WHITE PAPER | APRILE 2016

Sopravvivere all'approccio "shift-left" come tester

In una serie di articoli Paul Gerrard, guru del testing e consulente, discute di argomenti collegati all'argomento testing. Per "shift-left" si intende il trasferimento del testing in una fase anticipata del processo. In questo articolo, Paul parla delle motivazioni alla base di questo approccio incentrato sull'anticipazione, di come il ruolo del testing stia cambiando e dell'effetto di questo sui tester in un mondo più Agile.

Paul Gerrard
Gerrard Consulting

Sponsorizzato da

ca technologies

Sopravvivere all'approccio "shift-left" come tester

Contesto

Circa cinque anni fa ho coniato l'espressione "testing ridistribuito" per descrivere un cambiamento che stava avvenendo nel business di testing. Utenti, analisti, sviluppatori e tester ridistribuiscono la responsabilità per i test e collaborano in modo più efficace. Il cambiamento comporta l'anticipazione delle attività di test (e possibilmente delle relative responsabilità), e per definire questo approccio si utilizza comunemente la denominazione di "shift-left".

L'approccio "shift-left" può significare che gli sviluppatori assumono maggiore controllo e responsabilità per i propri test; può anche significare che i tester vengono coinvolti in una fase anticipata, discutono eventuali requisiti e forniscono esempi agli sviluppatori attraverso un processo di sviluppo basato sul comportamento (BDD, Behaviour Driven Development). Può significare che utenti e analisti di business, insieme agli sviluppatori, si assumono la piena responsabilità delle attività di test; ma, anche, l'eliminazione di team di test e tester. Abbiamo visto realizzate tutte le configurazioni, e naturalmente una risposta unica non esiste.

L'articolo illustra come il ruolo del testing stia cambiando e l'effetto di questo sui tester in un mondo più Agile.

Il concetto di "shift-left" non è nuovo

È da quando lavoro nel settore (1992), e certamente anche da prima, che i sostenitori del testing predicano il mantra del "test early, test often" (test prima e spesso). Il modello W[1], introdotto dal mio amico e collega Paul Herzlich già nel 1993, suggeriva che tutti gli elementi in un processo in fasi, documenti e software, possono (e spesso devono) essere testati.

Anche se in quel momento l'approccio dominante al ciclo di vita era quello a cascata, il numero o la durata delle fasi non sono l'elemento essenziale. Il principio alla base di tutto è che *le fonti di conoscenza che determinano la direzione della progettazione e dello sviluppo del software devono essere messe in discussione o testate*. In un progetto in fasi, questo potrebbe comportare revisioni formali. In un progetto Agile, il tester (o lo sviluppatore o l'analista di business o l'utente) può suggerire scenari (o esempi) che portano l'autore di un requisito o di una story a considerare con attenzione esempi concreti e una certa discussione prima della scrittura del codice.

Per "shift-left" si intende il trasferimento del testing in una fase anticipata del processo.

Quindi, tutto ciò che l'approccio "shift-left" richiede è che i tester siano coinvolti in una fase precoce e che vengano poste loro le domande davvero rilevanti? È davvero così semplice? Non proprio.

Quali sono i driver dell'approccio "shift-left"?

Varie modifiche a livello di mercato sono in fase di attuazione e stanno determinando nuovi comportamenti all'interno del settore. Alcune hanno avuto origine circa cinque anni fa, altre sono più recenti. Ma cosa intendo per "hanno avuto origine"? Intendo che il numero di persone che promuovevano i nuovi approcci era sufficiente per dimostrarne il successo, con credibilità tale perché anche altri li adottassero. Questi approcci hanno "attraversato l'abisso", nelle parole di Geoffrey Moore[2], e sono diventati praticabili per la community IT e il business nella loro definizione più ampia.

Le modifiche principali coinvolte nei fenomeni di "shift-left":

1. L'approccio BDD ha consentito a sviluppatori, utenti/analisti di business e tester di collaborare mediante business story. Il BDD è in una fase di adozione sempre più ampia, perché promuove una migliore collaborazione tra i team Agile.

2. Anche il concetto di continuous delivery (CD)[3] esiste da 5-10 anni e le sue radici affondano negli approcci di automazione altamente automatizzati a build e release di cui i grandi business online sono stati pionieri. Oggi è adottato dalla maggior parte delle aziende con una presenza online.
3. La CD ha sistematizzato e accelerato il processo di release attraverso l'automazione. Successivamente, però, ha messo in luce i ritardi nel deployment in produzione e nel cambiamento dell'infrastruttura precedentemente mascherati dalla lentezza dei processi di build, testing e release. DevOps è un cambiamento culturale e di mentalità, che prevede che gli sviluppatori collaborino più da vicino, in particolare, con il personale delle operations. In questo momento, quasi ogni giorno nuovi strumenti arrivano sul mercato e i vendor promuovono DevOps come "the next big thing". Si tratta di una situazione molto intensa e dinamica.
4. Il concetto di SMAC (Social, Mobile, Analytrics e Cloud) rappresenta un cambiamento epocale nel modo in cui le aziende gestiscono il cambiamento di sistemi e business nell'ambito mobile. La sperimentazione nel business, implementata come modifiche ai sistemi di produzione, viene monitorata a un livello di dettaglio. I "Big Data" acquisiti vengono elaborati e le decisioni di business adottate sulla base dei dati di analisi ottenuti.

La sperimentazione frequente con i sistemi di produzione consente l'innovazione di business "alla velocità del marketing". La sperimentazione è il fulcro di quello che sembra essere l'elemento più importante del primo decennio degli anni 2000: la "digital transformation". Sulla digital transformation, o "digital", si concentra oggi la maggior parte dell'attenzione (e delle risorse di bilancio). Gli esperti di marketing promettono un accesso migliore e più rapido ai consumatori attraverso più canali, per lo più mobile.

A questo riguardo potrebbe essere d'interesse il mio documento "Digital Transformation, Testing and Automation"[4], che descrive la rivoluzione digitale e propone alcune risposte.

Cosa significa l'approccio "shift-left" per i tester?

L'approccio "shift-left" implica che, ogni volta che è possibile fornire un feedback che aiuterà il team a comprendere, mettere in discussione e migliorare obiettivi, requisiti, progettazione o implementazione, quel feedback va comunicato. Questo comportamento diventa per molti tester (ma non per tutti) una seconda natura. Gli utenti, gli analisti di business, gli sviluppatori e l'intero team devono essere pronti a fornire e ricevere questo feedback. Una certa resistenza è possibile, ma l'obiettivo generale consiste solo e soltanto in un progetto migliore e più informato.

Qual è il ruolo del tester nel mondo del testing incentrato sull'approccio "shift-left"? Il modo più semplice per riassumerlo è "farsi coinvolgere precocemente", anzi il più presto possibile. Entrare nella discussione e collaborare su idee, requisiti e su tutte le fasi il cui risultato ha un impatto sul valore del prodotto finale del progetto. In parole povere, il tester mette in discussione le fonti di conoscenza, che si tratti di stakeholder, utenti, sviluppatori, business story, documenti o informazioni trasmesse.

L'approccio più comune consiste nel farlo "dando l'esempio". In tutte le fasi, questi esempi possono essere considerati come test. Potrebbero venire scartati rapidamente dopo l'uso, oppure venire codificati nell'automazione di test o in controlli manuali. Potrebbero essere utilizzati solo a livello tattico, per sottolineare eventuali carenze di ragionamento; oppure venire forniti agli sviluppatori, ad esempio, come idee o basi di partenza per i test eseguiti da loro. Potrebbero essere utilizzati anche come ausilio di coaching, per aiutare utenti o sviluppatori a individuare opportunità di miglioramento dei test.

I progetti software sono stati descritti come un processo di acquisizione di conoscenze[4]. Queste conoscenze vengono raccolte in tutto il corso del progetto e spesso evolvono nel tempo. L'obiettivo dell'approccio "shift-left" è garantire che conoscenze, in tutto il processo di messa in discussione e test, rimangano vicine alla relativa origine e, ove possibile, che siano attendibili prima di venire "congelate" nel codice.

L'approccio "shift-left" porta ancora più in là la filosofia test-first. Agile promuove da sempre la collaborazione e il feedback rapido; e l'approccio "shift-left" potrebbe essere considerato semplicemente come l'approccio definitivo al feedback rapido.

Se lo si adotta, l'approccio "shift-left" ha un effetto profondo sulla modalità di lavoro dei tester.

In che modo i tester possono applicare l'approccio "shift-left"?

Apparentemente, l'approccio "shift-left" non è solo una moda passeggera e sta per sbarcare nell'ambito del testing. Come influenzerà il tuo lavoro come membro di team di system test? Conserva la sua importanza se fai parte di un team Agile? Come dovresti comportarti?

Da qualche tempo sosteniamo l'approccio dell'approccio "shift-left" come nucleo della strategia di test in progetti Agile. In un contesto Agile, la strategia di test può essere vista come una serie di "interventi Agile". In tutti i progetti arrivano momenti critici in cui si presentano opportunità per raccogliere e presentare feedback. Il tester deve concentrarsi su questi momenti critici ed essere pronto a offrire il proprio contributo.

Ho presentato i concetti alla base di questo approccio in una recente webinar[6] e utilizzo il case study di un cliente per illustrare dove questi interventi potrebbero verificarsi. All'interno dei tuoi progetti, dovrai identificare i "momenti critici" specifici e individuare le scelte possibili per te e per il tuo team. Ad esempio: dovresti scrivere test unitari per gli sviluppatori, fornire loro esempi per iniziare o fare loro da coach perché migliorino le loro capacità di testing?

Il tuo ruolo sarà quasi certamente destinato a cambiare. Può essere che i tester non si limitino a considerare l'approccio "shift-left", ma che lo stiano realizzando effettivamente, e che siano destinati a diventare "cavalier serventi" di test degli sviluppatori. Questo probabilmente non è il risultato ideale per te, o per il progetto. Quello che ti suggeriamo è di identificare i momenti critici, proporre il tuo contributo e trovare un compromesso con il tuo team. Offrire leadership e orientamento per le attività di test, piuttosto che offrirti semplicemente volontario per assumerne la responsabilità. Se adotti questo approccio, sarà molto più facile dimostrare il tuo valore al team: team che non avrà bisogno di un numero così elevato di tester.

Sono un test lead/test manager. Come mi comporto?

Per i test manager o i test lead oggi potrebbe essere più difficile giustificare il proprio ruolo, se l'intento del management è ridurre il costo dei test mediante l'approccio "shift-left". Se l'azienda si sta muovendo in quella direzione, probabilmente dovrai prendere una decisione più a lungo termine per la tua carriera. Dove vorresti essere tra cinque anni? E tra sei mesi? Abbiamo identificato cinque ventagli di scelte praticabili per te.

1. **Offerta di capacità di test e assurance al business:** risalendo la catena alimentare verso gli stakeholder, il tuo ruolo potrebbe essere quello di fornire consulenza ai leader di business che desiderano assumere il controllo dei progetti IT. In qualità di agente indipendente, comprendi le problematiche di business e le comunichi ai progetti. Fungi da consulente e da persuasore della leadership di progetto, ne rivedi la performance e i risultati, ne interpreti i risultati, consigli gli stakeholder.
2. **Gestione delle conoscenze dei requisiti:** in questo ruolo assumi il controllo delle conoscenze necessarie per definire e costruire i sistemi. Utilizzando le tue competenze essenziali, garantisci chiarezza e precisione dei requisiti e degli esempi che illustrano le funzionalità in uso. Aiuti il business e gli sviluppatori a decidere quando i requisiti possono essere considerati attendibili al punto da consentire ragionevolmente la compilazione e il testing del software. Ti occupi della gestione dei requisiti, del glossario e del dizionario di utilizzo dei concetti di business e degli elementi di dati. Fornisci un servizio di analisi dell'impatto sul business.
3. **Funzione di TestMaster:** fornisci una funzione di assurance a team, progetti e stakeholder: un ruolo simile a quello del precedente punto 1, ma per ambienti più orientati all'Agile. Sei uno specialista di test e un esperto di assurance che mantiene i progetti Agile sulla giusta strada. Lavori a stretto contatto con i clienti on-site e con i responsabili di prodotto. Faciliti riconoscimento e reazione al rischio all'interno dei progetti, fungi da coach e da mentore per il team e ne gestisci le attività di testing; in alcuni casi, te ne occupi in prima persona.

4. **Funzione di DevOpsMaster:** gestisci il flusso delle informazioni critiche da e verso i processi DevOps (processi di compilazione, test e deployment automatizzati). Il flusso delle informazioni è essenziale. Potresti anche definire e supervisionare i processi utilizzati per gestire i flussi che rendono possibile il controllo del cambiamento, il testing e la delivery.
5. **Gestione di team in outsourcing/offshore:** in questo caso rinunci alla presenza di un tuo team di test in loco e gestisci il trasferimento del lavoro a fornitori in outsourcing o offshore. Sei esperto del flusso delle informazioni e gestisci il rapporto con il team di test in outsourcing, ne monitori la performance e ne garantisci l'output.

Se ancora non ti sei impegnato in questo approccio "shift-left", ti consigliamo di dare un'occhiata all'interno e all'esterno del tuo team, e di pensare a come il tuo ruolo potrebbe cambiare nel prossimo futuro. Questo cambiamento è inevitabile, ma è importante che tu abbia voce in capitolo sulla sua evoluzione. Buona fortuna!

L'autore

Paul Gerrard è un consulente, formatore, autore, webmaster, sviluppatore, tester, relatore, allenatore di canottaggio ed editore. Ha svolto incarichi di consulenza su tutti gli aspetti del test e del QA software, specializzandosi nella test assurance. Partecipa con presentazioni e tutorial a conferenze sul testing in Europa, USA, Australia, Sud Africa e, occasionalmente, ha anche vinto dei premi grazie a questi interventi.

Ha studiato presso l'università di Oxford e l'Imperial College di Londra; nel 2010, ha vinto l'Eurostar European Testing excellence Award e, nel 2013, l'European Software Testing Awards (TESTA) Lifetime Achievement Award.

Nel 2002, Paul ha scritto "Risk-Based E-Business Testing" insieme a Neil Thompson. Ha scritto "The Tester's Pocketbook" nel 2009. È stato co-autore di "The Business Story Pocketbook" con Susan Windsor nel 2011 e ha pubblicato "Lean Python" nel 2014.

Sempre nel 2014 Paul è stato Programme Chair della Conferenza EuroSTAR di Dublino.

È Principal di Gerrard Consulting Limited, direttore di TestOpera Limited e host del Test Management Forum.

E-mail: paul@gerrardconsulting.com
Twitter: @paul_gerrard
Web: gerrardconsulting.com

Per ulteriori informazioni, visita **Sviluppo e test** con CA Technologies.

Entra in contatto con CA Technologies all'indirizzo ca.com/it

CA Technologies (NASDAQ: CA) crea software che promuove l'innovazione all'interno delle aziende, consentendo loro di cogliere le opportunità offerte dall'economia delle applicazioni. Il software rappresenta il cuore di qualsiasi business, in ogni settore. Dalla pianificazione allo sviluppo, fino alla gestione e alla sicurezza, CA Technologies lavora con le aziende di tutto il mondo per cambiare il nostro modo di vivere, interagire e comunicare, in ambienti mobile, cloud pubblici e privati, distribuiti e mainframe. Per ulteriori informazioni, visita il sito **ca.com/it**.

Riferimenti

1. "The W-Model", http://blog.gerrardconsulting.com/?q=node/531
2. "Crossing the Chasm" e altri titoli di Geoffrey A Moore, http://www.chasminstitute.com/
3. Definizione di continuous delivery, Martin Fowler, http://martinfowler.com/bliki/ContinuousDelivery.html
4. "Digital Transformation, Testing and Automation", blog di Paul Gerrard, http://blog.gerrardconsulting.com/?q=node/660
5. "The Laws of Software Process", Philip G Armour.
6. Webinar: "Agile Test Strategy", Paul Gerrard, http://blog.gerrardconsulting.com/?q=node/627

Copyright © 2016 CA Technologies. Tutti i diritti riservati. Tutti i marchi, i nomi commerciali, i marchi di servizio e i loghi citati nel presente documento sono di proprietà delle rispettive società. Il presente documento non contiene alcuna garanzia e ha esclusivamente scopo informativo.

CS200-181168_0416