l'aumento della capacità di generazione del carico se l'infrastruttura non genera i carichi desiderati, come previsto per particolari Performance Testing
- hardware modificato, nuovo o sostituito
- modifiche ai componenti di rete
- modifiche all'implementazione del software

Gli obiettivi del Performance Testing vengono valutati per verificare il raggiungimento dei criteri d'uscita.

Analisi dei test
Il Performance Testing efficace si basa su un'analisi dei requisiti di performance, degli obiettivi dei test, dei Service Level Agreement (SLA), dell'architettura IT, dei modelli di processo e di altri elementi che costituiscono la base dei test. Questa attività può essere supportata dalla modellazione e dall'analisi dei requisiti e/o del comportamento delle risorse di sistema mediante fogli di calcolo o strumenti di pianificazione della capacità.

Vengono identificate specifiche condizioni di test, come livelli di carico, condizioni temporali e transazioni da testare. Vengono quindi determinati i tipi richiesti di Performance Testing (ad es. carico, stress, scalabilità).

Progettazione dei test
Vengono progettati performance test case. Questi sono generalmente creati in forma modulare in modo da poter essere utilizzati come elementi costitutivi di performance test più ampi e complessi (si veda la sezione 4.2).

Implementazione dei test
Nella fase di implementazione, i performance test case vengono ordinati all'interno di procedure di Performance Testing. Queste procedure di Performance Testing dovrebbero riflettere i passaggi normalmente eseguiti dall'utente e altre attività funzionali che devono essere eseguite durante il Performance Testing.

Un'attività di implementazione del test stabilisce e/o ripristina l'ambiente di test prima dell'esecuzione di ogni test. Poiché i performance test sono in genere basati sui dati, è necessario un processo per stabilire i dati dei test che rappresentino i dati di produzione effettivi in termini di volume e tipo, in modo da poter simulare l'uso in produzione.

Esecuzione dei test
L'esecuzione dei test si verifica quando viene condotto il Performance Testing, spesso tramite l'impiego di strumenti. I risultati dei test vengono valutati per determinare se le prestazioni del sistema soddisfano i requisiti e gli altri obiettivi dichiarati. Eventuali difetti vengono segnalati.

Completamento dei test
Gli stakeholder (ad es. architetti, manager, Product Owner) ricevono i risultati del Performance Testing in un test summary report. I risultati vengono espressi attraverso parametri che sono spesso aggregati per rendere più immediato il significato dei risultati del test. I mezzi visivi di reporting, come i pannelli di controllo, vengono spesso utilizzati per esprimere i risultati del Performance Testing in modi più facili da comprendere rispetto ai parametri basati su testo.

Il Performance Testing è spesso considerato attività in corso in quanto viene eseguito più volte e a tutti i livelli di test (componenti, integrazione, sistema, integrazione di sistema e test di accettazione). Al termine di un periodo definito di Performance Testing, è possibile raggiungere un punto di chiusura del test in cui i test progettati, le risorse degli strumenti di test (casi e procedure di test), i dati di test e altri elementi vengono archiviati o passati ad altri tester per un uso successivo durante le attività di manutenzione del sistema.

3.2 Categorie di rischi sulle performance per architetture diverse

Come accennato in precedenza, le prestazioni dell'applicazione o del sistema variano notevolmente in base all'architettura, all'applicazione e all'ambiente host. Sebbene non sia possibile fornire un elenco completo dei rischi sulle performance per tutti i sistemi, l'elenco seguente include alcune tipologie classiche di rischi associati ad architetture particolari :

Sistemi Single Computer
Si tratta di sistemi o applicazioni eseguiti interamente su un computer non virtualizzato. Le prestazioni possono peggiorare a causa di

- un consumo eccessivo di risorse, tra cui perdite di memoria, attività in background, come software di sicurezza, sottosistemi di archiviazione lenti (ad es. dispositivi esterni a bassa velocità o frammentazione del disco) e cattiva gestione del sistema operativo.
- l'implementazione inefficiente di algoritmi che non usano le risorse disponibili (ad es. memoria principale) e di conseguenza vengono eseguiti più lentamente del necessario.

Sistemi Multi-tier
Si tratta di sistemi di sistemi eseguiti su server multipli, ognuno dei quali esegue una serie specifica di attività, come il database server, il server di applicazione e il server

di presentazione. Essendo il server, ovviamente, un computer, è soggetto ai rischi indicati in precedenza. Inoltre, le prestazioni possono peggiorare a causa della progettazione del database scadente o non scalabile, di bottleneck della rete e di larghezza di banda o capacità inadeguate su ogni server.

Sistemi distribuiti
Si tratta di sistemi di sistemi, simili a un'architettura multilivello, ma i vari server possono cambiare in modo dinamico, come un sistema di e-commerce che accede a diversi database di inventario a seconda della posizione geografica della persona che effettua l'ordine. Oltre ai rischi associati alle architetture multilivello, questa architettura può riscontrare problemi di prestazioni dovuti a flussi di lavoro o di dati critici verso, da o attraverso server remoti inaffidabili o imprevedibili, soprattutto quando tali server presentano problemi di connessione periodici o periodi intermittenti di carico intenso.

Sistemi virtualizzati
Si tratta di sistemi in cui l'hardware fisico ospita vari computer virtuali. Queste macchine virtuali possono ospitare sistemi e applicazioni a computer singolo, nonché server che fanno parte di un'architettura multilivello o distribuita. I rischi sulle performance che derivano specificamente dalla virtualizzazione includono un carico eccessivo sull'hardware in tutte le macchine virtuali o una configurazione impropria della macchina virtuale host con conseguenti risorse inadeguate.

Sistemi dinamici / basati su cloud
Si tratta di sistemi che offrono una scalabilità su richiesta, aumentando la capacità all'aumentare del livello di carico. Questi sistemi sono in genere sistemi multilivello distribuiti e virtualizzati, seppur con funzionalità di scalabilità automatica progettate specificamente per mitigare alcuni dei rischi sulle performance associati a tali architetture. Tuttavia, esistono rischi associati a errori nel configurare in modo appropriato queste funzionalità durante l'installazione iniziale o gli aggiornamenti successivi.

Sistemi client-server
Si tratta di sistemi in esecuzione su un client che comunicano tramite un'interfaccia utente con un singolo server, un server multilivello o un server distribuito. Poiché sul client è in esecuzione un codice, i rischi relativi al singolo computer si applicano a quel codice, mentre sono possibili anche le sopra citate problematiche sul lato server. Inoltre, esistono rischi sulle performance dovuti a problemi di velocità e affidabilità della connessione, congestione della rete nel punto di collegamento del client (ad es. Wi-Fi pubblico) e potenziali problemi dovuti a firewall, packet inspection e bilanciamento del carico del server.

Applicazioni mobili
Si tratta di applicazioni in esecuzione su uno smartphone, un tablet o un altro dispositivo mobile. Tali applicazioni sono soggette ai rischi descritti per le applicazioni client-server e basate su browser (app Web). Inoltre, possono sorgere problemi di

prestazioni a causa delle risorse e della connettività limitate e variabili disponibili sul dispositivo mobile (che possono essere influenzate da posizione, durata della batteria, stato di carica, memoria disponibile sul dispositivo e temperatura). Per quelle applicazioni che utilizzano sensori di dispositivi o radio, come accelerometri o Bluetooth, i flussi di dati lenti provenienti da tali fonti potrebbero creare problemi. Infine, le applicazioni mobili hanno spesso forti interazioni con altre app mobili locali e servizi Web remoti, ciascuno dei quali può potenzialmente diventare un bottleneck per l'efficienza delle prestazioni.

Sistemi Real-time Embedded
Si tratta di sistemi che funzionano all'interno o addirittura controllano oggetti di uso quotidiano, come automobili (ad esempio, sistemi di intrattenimento e sistemi di frenata intelligenti), ascensori, segnali stradali, sistemi di riscaldamento, ventilazione e aria condizionata (HVAC) e altro ancora. Questi sistemi presentano spesso molti dei rischi dei dispositivi mobili, compresi i problemi (crescenti) relativi alla connettività, poiché questi dispositivi sono connessi a Internet. Tuttavia, le prestazioni ridotte di un videogioco mobile di solito non rappresentano un pericolo per la sicurezza dell'utente, mentre tali rallentamenti in un sistema di frenata del veicolo potrebbero rivelarsi catastrofici.

Applicazioni mainframe
Si tratta di applicazioni - in molti casi in servizio da decenni - che supportano spesso funzioni aziendali fondamentali per una missione, in un data center, a volte tramite l'elaborazione batch. La maggior parte è abbastanza prevedibile e veloce se utilizzata come previsto in origine, ma molte sono ora accessibili tramite API, servizi Web o tramite il loro database, il che può comportare carichi imprevisti che influiscono sulla velocità di elaborazione delle applicazioni stabilite.

Si noti che qualsiasi particolare applicazione o sistema può incorporare due o più delle architetture sopra elencate, il che significa che tutti i relativi rischi saranno presenti per l'applicazione o sistema. Infatti, in considerazione dell'Internet of Things e l'esplosione delle applicazioni mobili, due ambiti in cui livelli estremi di interazione e connessione sono la norma, è possibile che tutte le architetture siano presenti in qualche forma in un'applicazione, e quindi tutti i rischi siano possibili.

L'architettura è chiaramente un'importante decisione tecnica con conseguenze importanti sui rischi sulle performance, ma anche altre decisioni tecniche influenzano e creano rischi. Ad esempio, le perdite di memoria sono più comuni con i linguaggi che consentono la gestione diretta della memoria dell'heap, come C e C++, e i problemi di performance sono diversi per i database relazionali rispetto a quelli non relazionali. Tali decisioni si estendono fino alla progettazione di singoli funzioni o metodi (ad esempio, la scelta di un algoritmo ricorsivo rispetto a un algoritmo iterativo). Come tester, la capacità di conoscere o addirittura influenzare tali decisioni varierà, a seconda dei ruoli e delle responsabilità dei tester all'interno del ciclo di sviluppo del software e dell'organizzazione.

3.3 Rischi sulle performance in tutto il ciclo di sviluppo del software

Il processo di analisi dei rischi per la qualità di un prodotto software in generale è oggetto di vari syllabi ISTQB (ad esempio, [ISTQB_FL_SYL] e [ISTQB_ALTM_SYL]). È possibile anche trovare discussioni su rischi specifici e considerazioni associate a particolari caratteristiche di qualità (ad esempio, [ISTQB_UT_SYL]) e da una prospettiva di business o tecnica (ad esempio, vedi [ISTQB_ALTA_SYL] e [ISTQB_ALTTA_SYL], rispettivamente). In questa sezione, l'attenzione è rivolta ai rischi legati alla performance per la qualità del prodotto, compresi i modi in cui cambiano il processo, i partecipanti e le considerazioni.

Per i rischi sulle performance per la qualità del prodotto, il processo è:

1. Identificare i rischi per la qualità del prodotto, concentrandosi su caratteristiche come il comportamento nel tempo, l'utilizzo delle risorse e la capacità.
2. Valutare i rischi identificati, verificando che vengano trattate le categorie di architettura pertinenti (si veda la Sezione 3.2). Valutare il livello di rischio complessivo per ciascun rischio identificato in termini di probabilità ed effetto, utilizzando criteri definiti con chiarezza.
3. Intraprendere azioni di mitigazione del rischio adeguate a ciascun elemento di rischio, in base alla natura dell'elemento e al livello di rischio.
4. Gestire i rischi su base continuativa per garantire che vengano adeguatamente mitigati prima della pubblicazione.

Come per l'analisi del rischio sulla qualità in generale, i partecipanti a questo processo dovrebbero includere sia soggetti aziendali, sia tecnici. Per l'analisi del rischio sulle performance, gli stakeholder di business devono disporre di una particolare consapevolezza di come i problemi di prestazioni in produzione influenzeranno effettivamente i clienti, gli utenti, l'azienda e altri soggetti a valle. Gli stakeholder di business devono comprendere che l'uso previsto, le criticità aziendali, sociali o di sicurezza, i potenziali danni finanziari e/o reputazionali, la responsabilità legale civile o penale e fattori simili comportano conseguenze sul rischio dal punto di vista aziendale, creando rischi e influenzando gli impatti dei failure.

Inoltre, gli stakeholder tecnici devono includere quelli con una profonda comprensione delle implicazioni in termini di prestazioni dei requisiti, dell'architettura, del design e delle decisioni sull'implementazione. Gli stakeholder tecnici devono comprendere che le decisioni relative all'architettura, alla progettazione e all'implementazione hanno conseguenze sui rischi di performance dal punto di vista tecnico, creando rischi e influenzando la probabilità di difetti.

Il processo specifico di analisi del rischio scelto dovrebbe avere un livello adeguato di formalità e rigore. Per i rischi relativi alle prestazioni, è particolarmente importante che

il processo di analisi dei rischi venga avviato nella fase iniziale e sia ripetuto regolarmente. In altre parole, il tester dovrebbe evitare di fare affidamento interamente sul Performance Testing condotto verso la fine del livello di testing di sistema e del livello di testing di integrazione di sistema. Molti progetti, in particolare i progetti più grandi e complessi di 'sistemi di sistemi', hanno incontrato sorprese negative a causa della scoperta tardiva di errori risultanti da decisioni in termini di requisiti, progettazione, architettura e implementazione prese all'inizio del progetto. Ci si dovrebbe quindi concentrare su un approccio iterativo all'identificazione, la valutazione, la mitigazione e la gestione del rischio sulle performance durante tutto il ciclo di sviluppo del software.

Ad esempio, se grandi volumi di dati verranno gestiti tramite un database relazionale, le prestazioni lente di collegamenti multipli causati della cattiva progettazione del database potrebbero rivelarsi solo durante i test dinamici con set di dati di test su larga scala, come quelli utilizzati durante test di sistema. Tuttavia, un'attenta revisione tecnica che coinvolga ingegneri di database esperti può prevedere i problemi prima dell'implementazione del database. In un approccio iterativo, dopo tale revisione, i rischi vengono identificati e valutati nuovamente.

Inoltre, la mitigazione e la gestione del rischio devono estendersi e comportare effetti sull'intero processo di sviluppo del software e non limitarsi ai test dinamici. Ad esempio, quando le decisioni critiche relative alle prestazioni, come il numero previsto di transazioni o utenti in contemporanea, non possono essere definite all'inizio del progetto, è importante che le decisioni relative a progettazione e architettura consentano una scalabilità molto variabile (ad esempio, risorse di elaborazione basate su cloud e su richiesta). Ciò consente di prendere decisioni tempestive per la mitigazione del rischio.

Una buona progettazione delle prestazioni può aiutare i team di progetto a evitare la scoperta tardiva di difetti critici sulle performance durante i livelli di test più elevati, come i test di integrazione del sistema o i test di accettazione dell'utente. I difetti di performance rilevati in una fase avanzata del progetto possono essere estremamente costosi e possono persino portare all'eliminazione di interi progetti.

Come con qualsiasi tipo di rischio qualitativo, i rischi relativi alle prestazioni non possono mai essere evitati del tutto, ovvero è sempre presente un rischio di errore della produzione correlato alle prestazioni. Pertanto, il processo di gestione del rischio deve includere la fornitura di una valutazione realistica e specifica del livello di rischio residuo per gli stakeholder di business e tecnici coinvolti nel processo. Ad esempio, dire semplicemente "Sì, è ancora possibile per i clienti subiscano lunghi ritardi durante il check out" non è utile, in quanto non dà idea della portata della mitigazione del rischio o del livello di rischio rimanente. Invece, fornire una visione chiara della percentuale di clienti che potrebbero subire ritardi pari o superiori a determinate soglie aiuterà le persone a comprendere lo stato.

3.4 Attività di Performance Testing

Le attività di Performance Testing saranno organizzate ed eseguite in modo diverso, a seconda del tipo di ciclo di sviluppo del software in uso.

Modelli di sviluppo sequenziale
La pratica ideale del Performance Testing nei modelli di sviluppo sequenziale consiste nell'includere i criteri di performance come parte dei criteri di accettazione, che sono definiti all'avvio di un progetto. In virtù di una più estesa visione ciclica dei test, le attività di Performance Testing dovrebbero essere condotte durante l'intero ciclo di sviluppo del software. Man mano che il progetto avanza, ogni successiva attività di Performance Testing dovrebbe basarsi su elementi definiti nelle attività precedenti, come mostrato di seguito.

- Concetto: verificare che gli obiettivi di prestazione del sistema siano definiti come criteri di accettazione per il progetto.
- Requisiti: verificare che i requisiti di performance siano definiti e rappresentino correttamente le esigenze degli stakeholder.
- Analisi e progettazione: verificare che la progettazione del sistema rifletta i requisiti di performance.
- Codifica/Implementazione: verificare che il codice sia efficiente e rifletta i requisiti e la progettazione in termini di prestazioni.
- Test dei componenti: eseguire Performance Testing a livello dei componenti.
- Test di integrazione dei componenti: eseguire Performance Testing a livello di integrazione dei componenti.
- Test di sistema: eseguire Performance Testing a livello di sistema, che includano hardware, software, procedure e dati rappresentativi dell'ambiente di produzione. È possibile simulare le interfacce di sistema purché forniscano una rappresentazione reale delle prestazioni.
- Test di integrazione del sistema: eseguire Performance Testing con l'intero sistema che è rappresentativo dell'ambiente di produzione.
- Test di accettazione: verificare che le prestazioni del sistema soddisfino le esigenze dell'utente e i criteri di accettazione dichiarati in origine.

Modelli di sviluppo iterativo e incrementale
In questi modelli di sviluppo, come Agile, il Performance Testing è visto anche come attività iterativa e incrementale (si veda [ISTQB_FL_AT]). Il Performance Testing può essere eseguito come parte della prima iterazione o come iterazione interamente dedicata al Performance Testing. Tuttavia, con questi modelli di ciclo di vita, l'esecuzione del Performance Testing può essere eseguita da un team separato incaricato del Performance Testing.

L'integrazione continua (CI) viene comunemente eseguita nei cicli iterativi e incrementali di sviluppo del software, il che facilita l'esecuzione di test altamente automatizzata. L'obiettivo più comune dei test in CI è quello di eseguire test di regressione e verificare la stabilità di ogni build. Il Performance Testing può far parte

dei test automatici eseguiti in CI, se i test sono progettati in modo tale da essere eseguiti a livello di build. Tuttavia, a differenza dei test funzionali automatizzati, sono presenti ulteriori punti di attenzione:

- Impostazione dell'ambiente di Performance Testing: spesso è necessario un ambiente di test disponibile su richiesta, ad esempio un ambiente di Performance Testing basato su cloud.
- Determinazione del Performance Testing da automatizzare in CI: a causa del breve lasso di tempo disponibile per i test CI, il Performance Testing CI possono essere un sottoinsieme di Performance Testing più estesi, condotti da un team di specialisti più volte durante un'iterazione.
- Creazione del Performance Testing per CI: l'obiettivo principale del Performance Testing come parte del CI è garantire che una modifica non influisca negativamente sulle prestazioni. A seconda delle modifiche apportate per un certo build, potrebbero essere necessari nuovi Performance Testing.
- Esecuzione di Performance Testing su parti di un'applicazione o di un sistema: ciò spesso richiede che gli strumenti e gli ambienti di test siano in grado di eseguire rapidamente Performance Testing, inclusa la possibilità di selezionare sottoinsiemi di test applicabili.

I test di performance nei cicli di sviluppo iterativo e incrementale del software possono anche avere proprie attività riservate nel ciclo di vita:

- Release Planning: in questa attività, il Performance Testing viene considerato dal punto di vista di tutte le iterazioni che compongono una release, dalla prima iterazione a quella finale. I rischi di performance vengono identificati e valutati e si prevedono misure di mitigazione. Ciò include spesso la pianificazione di eventuale Performance Testing finale, prima del rilascio dell'applicazione.
- Iteration Planning: nell'ambito di ciascuna iterazione, è possibile eseguire Performance Testing all'interno dell'iterazione e al completamento di ciascuna. I rischi relativi alle prestazioni sono valutati in modo più dettagliato per ogni User Story.
- Creazione di User Story: le User Story spesso costituiscono la base dei requisiti di performance nelle metodologie Agile, con criteri di performance specifici descritti nei criteri di accettazione associati. Queste sono definite User Story "non funzionali".
- Progettazione di Performance Testing: i requisiti e i criteri relativi alle prestazioni vengono descritti in User Story specifiche e utilizzati per la progettazione dei test (si veda sezione 4.2)
- Codifica/Implementazione: durante la codifica, il Performance Testing può essere eseguito a livello di componente. Un esempio di ciò potrebbe essere la messa a punto di algoritmi per ottimizzare l'efficienza delle prestazioni.
- Test/valutazione: mentre i test vengono generalmente eseguiti in prossimità delle attività di sviluppo, il Performance Testing può essere eseguito come attività separata, a seconda della portata e degli obiettivi del Performance Testing durante l'iterazione. Ad esempio, se l'obiettivo del Performance Testing

è testare le prestazioni dell'iterazione verso un set di User Story completate, sarà necessaria una maggiore portata del Performance Testing rispetto a quella per il Performance Testing di una singola User Story. Questo può essere programmato in un'iterazione dedicata per il Performance Testing.

- Consegna: poiché la consegna introdurrà l'applicazione nell'ambiente di produzione, sarà necessario monitorare le prestazioni per determinare se l'applicazione raggiunge i livelli di prestazione desiderati nell'uso effettivo.

COTS (Commercial Off-the-Shelf) e altri modelli di fornitore/acquirente

Molte organizzazioni non sviluppano applicazioni e sistemi, ma sono invece nella posizione di acquisire software da fonti diverse: da fornitori o da progetti open source. In tali modelli fornitore/acquirente, le prestazioni sono un fattore importante che richiede test sia dal punto di vista del fornitore (fornitore/sviluppatore) sia dell'acquirente (cliente).

Indipendentemente dalla fonte dell'applicazione, il cliente è spesso tenuto a verificare che le prestazioni soddisfino i requisiti. Nel caso di software personalizzato sviluppato dal fornitore, i requisiti di performance e i criteri di accettazione associati dovrebbero essere indicati come parte del contratto tra il fornitore e il cliente. Nel caso di applicazioni COTS, il cliente ha la sola responsabilità di testare le prestazioni del prodotto in un ambiente di test realistico prima della distribuzione.

4. Attività di Performance Testing - 475 min.

Parole chiave

concorrenza, profilo di carico, generazione del carico, profilo operativo, decremento, incremento, sistema di sistemi, velocità di elaborazione del sistema, test plan, think time, utente virtuale

Obiettivi formativi

4.1 Pianificazione

PTFL-4.1.1 (K4) Derivare gli obiettivi del Performance Testing dalle informazioni pertinenti
PTFL-4.1.2 (K4) Delineare un piano di Performance Testing che consideri gli obiettivi di performance per un determinato progetto
PTFL-4.1.3 (K4) Creare una presentazione che consenta a vari soggetti di comprendere le motivazioni alla base del Performance Testing pianificato

4.2 Analisi, progettazione e implementazione

PTFL-4.2.1 (K2) Fornire esempi di protocolli tipici riscontrati nel Performance Testing
PTFL-4.2.2 (K2) Comprendere il concetto di transazioni nel Performance Testing
PTFL-4.2.3 (K4) Analizzare i profili operativi per l'utilizzo del sistema
PTFL-4.2.4 (K4) Creare profili di carico derivati da profili operativi per determinati obiettivi di prestazione
PTFL-4.2.5 (K4) Analizzare il tempo di elaborazione e la concorrenza durante lo sviluppo di Performance Testing
PTFL-4.2.6 (K2) Comprendere la struttura di base di uno script di Performance Testing
PTFL-4.2.7 (K3) Implementare script di Performance Testing coerenti con il piano e i profili di carico
PTFL-4.2.8 (K2) Comprendere le attività coinvolte nella preparazione per l'esecuzione del Performance Testing

4.3 Esecuzione

PTFL-4.3.1 (K2) Comprendere le principali attività nell'esecuzione degli script di Performance Testing

4.4 Analisi dei risultati e reporting

PRFL-4.4.1 (K4) Analizzare e riferire i risultati e le implicazioni del Performance Testing

4.1 Pianificazione

4.1.1 Derivazione degli obiettivi del Performance Testing

Gli stakeholder (ovvero le parti interessate al prodotto) possono includere utenti e persone con un background di business o tecnico. Possono avere obiettivi diversi in merito al Performance Testing. Gli stakeholder definiscono gli obiettivi, la terminologia da utilizzare e i criteri per determinare se l'obiettivo sia stato raggiunto

Gli obiettivi del Performance Testing si riferiscono a questi diversi tipi di soggetti. È buona norma distinguere tra obiettivi basati sull'utente e obiettivi tecnici. Gli obiettivi basati sull'utente si concentrano principalmente sulla soddisfazione dell'utente finale e sugli obiettivi di business. In generale, gli utenti sono meno interessati ai tipi di funzionalità o a come un prodotto venga consegnato. Vogliono solo essere messi in condizione di fare ciò di cui necessitano.

Gli obiettivi tecnici, d'altro canto, si concentrano sugli aspetti operativi e forniscono risposte a domande riguardanti la capacità di un sistema in termini di scalabilità o in quali condizioni il degrado delle prestazioni può apparire evidente.

Gli obiettivi chiave del Performance Testing includono l'identificazione di potenziali rischi, la ricerca di opportunità di miglioramento e il rilevamento delle modifiche necessarie.

Quando si raccolgono informazioni provenienti dai diversi stakeholder, è necessario rispondere alle seguenti domande:

- Quali transazioni verranno eseguite nel Performance Testing e quale tempo di risposta medio è previsto?
- Quali parametri di sistema devono essere acquisiti (ad es. utilizzo della memoria, velocità di elaborazione di rete) e quali valori sono previsti?
- Quali miglioramenti delle prestazioni sono previsti da questi test rispetto ai precedenti cicli di test?

4.1.2 Il piano di Performance Testing

Il piano di Performance Testing (PTP) è un documento creato prima di eseguire qualsiasi test di performance. Il PTP deve essere indicato dal test plan (si veda [ISTQB_FL_SYL]), che include anche le informazioni di pianificazione pertinenti. Continua ad essere aggiornato una volta iniziato il Performance Testing.

Un PTP deve includere le seguenti informazioni:

Obiettivo
L'obiettivo del PTP descrive le finalità, le strategie e i metodi per il Performance Testing. Consente una risposta quantificabile alla domanda centrale sull'adeguatezza e la prontezza del sistema a funzionare sotto carico.

Obiettivi del test
Gli obiettivi di test generali per l'efficienza delle prestazioni che devono essere raggiunti dal sistema in prova (SUT) sono elencati per ciascun tipo di soggetto (si veda Sezione 4.1.1)

Panoramica del sistema
Una breve descrizione del SUT fornirà il contesto per la misurazione dei parametri del Performance Testing. La panoramica deve includere una descrizione di alto livello della funzionalità sottoposta a test sotto carico.

Tipi di Performance Testing da condurre
Vengono elencati i tipi di Performance Testing da condurre (si veda la Sezione 1.2), insieme a una descrizione della finalità di ciascun tipo.

Criteri di accettazione
Il Performance Testing ha lo scopo di determinare la reattività, la velocità di elaborazione, l'affidabilità e/o la scalabilità del sistema con un determinato carico di lavoro. In generale, gli utenti sono attenti ai tempi di risposta, la velocità di elaborazione è una preoccupazione aziendale e l'utilizzo delle risorse è una questione di sistema. I criteri di accettazione devono essere definiti per tutte le misure pertinenti e correlati a quanto segue, a seconda dei casi:

- Obiettivi generali del Performance Testing
- Service Level Agreements (SLA)
- Valori di base: si tratta di una serie di parametri utilizzati per confrontare le misurazioni delle prestazioni attuali e precedentemente raggiunte. Ciò consente di dimostrare particolari miglioramenti delle prestazioni e/o di confermare il raggiungimento dei criteri di accettazione del test. Se possibile, potrebbe essere necessario creare prima la linea di riferimento utilizzando i dati ripuliti provenienti da un database.

Dati di test
I dati di test includono una vasta gamma di dati che devono essere specificati per un Performance Testing. Questi dati possono includere quanto segue:

- Dati dell'account utente (ad es. account utente disponibili per l'accesso simultaneo)
- Dati di input dell'utente (ad esempio, i dati che un utente inserisce nell'applicazione per eseguire un processo aziendale)
- Database (ad esempio, il database precompilato in cui vengono inseriti dati da utilizzare nei test)

Il processo di creazione dei dati di test dovrebbe riguardare i seguenti aspetti:

- estrazione dei dati dai dati di produzione
- importazione di dati nel sistema in prova

- creazione di nuovi dati
- creazione di backup che possano essere utilizzati per ripristinare i dati quando vengono eseguiti nuovi cicli di test
- mascheramento o anonimizzazione dei dati. Questa pratica viene utilizzata sui dati di produzione che contengono informazioni di identificazione personale ed è obbligatoria ai sensi dei regolamenti generali sulla protezione dei dati (GDPR). Tuttavia, nel Performance Testing, il mascheramento dei dati aggiunge rischi al Performance Testing, in quanto potrebbe non avere le stesse caratteristiche dei dati osservate nell'uso reale.

Configurazione di sistema
La sezione di configurazione del sistema del PTP include le seguenti informazioni tecniche:

- Una descrizione dell'architettura di sistema specifica, inclusi i server (ad es. Web, database, bilanciamento del carico)
- Definizione di livelli multipli
- Dettagli specifici dell'hardware di elaborazione (ad es. core della CPU, RAM, dischi a stato solido (SSD), dischi rigidi (HDD)), versioni comprese
- Dettagli specifici del software (ad es. applicazioni, sistemi operativi, database, servizi utilizzati per supportare l'azienda), versioni comprese
- Sistemi esterni che funzionano con il SUT e relative configurazione e versione (ad es. sistema di e-commerce con integrazione in NetSuite)
- Identificatore versione/build SUT

Ambiente di test
L'ambiente di test è spesso un ambiente separato che imita la produzione, ma su scala inferiore. Questa sezione del PTP illustra come i risultati del Performance Testing saranno estrapolati e applicati all'ambiente di produzione più ampio. Con alcuni sistemi, l'ambiente di produzione diventa l'unica opzione praticabile per i test, ma in questo caso è necessario discutere i rischi specifici di questo tipo di test.

Gli strumenti di test a volte risiedono al di fuori dell'ambiente di test e può essere necessario disporre di diritti di accesso speciali per interagire con i componenti del sistema. Questo vale per l'ambiente di test e la configurazione.

Il Performance Testing può anche essere condotto con una parte del sistema in grado di funzionare senza altri componenti. Spesso questo risulta più economico del test con l'intero sistema e può essere condotto non appena il componente viene sviluppato.

Strumenti di test
Questa sezione include una descrizione di quali strumenti di test (e versioni) verranno utilizzati nello scripting, nell'esecuzione e nel monitoraggio del Performance Testing (si veda il Capitolo 5). Questo elenco normalmente include:

- Strumenti usati per simulare le transazioni dell'utente

- Strumenti per fornire carico da più punti all'interno dell'architettura di sistema (punti di presenza)
- Strumenti per monitorare le prestazioni del sistema, compresi quelli descritti nella configurazione del sistema

Profili
I profili operativi forniscono un flusso graduale ripetibile attraverso l'applicazione, per un particolare utilizzo del sistema. L'aggregazione di questi profili operativi si traduce in un profilo di carico (comunemente chiamato scenario). Si veda la Sezione 4.2.3 per ulteriori informazioni sui profili.

Metriche più rilevanti
Durante l'esecuzione del Performance Testing è possibile raccogliere un gran numero di misure e parametri (si veda il Capitolo 2). Tuttavia, le troppe misurazioni possono rendere difficile l'analisi e influire negativamente sulle prestazioni effettive dell'applicazione. Per questi motivi, è importante identificare le misure e i parametri più rilevanti per raggiungere gli obiettivi del Performance Testing.

La tabella seguente, spiegata più in dettaglio nella Sezione 4.4, mostra un insieme tipico di parametri per il test e il monitoraggio delle prestazioni. Gli obiettivi di test per le performance dovrebbero essere definiti per questi parametri, ove richiesto, per il progetto:

Parametri di performance	
Tipologia	**Parametro**
Stato dell'utente virtuale	# Superato # Non superato
Tempo di risposta della transazione	Minimo Massimo Media 90% percentile
Transazioni al secondo	# Superato / secondo # Non superato / secondo # Totale / secondo
Hit (ad es. su database o web server)	Hit / secondo ▪ Minimo ▪ Massimo ▪ Media ▪ Totale

Parametri di performance	
Tipologia	**Parametro**
Velocità di elaborazione	Bit / secondo ▪ Minimo ▪ Massimo ▪ Media ▪ Totale
Risposte HTTP al secondo	Risposte / secondo ▪ Minimo ▪ Massimo ▪ Media ▪ Totale Risposta tramite codici di risposta HTTP

Monitoraggio delle prestazioni	
Tipologia	**Parametro**
utilizzo della CPU	% della CPU disponibile utilizzata
Utilizzo della memoria	% della memoria disponibile utilizzata

Rischi
I rischi possono riguardare ambiti non misurati durante il Performance Testing, nonché limitazioni al Performance Testing (ad es. interfacce esterne che non possono essere simulate, carico insufficiente, incapacità di monitorare i server). Le limitazioni dell'ambiente di test possono anche dar luogo a rischi (ad es. dati insufficienti, ambiente ridotto). Si vedano le sezioni 3.2 e 3.3 per altre tipologie di rischio.

4.1.3 Comunicazione sul Performance Testing

Il tester deve essere in grado di comunicare a tutti gli stakeholder la logica alla base dell'approccio di Performance Testing e le attività da intraprendere (come illustrato nel Piano di Performance Testing). Gli argomenti da trattare in questa comunicazione possono variare considerevolmente in funzione degli stakeholder, a seconda che abbiano un interesse "business/rivolto agli utenti" o piuttosto "tecnologico/operativo".

Stakeholder con focus sul business
È necessario considerare i seguenti fattori nella comunicazione con questi stakeholder:

- Gli stakeholder con focus sul business sono meno interessati alle distinzioni tra caratteristiche di qualità funzionali e non funzionali.
- Le questioni tecniche relative a strumenti, script e generazione di carico sono generalmente di interesse secondario.
- Il legame tra i rischi del prodotto e gli obiettivi del Performance Testing deve essere indicato con chiarezza.
- Gli stakeholder devono essere informati dell'equilibrio tra il costo del Performance Testing pianificato e la rappresentatività dei risultati dei test, rispetto alle condizioni di produzione.
- È necessario indicare la ripetibilità del Performance Testing pianificato. Il test sarà difficile da ripetere o può essere ripetuto facilmente?
- È necessario comunicare i rischi del progetto. Questi includono vincoli e dipendenze riguardanti l'impostazione dei test, i requisiti dell'infrastruttura (ad es. hardware, strumenti, dati, larghezza di banda, ambiente di test, risorse) e dipendenze dal personale chiave.
- È necessario comunicare le attività di alto livello (si vedano le Sezioni 4.2 e 4.3), insieme a un ampio piano contenente costi, calendario e tappe fondamentali.

Stakeholder con un focus tecnologico

È necessario considerare i seguenti fattori per la comunicazione con Stakeholder con un focus tecnologico:

- L'approccio pianificato per generare i profili di carico richiesti deve essere spiegato e chiarito il coinvolgimento previsto per gli stakeholder tecnici.
- I passaggi dettagliati nell'impostazione e nell'esecuzione del Performance Testing devono essere spiegati per mostrare la relazione dei test con i rischi architetturali.
- È necessario comunicare i passaggi utili per rendere ripetibile il Performance Testing. Questi possono includere aspetti organizzativi (ad es. partecipazione del personale chiave), nonché questioni tecniche.
- Laddove gli ambienti di test debbano essere condivisi, è necessario comunicare la pianificazione del Performance Testing per garantire che i risultati del test non vengano influenzati negativamente.
- È necessario comunicare e accettare le mitigazioni del potenziale effetto sugli utenti reali, se il Performance Testing deve essere eseguito nell'ambiente di produzione.
- Gli stakeholder tecnici devono essere chiari rispetto ai loro compiti e a quando questi sono programmati.

4.2 Analisi, progettazione e implementazione

4.2.1 Protocolli di comunicazione tipici

I protocolli di comunicazione definiscono una serie di regole di comunicazione tra computer e sistemi. La progettazione corretta dei test in modo da indirizzare parti specifiche del sistema richiede che i protocolli vengano compresi.

I protocolli di comunicazione sono spesso descritti dai livelli del modello Open Systems Interconnection (OSI) (si veda ISO/IEC 7498-1), sebbene alcuni protocolli possano non rientrare in questo modello. I protocolli dal Layer 5 (livello di sessione) al Layer 7 (livello di applicazione) sono i più comunemente utilizzati per il Performance Testing. I protocolli comuni includono:

- Database: ODBC, JDBC, altri protocolli specifici del fornitore
- Web: HTTP, HTTPS, HTML
- Servizio Web - SOAP, REST

In generale, il livello del layer OSI che risulta più centrale nel Performance Testing si riferisce al livello dell'architettura da testare. Quando si testano, ad esempio, alcune architetture incorporate di livello basso, i layer con numerazione inferiore del modello OSI saranno quelli su cui ci si concentrerà maggiormente.

I protocolli aggiuntivi utilizzati nel Performance Testing includono:

- Rete: DNS, FTP, IMAP, LDAP, POP3, SMTP, Windows Socket, CORBA
- Mobile: TruClient, SMP, MMS
- Accesso remoto: Citrix ICA, RTE
- SOA: MQSeries, JSON, WSCL

È importante comprendere l'architettura generale del sistema perché il Performance Testing possa essere eseguito su un singolo componente del sistema (ad esempio, server Web, server di database) o su un intero sistema tramite test end-to-end. Le applicazioni tradizionali a 2 livelli costruite con un modello client-server specificano il "client" come interfaccia grafica e interfaccia utente principale e il "server" come database back-end. Per accedere al database, queste applicazioni richiedono l'uso di un protocollo come ODBC. Con l'evoluzione delle applicazioni basate sul Web e delle architetture multilivello, vengono coinvolti molti server nell'elaborazione delle informazioni che vengono infine visualizzate sul browser dell'utente.

A seconda della parte del sistema destinata ai test, è necessaria la comprensione del protocollo appropriato da utilizzare. Pertanto, se è necessario eseguire test end-to-end che emulano l'attività dell'utente dal browser, verrà utilizzato un protocollo Web come HTTP/HTTPS. In questo modo, è possibile bypassare la GUI e incentrare i test sulla comunicazione e sulle attività dei server back-end.

4.2.2 Transazioni

Le transazioni descrivono l'insieme di attività eseguite da un sistema dal punto di inizio a quando uno o più processi (richieste, operazioni o processi operativi) sono stati completati. Il tempo di risposta delle transazioni può essere misurato ai fini della valutazione delle prestazioni del sistema. Durante un Performance Testing, queste misurazioni vengono utilizzate per identificare eventuali componenti che necessitano di correzione o ottimizzazione.

Le transazioni simulate possono includere il think time per rivelare meglio i tempi in cui un utente reale esegue un'azione (ad esempio, per premere il pulsante "INVIA"). Il tempo di risposta della transazione più il think time corrispondono al tempo trascorso per quella transazione.

I tempi di risposta della transazione raccolti durante il Performance Testing mostrano come questa misurazione cambi in base a diversi carichi imposti sul sistema. L'analisi potrebbe non mostrare alcun degrado sotto carico, mentre altre misurazioni potrebbero mostrare un degrado grave. Aumentando il carico e misurando i tempi di transazione sottostanti, è possibile correlare la causa del degrado con i tempi di risposta di una o più transazioni.

Le transazioni possono anche essere annidate in modo da poter misurare le attività individuali e aggregate. Questo può essere utile, ad esempio, per la comprensione dell'efficienza delle prestazioni di un sistema di ordinazione online. Il tester potrebbe decidere di misurare i singoli passaggi del processo d'ordine (ad es. ricerca di un articolo, aggiunta di un articolo al carrello, pagamento di un articolo, conferma dell'ordine) nonché l'intero processo di ordinazione. Annidando le transazioni, entrambe le serie di informazioni possono essere raccolte in un unico test.

4.2.3 Identificazione dei profili operativi

I profili operativi indicano modelli distinti di interazione con un'applicazione, ad esempio da utenti o altri componenti del sistema. È possibile specificare più profili operativi per una determinata applicazione. Questi possono essere combinati per creare il profilo di carico desiderato per il raggiungimento di particolari obiettivi di Performance Testing (si veda la Sezione 4.2.4).

La presente sezione descrive i seguenti passaggi principali per l'identificazione dei profili operativi:

1. Identificare i dati da raccogliere
2. Raccogliere i dati utilizzando una o più fonti
3. Valutare i dati per costruire i profili operativi

Identificazione dei dati

Laddove gli utenti interagiscono con il sistema in prova, vengono raccolti o stimati i seguenti dati al fine di modellare i loro profili operativi (ovvero, come interagiscono con il sistema):

- Diversi tipi di utenti e loro ruoli (ad es. utente standard, membro registrato, amministratore, gruppi di utenti con privilegi specifici).
- Diverse attività generiche eseguite da tali utenti/ruoli (ad es., navigazione di un sito Web per informazioni, ricerca di un prodotto specifico in un sito Web, esecuzione di attività specifiche per ruolo). Si noti che queste attività sono generalmente modellate al meglio con un alto livello di astrazione (ad esempio, a livello di processi aziendali o principali storie utenti).
- Numero stimato di utenti per ciascun ruolo/attività per unità di tempo in un determinato periodo. Queste informazioni saranno utili anche per la creazione successiva di profili di carico (si veda la Sezione 4.2.4).

Raccolta dei dati

I dati sopra citati possono essere raccolti da diverse fonti:

- Conduzione di interviste o seminari con gli stakeholder, quali Product Owner, responsabili delle vendite e (potenziali) utenti finali. Queste discussioni spesso rivelano i principali profili operativi degli utenti e forniscono risposte alla domanda fondamentale "A chi è destinata questa applicazione".
- I requisiti e le specifiche funzionali (ove disponibili) sono una preziosa fonte di informazioni sui modelli di utilizzo previsti che possono anche aiutare a identificare i tipi di utenti e i loro profili operativi. Laddove le specifiche funzionali siano espresse come User Story, il formato standard consente di identificare direttamente i tipi di utenti (ovvero, in quanto *<tipo di utente>,* voglio *<una capacità>* cosicché *<un vantaggio>).* Allo stesso modo, i diagrammi e le descrizioni dei casi d'uso UML identificano il soggetto per il caso d'uso.
- La valutazione dei parametri e dei dati di utilizzo acquisita da applicazioni simili può supportare l'identificazione delle tipologie di utenti e fornire alcune indicazioni iniziali del numero previsto di utenti. Si consiglia l'accesso ai dati monitorati in modo automatico (ad es. dallo strumento di amministrazione di un web master). Ciò includerà i registri di monitoraggio e i dati ricavati dall'uso dell'attuale sistema operativo, nel caso in cui sia previsto un aggiornamento a tale sistema
- Il monitoraggio del comportamento degli utenti durante l'esecuzione di attività predefinite con l'applicazione può fornire informazioni sui tipi di profili operativi da modellare per il Performance Testing. Si consiglia di coordinare questa attività con tutti i test di usabilità pianificati (soprattutto se è disponibile un laboratorio di usabilità).

Costruzione dei profili operativi

Per l'identificazione e la costruzione di profili operativi per gli utenti vengono realizzate le seguenti fasi:

- Viene adottato un approccio dall'alto verso il basso. Inizialmente vengono creati profili operativi ampi e relativamente semplici, che vengono ulteriormente suddivisi solo se ciò è necessario per raggiungere gli obiettivi del Performance Testing (si veda la Sezione 4.1.1)
- È possibile individuare particolari profili utente come rilevanti ai fini del Performance Testing, se questi comportano attività che vengono eseguite di frequente, richiedono transazioni critiche (ad alto rischio) o frequenti tra diversi componenti del sistema o richiedono potenzialmente il trasferimento di grandi volumi di dati.
- I profili operativi vengono rivisti e perfezionati con i principali stakeholder prima di essere utilizzati per la creazione di profili di carico (si veda la Sezione 4.2.4).

Il sistema in prova non è sempre soggetto a carichi imposti dall'utente. I profili operativi possono anche essere richiesti per il Performance Testing dei seguenti tipi di sistema (l'elenco non è esaustivo):

Sistemi di elaborazione batch offline
L'attenzione è posta principalmente sulla velocità di elaborazione del sistema di elaborazione batch (si veda la Sezione 4.2.5) e sulla sua capacità di completare il processo entro un determinato periodo di tempo. I profili operativi si concentrano sui tipi di elaborazione richiesti per i processi batch. Ad esempio, i profili operativi per un sistema di trading azionario (che generalmente include l'elaborazione di transazioni online e batch) possono includere quelli relativi alle transazioni di pagamento, alla verifica delle credenziali e alla verifica della conformità delle condizioni legali per particolari tipi di transazioni di borsa. Ciascuno di questi profili operativi comporterebbe percorsi diversi lungo l'intero processo batch per un titolo. I passaggi sopra descritti per l'identificazione dei profili operativi dei sistemi online basati sugli utenti possono essere applicati anche nel contesto dell'elaborazione batch.

Sistemi di sistemi
I componenti all'interno di un ambiente multi-sistema (che può anche essere embedded) rispondono a diversi tipi di input da altri sistemi o componenti. A seconda della natura del sistema in esame, può essere necessaria la modellizzazione di profili operativi diversi per rappresentare efficacemente le tipologie di input forniti da tali sistemi fornitore. Ciò può comportare un'analisi dettagliata (ad esempio buffer e code) insieme agli architetti di sistema e sulla base delle specifiche di sistema e di interfaccia.

4.2.4 Creazione di profili di carico

Un profilo di carico specifica l'attività a cui può essere sottoposto un componente o un sistema in fase di test in produzione. Consiste in un numero determinato di istanze che eseguiranno le azioni di profili operativi predefiniti per un periodo di tempo specificato. Se le istanze sono utenti, viene normalmente impiegato il termine "utenti virtuali".

Le principali informazioni richieste per creare un profilo di carico realistico e ripetibile sono:

- L'obiettivo del Performance Testing (ad esempio, valutare il comportamento del sistema in presenza di carichi di stress)
- I profili operativi che rappresentano accuratamente i singoli modelli di utilizzo (si veda la Sezione 4.2.3)
- Problemi noti di velocità di elaborazione e concorrenza (si veda la Sezione 4.2.5)
- La quantità e la distribuzione del tempo con cui i profili operativi devono essere eseguiti in modo tale che il SUT raggiunga il carico desiderato. Esempi tipici sono:
    - Incrementi: Carico costantemente crescente (ad esempio, aggiungere un utente virtuale al minuto)
    - Decrementi: Carico costantemente decrescente
    - Passi: Modifiche istantanee del carico (ad esempio, aggiungere 100 utenti virtuali ogni cinque minuti)
    - Distribuzioni predefinite (ad esempio, il volume imita i cicli aziendali giornalieri o stagionali)

L'esempio seguente mostra la costruzione di un profilo di carico con l'obiettivo di generare condizioni di stress (pari o superiori al massimo previsto per il sistema da gestire) per un sistema sotto test.

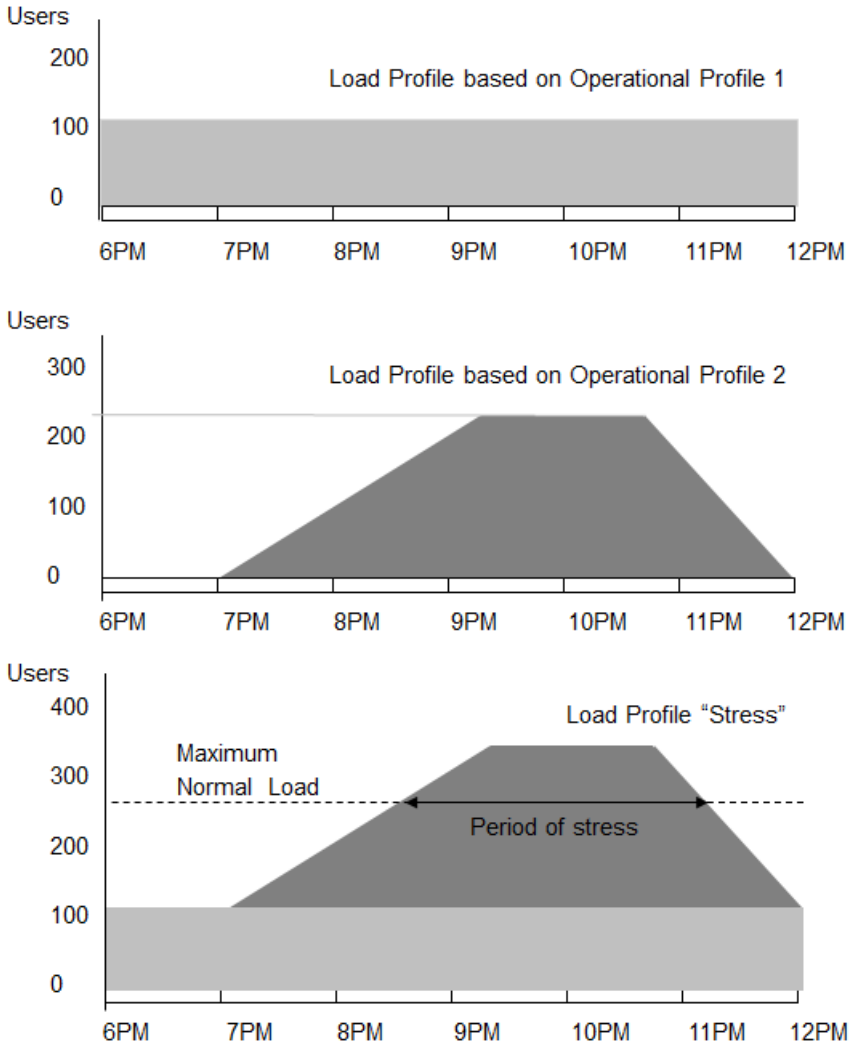

Diagramma 1: Esempio di costruzione di un profilo di carico "stress"

Nella parte superiore del diagramma viene mostrato un profilo di carico che consiste in un input di 100 utenti virtuali. Questi utenti eseguono le attività definite dal Profilo operativo 1 per l'intera durata del test. Questo è tipico di molti profili di carico di performance, che rappresentano un carico in background.

Il diagramma centrale mostra un profilo di carico costituito da un incremento fino a 220 utenti virtuali, che viene mantenuto per due ore prima di scendere. Ogni utente virtuale esegue attività definite nel Profilo operativo 2.

Il diagramma in basso mostra il profilo di carico risultante dalla combinazione dei due sopra descritti. Il sistema in prova è sottoposto a un periodo di stress di tre ore. Per ulteriori esempi, si veda [Bath14].

4.2.5 Analisi della velocità di elaborazione e della concorrenza

È importante comprendere aspetti diversi del carico di lavoro: velocità di elaborazione e concorrenza. Per modellare correttamente i profili operativi e di carico, è necessario prendere in considerazione entrambi gli aspetti.

Velocità di elaborazione del sistema (System Throughput)
La velocità di elaborazione del sistema è la misurazione del numero di transazioni di un determinato tipo che il sistema elabora in un'unità di tempo. Ad esempio, il numero di ordini all'ora o il numero di richieste HTTP al secondo. La velocità di elaborazione del sistema dovrebbe essere distinta dalla velocità di elaborazione della rete, che è la quantità di dati trasferiti sulla rete (Sezione 2.1).

La velocità di elaborazione del sistema definisce il carico sul sistema. Sfortunatamente, il numero di utenti simultanei viene utilizzato abbastanza spesso per definire il carico per i sistemi interattivi, anziché per la velocità di elaborazione. Ciò è in parte vero perché tale numero è spesso più facile da trovare e in parte perché è il modo in cui gli strumenti di test del carico definiscono il carico. Senza definire i profili operativi (cosa sta facendo ogni utente e con quale intensità, che è anche la velocità di elaborazione per un utente), il numero di utenti non risulta essere una buona misura del carico. Ad esempio, se ci sono 500 utenti al minuto che eseguono query brevi, la velocità di elaborazione è di 30.000 query all'ora. Se gli stessi 500 utenti eseguono le stesse query, ma una all'ora, la velocità è di 500 query all'ora. Quindi ci sono gli stessi 500 utenti, ma una differenza di 60x tra i carichi e una differenza di almeno 60x per i requisiti hardware del sistema.

La modellazione del carico di lavoro viene generalmente effettuata considerando il numero di utenti virtuali (thread di esecuzione) e il think time (ritardi tra le azioni dell'utente). Tuttavia, la velocità di elaborazione del sistema viene definita anche dal tempo di elaborazione e tale tempo può aumentare all'aumentare del carico.

Velocità di elaborazione del sistema = [numero di utenti virtuali] / ([tempo di elaborazione] + [think time])

Pertanto, quando aumenta il tempo di elaborazione, la velocità di elaborazione può diminuire in modo significativo anche se tutto il resto rimane invariato.

La velocità di elaborazione del sistema è un aspetto importante durante il test dei sistemi di elaborazione batch. In questo caso, la velocità di elaborazione viene in genere misurata in base al numero di transazioni che possono essere eseguite in un determinato intervallo di tempo (ad esempio, una finestra di elaborazione batch notturna).

Concorrenza

La concorrenza è una misura del numero di thread simultanei/paralleli in esecuzione. Per i sistemi interattivi, può essere un numero di utenti simultanei/paralleli. La concorrenza è in genere modellata negli strumenti di test del carico impostando il numero di utenti virtuali.
La concorrenza è una misura importante. Rappresenta il numero di sessioni parallele, ciascuna delle quali può utilizzare le proprie risorse. Anche se la velocità di elaborazione è la stessa, la quantità di risorse utilizzate può variare in base alla concorrenza. Le configurazioni tipiche per i test sono sistemi chiusi (dal punto di vista della teoria delle code), in cui è impostato il numero di utenti nel sistema (popolazione fissa). Se tutti gli utenti aspettano la risposta del sistema in un sistema chiuso, non possono arrivare nuovi utenti. Molti sistemi pubblici sono aperti: arrivano continuamente nuovi utenti, anche se tutti gli utenti attuali stanno aspettando la risposta del sistema.

4.2.6 Struttura di base di uno script di Performance Testing

Uno script di Performance Testing intende simulare un'attività di un utente o un componente, che contribuisce al carico sul sistema in prova (l'intero sistema o uno dei suoi componenti). Avvia le richieste al server in un ordine corretto e ad un determinato ritmo.

Il modo migliore per creare script di Performance Testing dipende dall'approccio di generazione del carico utilizzato (Sezione 4.1).

- Il modo tradizionale prevede la registrazione della comunicazione tra il client e il sistema o il componente a livello di protocollo e la riproduzione dopo che lo script è stato parametrizzato e documentato. La parametrizzazione si traduce in uno script scalabile e sostenibile, ma l'attività di parametrizzazione potrebbe richiedere molto tempo.
- La registrazione a livello di GUI in genere comporta l'acquisizione di azioni della GUI di un singolo client con uno strumento di esecuzione del test e l'esecuzione dello script con lo strumento di generazione del carico per rappresentare più client.
- È possibile eseguire la programmazione utilizzando richieste di protocollo (ad es. richieste HTTP), azioni della GUI o chiamate API. Nel caso degli script di programmazione, deve essere determinata l'esatta sequenza di richieste inviate e ricevute dal sistema reale, che potrebbe anche essere non banale.

Di solito uno script è una o più sezioni di codice (scritte in un linguaggio di programmazione generico con alcune estensioni o in un linguaggio specializzato) o un oggetto, che può essere presentato a un utente dallo strumento in una GUI. In entrambi i casi lo script includerà le richieste del server che creano il carico (ad es. richieste HTTP) e alcune relative logiche di programmazione, specificando come esattamente queste richieste sarebbero richiamate (ad es. in quale ordine, in quale momento, con quali parametri, cosa dovrebbe essere verificato). Più sofisticata è la logica, maggiore è la necessità di utilizzare linguaggi di programmazione potenti.

Struttura generale
Spesso lo script ha una sezione di inizializzazione (in cui il tutto viene preparato per la parte principale), sezioni principali che possono essere eseguite più volte e una sezione di pulizia (in cui vengono adottate le misure necessarie per completare correttamente il test).

Raccolta dati
Per raccogliere i tempi di risposta, è necessario aggiungere timer allo script per misurare il tempo impiegato da una richiesta o da una combinazione di richieste. Le richieste a tempo devono corrispondere a un'unità significativa di lavoro logico, ad esempio una transazione commerciale per l'aggiunta di un articolo a un ordine o l'invio di un ordine.

È importante capire cosa viene misurato esattamente: nel caso di script a livello di protocollo, è solo il tempo di risposta del server e della rete, mentre gli script GUI misurano il tempo end-to-end (sebbene ciò che viene misurato esattamente dipende dalla tecnologia utilizzata).

Verifica dei risultati e gestione degli errori
Una parte importante dello script è la verifica dei risultati e la gestione degli errori. Anche nei migliori strumenti di test di carico, la gestione degli errori predefinita tende a essere minima (come il controllo del codice di ritorno della richiesta HTTP), quindi si consiglia di aggiungere ulteriori controlli per verificare quello che viene effettivamente restituito dalle richieste. Inoltre, se è necessaria una pulizia in caso di errore, è probabile che debba essere implementata manualmente. Una buona pratica consiste nel verificare che lo script stia facendo quanto previsto, utilizzando metodi indiretti, ad esempio controllando il database per verificare che siano state aggiunte le informazioni appropriate.

Gli script possono includere altre logiche che indicano regole su quando e come verranno eseguite le richieste del server. Un esempio è l'impostazione dei punti di sincronizzazione, che viene eseguita specificando che lo script deve attendere un evento in quel punto prima di procedere. È possibile utilizzare i punti di sincronizzazione per garantire che un'azione specifica venga richiamata contemporaneamente o per coordinare il lavoro tra più script.

Gli script di Performance Testing sono software, quindi la creazione di uno script di Performance Testing è un'attività di sviluppo software. Deve quindi comprendere la garanzia di qualità e test per verificare che lo script funzioni come previsto con l'intera gamma di dati di input.

4.2.7 Implementazione di script di Performance Testing

Gli script di Performance Testing vengono implementati in base al PTP e ai profili di carico. Mentre i dettagli tecnici sull'attuazione saranno diversi a seconda dell'approccio e degli strumenti utilizzati, il processo complessivo rimane lo stesso. Uno script di performance viene creato utilizzando un ambiente di sviluppo integrato (IDE) o un editor di script, per simulare il comportamento di un utente o di un componente. Di solito lo script viene creato per simulare un profilo operativo specifico (sebbene sia spesso possibile combinare più profili operativi in uno script con istruzioni condizionali).

Quando viene determinata la sequenza di richieste, è possibile registrare o programmare lo script, a seconda dell'approccio. Normalmente la registrazione garantisce la simulazione esatta del sistema reale, mentre la programmazione si basa sulla conoscenza della sequenza di richieste corretta.

Se viene impiegata la registrazione a livello di protocollo, un passaggio essenziale dopo la registrazione, nella maggior parte dei casi, è la sostituzione di tutti gli elementi identificativi interni registrati che definiscono il contesto. Questi identificatori devono essere trasformati in variabili che possono essere modificate tra esecuzioni, con valori appropriati che vengono estratti dalle risposte alla richiesta (ad esempio, un identificativo utente che viene acquisito al momento dell'accesso e deve essere fornito per tutte le transazioni successive). Questa è una parte della parametrizzazione degli script, a volte definita "correlazione". In quel contesto, la parola correlazione ha un significato diverso rispetto a quando viene usata in statistica (dove significa relazione tra due o più cose). Gli strumenti avanzati di test del carico possono eseguire automaticamente alcune correlazioni, quindi in alcuni casi possono essere trasparenti, ma nei casi più complessi, potrebbe essere necessaria una correlazione manuale o l'aggiunta di nuove regole di correlazione. La correlazione errata o la mancanza di correlazione è il motivo principale per cui la riproduzione degli script registrati non riesce.

L’esecuzione di più utenti virtuali con lo stesso nome utente e l'accesso allo stesso set di dati (come di solito accade durante la riproduzione di uno script registrato senza ulteriori modifiche oltre alla necessaria correlazione) porta facilmente a risultati fuorvianti. I dati potrebbero essere interamente memorizzati nella cache (copiati dal disco alla memoria per accedervi più rapidamente) e i risultati sarebbero molto migliori rispetto alla produzione (dove tali dati possono essere letti da un disco). L'uso degli stessi utenti e/o dati può anche causare problemi di concorrenza (ad esempio, se i dati sono bloccati quando un utente li sta aggiornando) e i risultati sarebbero molto peggiori rispetto alla produzione, in quanto il software aspetterebbe che il blocco si liberi prima che l’utente successivo possa bloccare i dati per l'aggiornamento.

Pertanto, gli script e i test harness devono essere parametrizzati (ovvero, i dati fissi o registrati devono essere sostituiti con i valori di un elenco di possibili scelte), in modo che ciascun utente virtuale utilizzi un set di dati adeguato. Il termine "adeguato" in

questo caso significa abbastanza diverso da evitare problemi con la cache e la concorrenza, che è specifica per i requisiti di test, dati e sistema. Questa ulteriore parametrizzazione dipende dai dati nel sistema e dal modo in cui il sistema lavora con questi dati, quindi di solito viene eseguita manualmente, sebbene molti strumenti offrano assistenza.

Esistono casi in cui alcuni dati devono essere parametrizzati affinché il test funzioni più di una volta, ad esempio quando viene creato un ordine, il nome deve essere univoco. A meno che il nome dell'ordine non sia parametrizzato, il test fallirà non appena tenta di creare un ordine con un nome già esistente (registrato).

Per abbinare i profili operativi, i think time devono essere inseriti e/o adattati (se registrati), per generare un numero adeguato di richieste/velocità di elaborazione, come illustrato nella Sezione 4.2.5.

Quando vengono creati script per profili operativi separati, questi vengono combinati in uno scenario che applica l'intero profilo di carico. Il profilo di carico controlla quanti utenti virtuali vengono avviati utilizzando ogni script, quando e con quali parametri. I dettagli esatti dell'implementazione dipendono dallo specifico strumento di test di carico o dallo specifico test harness.

4.2.8 Preparazione all'esecuzione del Performance Testing

Le principali attività per la preparazione all'esecuzione del Performance Testing includono:

- Installazione del sistema da testare
- Mettere a disposizione l'ambiente
- Messa a punto degli strumenti di generazione e monitoraggio del carico e verifica che tutte le informazioni necessarie potranno essere raccolte

È importante assicurarsi che l'ambiente di test sia il più vicino possibile all'ambiente di produzione. Se ciò non è possibile, è necessario comprendere chiaramente le differenze e come verranno proiettati i risultati dei test nell'ambiente di produzione. Idealmente, verrebbero utilizzati ambienti di produzione e dati reali, ma i test in un ambiente ridotto potrebbero comunque contribuire a mitigare una serie di rischi sulle performance.

È importante ricordare che le prestazioni sono una funzione non lineare dell'ambiente, quindi più l'ambiente si discosta dallo standard di produzione, più diventa difficile fare proiezioni accurate per le performance in produzione. La mancanza di affidabilità delle proiezioni e l'aumento del livello di rischio crescono man mano che il sistema di test assomiglia meno a quello di produzione.

Le parti più importanti dell'ambiente di test sono: dati, configurazione hardware e software e configurazione di rete. Le dimensioni e la struttura dei dati potrebbero

influire notevolmente sui risultati dei test di carico. L'uso di un set di dati campione ridotto o un set campione con una complessità dei dati diversa per il Performance Testing può dare risultati fuorvianti, in particolare quando il sistema di produzione utilizzerà un set di dati ampio. È difficile prevedere in che misura la dimensione dei dati influisca sulle prestazioni prima di eseguire test reali. Più i dati del test sono simili ai dati di produzione in termini di dimensioni e struttura, più affidabili saranno i risultati del test.

Se i dati vengono generati o modificati durante il test, potrebbe essere necessario ripristinare i dati originali prima del successivo ciclo di test, così da garantire che il sistema si trovi nelle giuste condizioni.

Se alcune parti del sistema o alcuni dati non sono disponibili per il Performance Testing, per qualsiasi motivo, è necessario adottare una soluzione alternativa. Ad esempio, è possibile implementare uno stub per sostituire ed emulare un componente terzo responsabile dell'elaborazione della carta di credito. Tale processo viene spesso definito "virtualizzazione del servizio". Vengono messi a disposizione strumenti speciali. L'uso di tali strumenti è fortemente raccomandato per isolare il sistema in prova.

Esistono molti modi per implementare gli ambienti. Ad esempio, le opzioni possono includere l'utilizzo di:

- Laboratori di test tradizionali interni (ed esterni)
- Il cloud come ambiente che utilizza l'Infrastructure as a Service (IaaS), quando alcune parti o l'intero sistema o il sistema viene implementato sul cloud
- Il cloud come ambiente che utilizza il Software as a Service (SaaS), quando i fornitori offrono il servizio di test di carico

A seconda degli obiettivi specifici e dei sistemi da testare, potrebbe essere preferibile un ambiente di test rispetto a un altro. Per esempio,

- Per testare l'effetto di un miglioramento delle prestazioni (ottimizzazione delle prestazioni), l'uso di un ambiente di laboratorio isolato può essere un'opzione migliore per vedere anche le piccole variazioni introdotte dalla modifica.
- Per eseguire il test di carico sull'intero ambiente di produzione end-to-end per assicurarsi che il sistema gestirà il carico senza problemi gravi, i test dal cloud o da un servizio potrebbero risultare più appropriati. (Si noti che questo funziona solo per i SUT raggiungibili da un cloud).
- Per ridurre al minimo i costi quando il Performance Testing è limitato nel tempo, la creazione di un ambiente di test nel cloud può essere una soluzione più economica.

Qualunque sia l'approccio all'implementazione adottato, sia l'hardware che il software devono essere configurati per soddisfare l'obiettivo e il piano del test. Se l'ambiente corrisponde alla produzione, questo dev'essere configurato allo stesso modo.

Tuttavia, se ci sono differenze, potrebbe essere necessario regolare la configurazione per adattarsi alle differenze. Ad esempio, se le macchine di prova hanno meno memoria fisica rispetto alle macchine di produzione, potrebbe essere necessario regolare i parametri di memoria del software (come la dimensione heap Java) per evitare la paginazione della memoria.

Una corretta configurazione/emulazione della rete è importante per i sistemi globali e mobili. Per i sistemi globali (ovvero con utenti o processi distribuiti in tutto il mondo) uno degli approcci potrebbe essere quello di implementare generatori di carico nei luoghi in cui si trovano gli utenti. Per i sistemi mobili, l'emulazione della rete rimane l'opzione più praticabile a causa delle varianze nei tipi di rete che è possibile utilizzare. Alcuni strumenti di test di carico dispongono di strumenti integrati di emulazione di rete ed esistono strumenti autonomi per l'emulazione di rete.

Gli strumenti di generazione del carico devono essere eseguiti correttamente e gli strumenti di monitoraggio devono essere configurati in modo da raccogliere tutti i parametri necessari per il test. L'elenco dei parametri dipende dagli obiettivi del test, ma si consiglia di raccogliere almeno i parametri di base comuni a tutti i test (si veda la Sezione 2.1.2).

A seconda del carico, dell'approccio per la generazione del carico o dello strumento specifico e in base alla configurazione della macchina, potrebbe essere necessaria più di una macchina per la generazione del carico. Per verificare l'installazione, è necessario monitorare anche le macchine coinvolte nella generazione del carico. Ciò contribuirà ad evitare una situazione in cui il carico non viene mantenuto correttamente perché uno dei generatori di carico funziona con lentezza.

A seconda della configurazione e degli strumenti utilizzati, gli strumenti di test di carico devono essere configurati in modo da creare il carico appropriato. Ad esempio, è possibile impostare specifici parametri di emulazione del browser o utilizzare l'IP spoofing (simulando che ciascun utente virtuale abbia un indirizzo IP diverso).

Prima di eseguire i test, è necessario convalidare l'ambiente e l'installazione. Per questo normalmente viene condotta una serie controllata di test, verificando l'esito dei test e che gli strumenti di monitoraggio stiano tracciando le informazioni importanti.

Per verificare che il test funzioni come previsto, è possibile utilizzare varie tecniche, tra cui l'analisi dei log e la verifica del contenuto dei database. La preparazione per il test include la verifica che le informazioni richieste vengano registrate, che il sistema sia nello stato corretto, ecc. Ad esempio, se il test modifica significativamente lo stato del sistema (aggiunge/cambia le informazioni nel database), potrebbe essere necessario riportare il sistema allo stato originale prima di ripetere il test.

4.3 Esecuzione

L'esecuzione del Performance Testing comporta la generazione di un carico rispetto al SUT, in base a un profilo di carico (solitamente implementato da script di Performance Testing richiamati in base a un determinato scenario), il monitoraggio di tutte le parti dell'ambiente e la raccolta e la conservazione di tutti i risultati e le informazioni relativi al test. Solitamente i collegamenti/strumenti avanzati per i test di carico eseguono queste attività automaticamente (previa, ovviamente, una corretta configurazione). Forniscono normalmente una console per consentire il monitoraggio dei dati sulle prestazioni durante il test e consentire le necessarie regolazioni (si veda la Sezione 5.1). Tuttavia, a seconda dello strumento utilizzato, del SUT e dei test specifici in esecuzione, potrebbe essere necessario eseguire alcuni passaggi manuali.

Il Performance Testing è generalmente incentrato su uno stato costante del sistema, cioè quando il comportamento del sistema è stabile. Ad esempio, quando tutti gli utenti/thread simulati vengono avviati e stanno eseguendo il lavoro come previsto. Quando il carico cambia (ad esempio, quando vengono aggiunti nuovi utenti), il comportamento del sistema si modifica e diventa più difficile monitorare e analizzare i risultati dei test. Il passaggio per raggiungere una condizione costante viene spesso definito incremento, mentre la fase di completamento del test viene spesso definita decremento .

Talvolta è importante testare gli stati transitori, quando il comportamento del sistema cambia. Ciò può applicarsi, ad esempio, alla registrazione simultanea di un gran numero di utenti o agli spike testing. Quando si verificano stati transitori, è importante comprendere la necessità di un attento monitoraggio e analisi dei risultati, poiché alcuni approcci standard, come il monitoraggio delle medie, possono essere molto fuorvianti.

Durante l'incremento è consigliabile implementare stati di carico incrementale per monitorare l'effetto di un carico in costante aumento sulla risposta del sistema. Ciò garantisce che sia dedicato un tempo sufficiente per l'incremento e che il sistema sia in grado di gestire il carico. Una volta raggiunto lo stato costante, è buona norma monitorare che sia il carico, sia le risposte del sistema siano stabili e che le eventuali variazioni (che sono sempre presenti) non siano sostanziali.

È importante specificare come gestire gli errori per assicurarsi che non vengano introdotti problemi di sistema. Ad esempio, può essere importante che l'utente si disconnetta quando si verifica un errore, così da garantire lo sblocco di tutte le risorse associate a tale utente.

Se il monitoraggio è integrato nello strumento dei test di carico ed è correttamente configurato, di solito si avvia contemporaneamente all'esecuzione del test. Tuttavia, se si utilizzano strumenti di monitoraggio autonomi, il monitoraggio deve essere avviato separatamente e le informazioni necessarie devono essere raccolte in modo tale da

poter effettuare l'analisi successiva insieme ai risultati del test. Lo stesso vale per l'analisi dei log. È essenziale sincronizzare nel tempo tutti gli strumenti utilizzati, in modo da poter individuare tutte le informazioni relative a uno specifico ciclo di esecuzione del test.

L'esecuzione del test viene spesso monitorata utilizzando la console dello strumento di Performance Testing e l'analisi del log in tempo reale per verificare la presenza di problemi ed errori sia nel test, sia nel SUT. Questo aiuta ad evitare la prosecuzione inutile di test su larga scala, che potrebbero anche avere un effetto su altri sistemi in caso di eventi negativi (ad esempio, errori, guasto dei componenti o se i carichi generati sono insufficienti o eccessivi). Questi test possono essere costosi da eseguire e potrebbe essere necessario interrompere il test o apportare alcune modifiche estemporanee al Performance Testing o alla configurazione del sistema, se il test si discosta dal comportamento previsto.

Una tecnica per verificare i test di carico che comunicano direttamente a livello di protocollo prevede l'esecuzione di diversi script (funzionali) a livello di GUI o persino l'esecuzione manuale di profili operativi simili in parallelo al test di carico in corso. Si verifica così che i tempi di risposta segnalati durante il test differiscano dai tempi di risposta misurati manualmente solamente a livello di GUI dal tempo trascorso sul lato client.

In alcuni casi, quando si eseguono i test di performance in modo automatizzato (ad esempio, nell'ambito dell'Integrazione continua, come spiegato nella Sezione 3.4), i controlli devono essere eseguiti automaticamente, poiché il monitoraggio e l'intervento manuali potrebbero non essere possibili. In questo caso, la configurazione del test dovrebbe essere in grado di riconoscere eventuali deviazioni o problemi ed emettere un avviso (di solito mentre si completa correttamente il test). Questo approccio è più facile da implementare per il Performance Testing di regressione quando il comportamento del sistema è generalmente noto, ma può essere più difficile con il Performance Testing esplorativo o con costosi Performance Testing su larga scala, che potrebbero richiedere regolazioni dinamiche durante il test.

4.4 Analisi dei risultati e reporting

La Sezione 4.1.2 ha illustrato i vari parametri in un piano di Performance Testing. La definizione preventiva dei parametri determina cosa debba essere misurato per ciascuna prova. Dopo il completamento di un ciclo di test, è necessario raccogliere i dati per i parametri definiti.

Quando si analizzano i dati, questi vengono prima confrontati con l'obiettivo del Performance Testing. Una volta compreso il comportamento, è possibile trarre conclusioni che forniscono un rapporto di sintesi significativo che include le azioni consigliate. Queste azioni possono includere la modifica di componenti fisici (ad es.

hardware, router), la modifica del software (ad es. ottimizzazione di applicazioni e chiamate al database) e l'alterazione della rete (ad es. bilanciamento del carico, routing).

I seguenti dati vengono tipicamente analizzati:

- **Stato degli utenti simulati (ad esempio, virtuali)**. Questo punto deve essere esaminato per primo. Normalmente si prevede che tutti gli utenti simulati siano stati in grado di eseguire le attività specificate nel profilo operativo. Qualsiasi interruzione a questa attività simulerebbe ciò che potrebbe sperimentare un utente reale. Ciò rende molto importante innanzitutto verificare che tutte le attività dell'utente siano state completate, poiché eventuali errori riscontrati possono influenzare gli altri dati sulle prestazioni.
- **Tempo di risposta della transazione**. Questo può essere misurato in diversi modi, tra cui minimo, massimo, medio e un percentile (ad es. 90°). Le letture di minima e massima mostrano gli estremi delle prestazioni del sistema. Le prestazioni medie non sono necessariamente indicative di qualcosa di diverso dalla media matematica e spesso possono essere distorte dai valori anomali. Il 90° percentile viene spesso utilizzato come obiettivo poiché rappresenta la maggioranza degli utenti che raggiungono una soglia di prestazione specifica. Non è consigliabile richiedere il 100% di conformità con gli obiettivi di performance, poiché le risorse richieste potrebbero essere troppo grandi e l'effetto netto per gli utenti sarà spesso minore.
- **Transazioni al secondo**. Questo fornisce informazioni su quanto lavoro è stato svolto dal sistema (produttività del sistema).
- **Failure nelle transazioni**. Questi dati vengono utilizzati per l'analisi delle transazioni al secondo. I failure indicano che l'evento o il processo previsto non è stato completato o non è stato eseguito. Eventuali failure riscontrati sono motivo di preoccupazione ed è quindi necessario ricercarne la causa principale. Le transazioni non riuscite possono anche comportare numeri non validi di transazioni al secondo, dato che una transazione non riuscita richiederà molto meno tempo di una completata.
- **Accessi (o richieste) al secondo**. Dà un'idea del numero di accessi a un server da parte degli utenti simulati per ogni secondo del test.
- **Velocità di elaborazione di rete.** Questo valore viene di solito misurato in bit per intervallo di tempo, ad esempio in bit al secondo. Rappresenta la quantità di dati che gli utenti simulati ricevono dal server ogni secondo. (si veda la Sezione 4.2.5)
- **Risposte HTTP.** Queste vengono misurate al secondo e includono possibili codici di risposta, come: 200, 302, 304, 404, quest'ultimo indica che non è stata trovata una pagina.

Sebbene gran parte di queste informazioni possano essere presentate in tabelle, le rappresentazioni grafiche semplificano la visualizzazione dei dati e l'identificazione delle tendenze.

Le tecniche utilizzate per l'analisi dei dati possono includere:

- Confronto dei risultati con i requisiti dichiarati
- Osservazione delle tendenze nei risultati
- Tecniche statistiche di controllo della qualità
- Individuazione degli errori
- Confronto dei risultati attesi e reali
- Confronto dei risultati con i risultati dei test precedenti
- Verifica del corretto funzionamento dei componenti (ad es. server, reti)

L'identificazione della correlazione tra i parametri può aiutarci a capire a che punto le prestazioni del sistema iniziano a peggiorare. Ad esempio, quante transazioni al secondo sono state elaborate quando la CPU ha raggiunto la capacità del 90% e il sistema ha rallentato?

L'analisi può aiutare a identificare la causa principale del degrado o del crollo delle prestazioni, che a sua volta agevolerà la correzione. I test confermativi aiuteranno a capire se l'azione correttiva ha risolto la causa principale.

Reporting
I risultati delle analisi sono consolidati e confrontati con gli obiettivi dichiarati nel piano di Performance Testing. Questi possono essere inseriti nel rapporto generale sullo stato dei test, insieme ad altri risultati dei test, o inclusi in un rapporto dedicato per il Performance Testing. Il livello di dettaglio riportato dovrebbe corrispondere alle esigenze degli stakeholder. Le raccomandazioni basate su questi risultati in genere riguardano i criteri di pubblicazione del software (incluso l'ambiente di destinazione) o i miglioramenti delle prestazioni richiesti.

Un rapporto di Performance Testing classico include:

Sintesi operativa
Questa sezione viene completata una volta eseguiti tutti i test di performance e quando tutti i risultati sono stati analizzati e compresi. L'obiettivo è presentare conclusioni, scoperte e raccomandazioni concise e comprensibili per la gestione, con l'obiettivo di ottenere un risultato attuabile.

Risultati del test
I risultati del test possono includere alcune o tutte le seguenti informazioni:

- Un riassunto che fornisce una spiegazione e l'elaborazione dei risultati.
- Risultati di un test di base che funge da "istantanea" delle prestazioni del sistema in un determinato momento e costituisce la base del confronto con i test successivi. I risultati dovrebbero includere la data e l'ora di inizio del test, l'obiettivo degli utenti concorrenti, la velocità di elaborazione misurata e i risultati

chiave. I risultati fondamentali possono includere il tasso di errore complessivo misurato, il tempo di risposta e la velocità di elaborazione media.
- Un diagramma di alto livello che mostra i componenti dell'architettura che potrebbero (o hanno avuto) conseguenze sugli obiettivi del test.
- Un'analisi dettagliata (tabelle e grafici) dei risultati del test che mostra i tempi di risposta, i tassi di transazione, i tassi di errore e l'analisi delle prestazioni. L'analisi include anche una descrizione di ciò che è stato osservato, come ad esempio a che punto un'applicazione stabile è diventata instabile e l'origine degli errori (ad esempio, server Web, server di database).

Log di test / Informazioni registrate

È opportuno tenere un log relativo a ogni test eseguito. Normalmente il registro include:
- Data/ora di inizio del test
- Durata del test
- Script utilizzati per il test (inclusa la combinazione di script se ne vengono utilizzati vari) e dati rilevanti di configurazione degli script
- File di dati del test utilizzati
- Nome e posizione dei file di dati/log creati durante il test
- Configurazione HW/SW testata (in particolare eventuali modifiche tra le esecuzioni)
- Utilizzo medio e massimo di CPU e RAM su server Web e database
- Note sulle prestazioni raggiunte
- Errori identificati

Raccomandazioni

Le raccomandazioni risultanti dai test possono includere:
- Modifiche tecniche consigliate, come la riconfigurazione di hardware o software o infrastruttura di rete
- Aree identificate per ulteriori analisi (ad es. analisi dei log del server Web per aiutare a identificare le cause di problemi e/o errori)
- Richiesta di un monitoraggio aggiuntivo di gateway, server e reti in modo da poter ottenere dati più dettagliati per la misurazione delle caratteristiche e delle tendenze delle prestazioni (ad es. degrado)

5. Strumenti: 90 min.

Parole chiave

generatore di carico, gestione del carico, strumento di monitoraggio, strumento di Performance Testing

Obiettivi formativi

5.1 Supporto strumentale
PTFL-5.1.1 (K2) Comprendere come gli strumenti supportano il Performance Testing

5.2 Idoneità dello strumento
PTFL-5.2.1 (K4) Valutazione dell'idoneità degli strumenti di Performance Testing in un determinato scenario di progetto

5.1 Supporto strumentale

Gli strumenti di Performance Testing includono i seguenti tipi di strumenti per supportare il Performance Testing.

Generatori di carico
Il generatore, tramite un IDE, un editor di script o una suite di strumenti, è in grado di creare ed eseguire più istanze client che simulano il comportamento dell'utente in base a un profilo operativo definito. La creazione di più istanze in brevi periodi di tempo porterà a un carico sul sistema in prova. Il generatore crea il carico e raccoglie anche i parametri per la redazione di report.

Quando si eseguono i test di performance, l'obiettivo del generatore di carico è quello di imitare il mondo reale per quanto possibile. Questo spesso significa che sono necessarie richieste degli utenti provenienti da vari luoghi e non solo da dove si eseguono i test. Gli ambienti che sono dotati di più punti di presenza distribuiranno la provenienza del carico, in modo che non venga tutto da una sola rete. In questo modo il test risulta più realistico, sebbene a volte possa distorcere i risultati se i rimbalzi intermedi della rete (network hops) creano ritardi.

Console di gestione del carico
La console di gestione del carico dà il comando per avviare e arrestare il generatore di carico. Inoltre, la console aggrega i parametri delle varie transazioni definite entro le istanze di carico utilizzate dal generatore. La console consente di visualizzare report e grafici delle esecuzioni di test e supporta l'analisi dei risultati.

Strumento di monitoraggio

Gli strumenti di monitoraggio vengono eseguiti contemporaneamente al componente o al sistema sotto test e controllano, registrano e/o analizzano il comportamento del componente o del sistema. I componenti tipici che vengono monitorati includono code del server Web, memoria di sistema e spazio su disco. Gli strumenti di monitoraggio possono supportare efficacemente l'analisi della causa principale del degrado delle prestazioni in un sistema sotto test e possono anche essere utilizzati per monitorare un ambiente di produzione all'uscita del prodotto. Durante l'esecuzione del Performance Testing, è possibile utilizzare i monitor anche sullo stesso generatore di carico.

I modelli di licenza per strumenti di Performance Testing comprendono la tradizionale licenza per postazione/sito con piena proprietà, un modello di licenza pay-as-you-go basato su cloud e licenze open source che possono essere utilizzate gratuitamente in un ambiente definito o tramite offerte basate su cloud. Ciascun modello implica una diversa struttura dei costi e può prevedere una manutenzione continua. Risulta chiaro che per qualsiasi strumento selezionato, la comprensione del suo funzionamento (attraverso la formazione e/o lo studio autonomo) richiederà tempo e budget.

5.2 Idoneità dello strumento

I seguenti fattori devono essere tenuti in considerazione per la scelta di uno strumento di Performance Testing:

Compatibilità
In generale, viene selezionato uno strumento adatto all'organizzazione e non solo per un progetto. Ciò significa prendere in considerazione i seguenti fattori nell'organizzazione:

- Protocolli: Come descritto nella Sezione 4.2.1, i protocolli sono un aspetto molto importante nella scelta degli strumenti di performance. Comprendere quali protocolli siano utilizzati da un sistema e quali saranno testati permetterà di avere le informazioni necessarie per valutare quale strumento di test sia appropriato.
- Interfacce con componenti esterni: Potrebbe essere necessario considerare le interfacce con i componenti software o altri strumenti come parte dei requisiti di integrazione completi per soddisfare i requisiti di processo o altri requisiti di interoperabilità (ad esempio, l'integrazione nel processo di CI).
- Piattaforme: La compatibilità con le piattaforme (e le loro versioni) all'interno di un'organizzazione è essenziale. Questo vale per le piattaforme utilizzate per ospitare gli strumenti e per quelle con cui gli strumenti interagiscono per il monitoraggio e/o la generazione del carico.

Scalabilità
Un altro fattore da considerare è il numero totale di simulazioni di utenti simultanei che lo strumento è in grado di gestire. Ciò includerà diversi fattori:

- Numero massimo di licenze richieste
- Requisiti di configurazione della workstation/server di generazione del carico
- Capacità di generare carico da più punti di presenza (ad es. server distribuiti)

Comprensibilità
Un altro fattore da considerare è il livello di conoscenza tecnica necessaria per utilizzare lo strumento. Questo è spesso trascurato e può portare a tester non qualificati che configurano erroneamente i test, che a loro volta forniscono risultati imprecisi. Per i test che richiedono scenari complessi e un livello elevato di programmabilità e personalizzazione, i team devono assicurarsi che il tester abbia le competenze, il background e la formazione necessari.

Monitoraggio
Il monitoraggio fornito dallo strumento è sufficiente? Nell'ambiente sono disponibili altri strumenti di monitoraggio, che possono essere utilizzati per integrare il monitoraggio da parte dello strumento? È possibile correlare il monitoraggio con le transazioni definite? È necessario rispondere a tutte queste domande per determinare se lo strumento fornirà un monitoraggio adeguato al progetto.

Se il monitoraggio è costituito da un programma, strumenti, intero stack separati, può essere utilizzato per controllare l'ambiente di produzione al lancio del prodotto.

6. Riferimenti

6.1 Standards

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

6.2 Documenti ISTQB

- [ISTQB_UT_SYL] ISTQB Foundation Level Usability Testing Syllabus, Version 2018
- [ISTQB_ALTA_SYL] ISTQB Advanced Level Test Analyst Syllabus, Version 2012
- [ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 2012
- [ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 2012
- [ISTQB_FL_SYL] ISTQB Foundation Level (Core) Syllabus, Version 2018
- [ISTQB_FL_AT] ISTQB Foundation Level Agile Tester Syllabus, Version 2014
- [ISTQB_GLOSSARY] ISTQB Glossary of Terms used in Software Testing, http://glossary.istqb.org

6.3 Libri

[Anderson01] Lorin W. Anderson, David R. Krathwohl (a cura di) “A Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom’s Taxonomy of Educational Objectives”, Allyn & Bacon, 2001, ISBN 978-0801319037

[Bath14] Graham Bath, Judy McKay, “The Software Test Engineer’s Handbook”, Rocky Nook, 2014, ISBN 978-1-933952-24-6

[Molyneaux09] Ian Molyneaux, “The Art of Application Performance Testing: From Strategy to Tools”, O’Reilly, 2009, ISBN: 9780596520663

[Microsoft07] Microsoft Corporation, “Performance Testing Guidance for Web Applications”, Microsoft, 2007, ISBN: 9780735625709

7. Indice