WHITE PAPER | MARZO 2016

DevOps e tester

In una serie di articoli Paul Gerrard, guru del testing e consulente, discute una serie di argomenti collegati all'argomento testing. In questo caso, tratta dell'adozione di DevOps dal punto di vista dei test e dei tester. DevOps è parte di un approccio globale che le aziende adottano per erogare con frequenza software di qualità elevata. Il risultato più evidente di una corretta implementazione di DevOps è la riduzione del tempo necessario perché le modifiche software passino dalla fase concettuale alla produzione.

Paul Gerrard
Gerrard Consulting

Sponsorizzato da
ca technologies

Cosa significa DevOps per i tester?

Contesto

In questo articolo desidero concentrarmi sull'adozione di DevOps dal punto di vista dei test e dei tester. Il movimento (in mancanza di una denominazione migliore) DevOps avanza rapidamente. Come molti altri progressi compiuti dal settore, l'adozione è più rapida della definizione del movimento. DevOps non è ancora definito con precisione e le sfumature culturali, la capacità emergente delle nuove tecnologie e la gamma dei case study (per lo più) di successo implicano che le questioni qui citate siano ancora ampiamente oggetto di dibattito.[1]

A seconda dell'interlocutore, DevOps può essere la soluzione a un problema, o un obiettivo in sé. Per alcuni business l'obiettivo è il passaggio al digitale, e DevOps è parte di un approccio complessivo alla delivery frequente di software di qualità elevata. Questo è il contesto da cui partirò in questo documento. Tuttavia, nell'ambito del marketing delle tecnologie e dei servizi collegati a DevOps, questo obiettivo può risultare oscurato. La sfida rappresentata dal cambiamento culturale (o più concretamente, comportamentale) necessario per il successo viene spesso sottovalutata.

L'altro presupposto sul quale mi baserò è che per i tester coinvolti e influenzati da DevOps l'intero concetto sia nuovo. Questo articolo vuole essere un'introduzione a DevOps per questi tester, e un'illustrazione del suo impatto sulle pratiche di test. Spero comunque che risulterà utile anche a chi è già esperto di DevOps; inoltre, coloro che non sono tester potranno quantomeno acquisirne la prospettiva.

Per i neofiti: Cos'è DevOps?

Semplificando, DevOps è un'etichetta associata al concetto di una più stretta collaborazione tra team di systems operations e di sviluppo. Nella cosiddetta pipeline di delivery, dalla commit del codice sorgente al funzionamento in produzione, gli sviluppatori gestiscono e automatizzano alcune delle attività delle operations. Le operations hanno maggiore visibilità e una certa influenza sulle attività degli sviluppatori. La motivazione di questo consiste principalmente nel velocizzare il deployment e l'implementazione del software. Avvicinare Dev e Ops, costituendo a tutti gli effetti un team Agile, determina la realizzazione di quelle che potrebbero essere definite "Agile operations".

Il risultato più evidente di una corretta implementazione di DevOps è la riduzione del tempo necessario perché le modifiche software passino dalla fase concettuale alle operations in produzione. Quando uno sviluppatore dichiara che una modifica software è "completa", la transizione verso l'utilizzo in produzione avviene con l'ausilio di un'automazione pervasiva. Strumenti e processi automatizzati sono utilizzati nella configurazione di sistema, nel processo di build, nel testing, nel deployment negli ambienti di test, staging e produzione, nel monitoraggio post-implementazione, nella valutazione e nelle operations.

DevOps, insomma, è solo una questione di strumenti?

A un certo livello, l'obiettivo di DevOps consiste nell'eliminare i colli di bottiglia nella pipeline di delivery, attraverso l'automazione. Ma l'automazione dei processi in fasi ha ancora bisogno di governance. La maggioranza dei processi automatizzati non è veramente autonoma: non è in grado di completare i propri compiti senza un intervento umano, che gestisca la manutenzione o le eccezioni. Un processo DevOps completamente automatizzato non ha senso se non si considera il fattore umano. Anche se gli strumenti svolgono molta parte del lavoro pesante, sono le persone a gestire il processo che fa funzionare tutto... oppure no.

Allora, DevOps è solo questione di risorse Dev e Ops che lavorano a più stretto contatto, con l'ausilio di strumenti?

No, nemmeno questo. Gli handoff tra i processi automatizzati coinvolgono spesso altri processi, di solito un qualche tipo di test. I test automatici devono essere creati da sviluppatori e tester. Il risultato di questi test si concentra sulla fornitura di informazioni sufficienti perché gli altri processi o, altrettanto spesso, le persone, eseguano la transizione tra i diversi stadi della pipeline. Tester e sviluppatori che eseguono i test costituiscono la garanzia della riuscita del processo DevOps in modo sicuro e affidabile.

"Non capisco, ma allora cos'è davvero DevOps?". Va detto che si tratta di una disciplina emergente e in evoluzione. Questa domanda viene posta e discussa in dettaglio in un ottimo post disponibile qui.[1] Quel dibattito ha avuto luogo solo poche settimane prima della stesura di questo articolo. Come puoi vedere, quindi, la definizione di DevOps non è stata ancora stabilita. Forse non lo sarà mai.

Cosa significa questo per i tester? Significa che non esiste ancora un unico metodo consolidato e che il tuo ruolo in un contesto DevOps in evoluzione (e in ogni contesto in evoluzione) non è ancora fissato. Puoi contribuire essenzialmente in due modi:

1. Devi prestare attenzione ai punti dolenti e lavorare per ridurne l'impatto.
2. Devi identificare le opportunità e gli interventi che aggiungeranno valore al processo DevOps.

Se c'è un mantra che descrive al meglio il driver a DevOps è "se qualcosa è difficile, continua a provare". Potrebbe suonare come un cliché, ma lo userò come contesto per l'implementazione e il miglioramento delle pratiche di test DevOps.

Se qualcosa è difficile, continua a provare (sempre più spesso)

La difficoltà o la frustrazione che proviamo svolgendo un determinato lavoro ci influenza negativamente. Se un'attività non ci è gradita, tendiamo a rimandarla. Quando finalmente ci decidiamo a svolgerla, risulta ancora meno piacevole. Questo vale per le visite dal dentista, la pulizia del garage, l'integrazione di software, i test e così via. Secondo la nostra esperienza è normale che, meno spesso svolgiamo queste attività, più frustranti risultano quando effettivamente ce ne occupiamo. Martin Fowler suggerisce tre ragioni per cui l'esecuzione frequente, o addirittura costante, di alcune attività le rende meno sgradevoli.[2]

La prima: attività più ampie e complesse sono difficili da pianificare, gestire e controllare; suddividerle le rende più semplici, meno rischiose e, se qualcosa dovesse andare storto, più facili da annullare. La seconda: da molte attività (e il testing è un esempio lampante) deriva un determinato feedback. Quel feedback, se ricevuto spesso e precocemente, implica la capacità di affrontare i problemi rapidamente e in modo certo prima di sprecare altro tempo. Infine, se svolgiamo un'attività più spesso, diventiamo più bravi nel farla: impariamo come svolgerla in modo efficiente. Possiamo anche arrivare a individuare opportunità per automatizzarla in qualche modo.

Dal punto di vista del tester, questo mantra costringe a prendere molto più seriamente il concetto di automazione nel processo di test. Se sono previsti (in genere tra le varie fasi automatizzate del processo DevOps), gli interventi manuali verranno visti come punti dolenti: colli di bottiglia, cause di ritardi e aspetti potenzialmente meno affidabili e più soggetti a errori del processo. I test manuali sono decisamente frustranti. C'è chi ama i test esplorativi, ritenendo che solo un essere umano sia in grado di trovare quei noiosi bug che all'automazione sfuggirebbero, che il tester sia l'unica persona in grado di evitare il peggio.

Per un tester l'idea di affidarsi agli sviluppatori e all'automazione perché svolgano correttamente l'attività di test potrebbe essere ben poco piacevole. Se qualcosa è difficile, continua a provare (sempre più spesso).

Test, automazione e fiducia

C'è molto dibattito, ad esempio, intorno al significato di controllo e testing[3], e all'affidabilità che possiamo attribuire a tester, controlli e automazione.[4, 5]

Non sto dicendo che dobbiamo affidarci completamente ai controlli automatici: sicuramente dobbiamo essere più sofisticati di così. Ma altrettanto sicuramente possiamo, agli scopi di questo articolo, separare almeno i test e l'attività di esecuzione dei test in quattro componenti.

1. Controlli che possono essere automatizzati dagli sviluppatori come parte dei loro processi di integrazione continua e di check-in a livello di componente.
2. Controlli che possono essere automatizzati (in genere dai tester di sistema) per eseguire transazioni a livello di API, collegamento o end-to-end.
3. Test in grado di eseguire controlli di compatibilità per dimostrare la compatibilità tra più browser, sistemi operativi e piattaforme.
4. Test che possono essere eseguiti solo da un operatore umano.

In questo articolo posso soltanto fornire alcuni suggerimenti su come operare queste distinzioni: ogni ambiente, ovviamente, è diverso. La domanda più pertinente in questa sede è come può il tester smettere di occuparsi dei controlli manuali delle ultime fasi del ciclo. Ho già parlato dell'eliminazione di questo tipo di controllo.[6] Essa richiede sforzo proattivo e fiducia.

Questi saranno l'obiettivo principale dei tuoi sforzi:

1. Dove possibile, i controlli manuali che potrebbero essere eseguiti a livello di componente devono essere affidati agli sviluppatori. Come tester, puoi suggerire questi test in una sessione di whiteboard o di pairing. Potresti doverli scrivere e includerli nel regime di integrazione continua.
2. I test end-to-end o dell'interfaccia utente potrebbero richiedere l'automazione. Questi devono essere ridotti al minimo, in quanto tendono a essere lenti da eseguire, fragili e a richiedere spesso manutenzione. Considera se siano necessari a ogni check-in del codice, o se potrebbero essere riservati a release più ampie e meno frequenti.
3. Quali test esclusivamente manuali potrebbero essere eseguiti su componenti non ancora integrati in una release candidate? I test manuali possono essere eseguiti in sessioni di pairing con gli sviluppatori? Esistono alternative a questi test? La creazione di story-board e la prototipazione in stile BDD potrebbero essere d'aiuto? Sarebbe possibile eseguire i controlli UI su mock-up o wire-frame?
4. Quali controlli devono essere eseguiti solo una volta, manualmente, in contrasto con quelli da mantenere a fini di regressione, e che costituiscono altrettanti candidati per l'automazione?

Più sopra ho citato il concetto di fiducia. Si può vederlo in altro modo valutando come un sistema potrebbe essere testato in modo affidabile in assenza totale di test manuali nelle ultime fasi. Immagina un ambiente in cui tutta l'attività di test viene compiuta da strumenti. Le tue preoccupazioni si concentrerebbero sulle incertezze circa la capacità degli sviluppatori di svolgere un buon lavoro di test? Adottare l'approccio "shift-left" per la visione del testing (come suggerito nel mio articolo precedente) dovrebbe eliminare questi dubbi. Se, come tester, fungi più da apripista per identificare i rischi e valutarli, selezionare i test e garantire che vengano integrati nello sviluppo e nell'automazione, le tue preoccupazioni potrebbero essere ridimensionate.

Certo, devi smettere di considerarti il solo responsabile della qualità, l'ultimo bastione di difesa, l'unico a cui sta a cuore questo aspetto. Devi invece pensare alla tua figura più come a quella di un visionario, un soggetto che identifica i rischi e li gestisce, un apripista, un facilitatore e un coach/mentore.

Pratica, monitoraggio e miglioramento

Malgrado tutte le buone intenzioni di ridurre o eliminare la dipendenza dai controlli manuali nelle ultime fasi del ciclo, è inevitabile che qualche bug non venga rilevato. I problemi sorgono quando il software viene rilasciato in produzione. Una delle discipline fondamentali di DevOps dal punto di vista delle operations è un livello più profondo di monitoraggio.

Il monitoraggio a ogni livello, dai componenti e semplici transazioni nelle applicazioni, fino all'integrazione e alla messaggistica e, naturalmente, alla stessa infrastruttura. Uno degli obiettivi del monitoraggio consiste nel raccogliere le segnalazioni di errori prima che gli utenti ne sperimentino l'impatto direttamente. Questo è l'obiettivo finale: ed è piuttosto ambizioso.

Quando si verificano problemi in produzione, occorre utilizzare l'analisi derivata dal monitoraggio non solo per individuare la causa e risolverla, ma anche per perfezionare il processo di test, automatico o manuale, in modo da ridurre la probabilità di problemi simili per il futuro. Il ruolo del test e dell'analisi all'interno dell'intero processo di pipeline è stato presentato e discusso qui.[7]

I test automatizzati all'interno del processo DevOps potrebbero ricadere nella definizione di "monitoraggio". Si potrebbe dire che, associato al monitoraggio in produzione, il monitoraggio in tutto il processo DevOps e verso la fase di produzione amplia il campo di applicazione dei test. DevOps, quindi, non riduce il ruolo dei tester.

Conclusioni

Recentemente mi è stato chiesto in quali casi sarebbe meglio evitare di implementare DevOps all'interno di un'azienda. È una buona domanda, ma credo che dietro di essa si nasconda il dubbio che, se DevOps non è una moda passeggera, forse i tester dovrebbero iniziare a prenderlo in considerazione. La mia risposta è semplice.

Perché non dovresti desiderare che sviluppatori e operations dialoghino tra loro? Build e deployment più affidabili in test e in produzione? Una tecnologia ottimizzata, che supporti pipeline più precise, efficienti e informative? DevOps è un ottimo concetto, ma non sempre facile da realizzare. Inutile dire che richiede un cambiamento culturale, il che può essere complesso.

Al tester DevOps offre una maggiore influenza nelle prime fasi dei progetti, lo costringe a pensare più seriamente all'automazione in fase di test, di provisioning delle informazioni e di processo decisionale. I tester devono abbracciare DevOps perché fornisce loro l'opportunità di essere proattivi e di acquisire maggiore autorità e rispetto all'interno dei team di progetto.

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. "What is DevOps", The Agile Admin, http://theagileadmin.com/what-is-devops/
2. "Frequency Reduces Difficulty", Martin Fowler, http://martinfowler.com/bliki/FrequencyReducesDifficulty.html
3. "Testing and Checking Refined", James Bach, Michael Bolton, http://www.satisfice.com/blog/archives/856
4. "A New Model for Testing", Paul Gerrard, http://dev.sp.qa/download/newModel
5. "The New Model and Testing v Checking", Paul Gerrard, http://blog.gerrardconsulting.com/?q=node/659
6. "How to Eliminate Manual Feature Checking", webinar di Paul Gerrard, http://blog.gerrardconsulting.com/?q=node/622
7. "Thinking Big: Introducting Test Analytics", Paul Gerrard, http://blog.gerrardconsulting.com/?q=node/630

CS200-181159_0316