LIVRE BLANC | SEPTEMBRE 2016

Tests, analyse et prise de décision

Dans cette série de livres blancs, Paul Gerrard, directeur de Gerrard Consulting Limited, aborde plusieurs des thèmes clés liés aux tests IT. Dans ce document, il explore les outils d'analyse et de création de rapports de test qui permettent de soutenir la prise de décision dans l'entreprise. Comment quantifier les tests ? S'il est assez facile de calculer le coût des tests, à partir de quel volume de tests pouvez-vous estimer que cela suffit ? Et quelle valeur ont les tests ? Voilà autant de questions difficiles auxquelles les testeurs doivent pouvoir répondre.

L'analyse des tests couvre l'intégralité du cycle de vie des applications, de l'idée à la production, en passant par le développement et le retrait. Surtout, elle va bien au-delà de la simple exploitation des résultats générés par des outils de gestion de tests.

Paul Gerrard
Gerrard Consulting

Sponsorisé par
ca technologies

Introduction

Au niveau le plus fondamental, l'objectif des tests est de collecter des informations afin de connaître certains aspects d'un système et de permettre la prise de décisions sur la base de résultats d'un ou de plusieurs tests.[1] Les tests permettent de générer les informations les plus précieuses dont ont besoin les développeurs (pour corriger les défauts), les chefs de projets (pour comprendre et gérer la progression des projets) et les parties prenantes (pour disposer d'informations à jour et être rassurées). À cet égard, ils sont tout puissants dans la mesure où ils constituent la seule source de connaissance sur l'état de réalisation des projets système.

Dans la sphère numérique ou mobile, les entreprises mettent en production des applications qui capturent des informations sur leurs utilisateurs et leurs modèles d'utilisation. Les applications actuelles collectent des données en temps réel et les renvoient en temps quasi réel au fournisseur. Ce dernier les analyse alors afin d'identifier des tendances, des modèles comportementaux, des préférences utilisateur, des opportunités d'améliorations ou de nouveaux marchés. Les applications intègrent des instruments dont le rôle est de collecter des informations qui serviront à la prise de décision.

Présentation du concept d'analyse des tests

Dans l'article « Thinking Big: Introducing Test Analytics »[2], je recommande de considérer les tests précédant la mise en production et les activités d'analyse et de supervision suivant la mise en production comme deux activités appartenant au même cycle. J'utilise le terme « Analyse des tests » pour désigner une discipline qui couvre l'intégralité du cycle de vie d'une application, de l'idée au retrait, en passant par son développement et sa production. Pour en savoir plus sur les tests et l'analyse des applications mobiles, vous pouvez consulter le document « The Mobile Analytics Playbook ».[3]

Je définis l'analyse des tests comme

> *la capture, l'intégration et l'analyse des données de supervision des tests et de la production, en vue d'alimenter la prise de décision au niveau du métier et du développement logiciel*

En règle générale, les testeurs voient dans les résultats des outils de gestion des tests la principale source d'informations pour le reporting, mais le point de vue de ces résultats est bien trop limité. Certes, il est intéressant de connaître le nombre de plans de test et de définitions des besoins couverts (quel qu'en soit le sens), de rapports d'incidents ou de bugs créés, triés, corrigés ou ignorés, résumés, et de séries chronologiques… Mais ces données sont restrictives.

Pratiques modernes : opportunités de test

Dans mes précédents articles de cette série, j'ai apporté un peu de lumière sur la manière dont nous pouvons améliorer notre création de rapports et nos analyses. L'ensemble des réflexions derrière le mouvement de déplacement plus en amont (« shift left »), les modèles de test, DevOps et l'automatisation, ainsi que les disciplines de tests émergentes qu'ils encouragent, nous obligent à réviser notre définition de la création de rapports de tests. L'objectif fondamental des tests, à savoir collecter et analyser des données afin de permettre une prise de décision informée, reste inchangé. En revanche, les opportunités d'améliorer la façon dont nous documentons les tests sont nombreuses :

1. Le principe de déplacement des tests vers l'amont (autrement dit vers la gauche) dans le cycle de vie des applications vise à réduire, voire éliminer, de façon précoce tout problème de compréhension des besoins. Cela permet d'éviter que des défauts, des modifications de la définition des besoins et des situations d'urgence viennent perturber le reporting.
2. Par ailleurs, le passage à une automatisation généralisée dans le cadre des approches DevOps permet de générer automatiquement une grande partie des données dont nous avons besoin. La capture et l'analyse des résultats ne sont plus manuelles, et la création de rapports est quasiment instantanée.

3. Un nouveau modèle de test[4] place la modélisation au cœur de la conception des tests. En utilisant des modèles pertinents, les testeurs peuvent obtenir une mesure plus pertinente de la couverture.
4. Des données d'analyse des tests peuvent également être capturées au cours de sessions exploratoires. Il est possible de créer des modèles de système et d'usage avant les sessions de tests, et de capturer les données de couverture des tests pendant les sessions, à la volée.
5. Les tests réalisés pendant la phase de production même sont de plus en plus courants. Pourquoi ? Parce que personne n'est capable de tester tous les types d'appareils mobiles (l'écosystème Android compte plus de 24 000 appareils).[5] Dans un tel contexte, la valeur d'une analyse de supervision de la production est évidente.
6. Les produits de supervision, de journalisation, de génération d'alertes et d'analyse sont de plus en plus répandus. Les applications en intègrent, comme les infrastructures. Les scénarios que les testeurs n'ont pas pu tester dans le laboratoire peuvent désormais être supervisés (testés) sur le terrain.
7. Certaines entreprises renoncent aux procédures bureaucratiques de gestion des incidents. Les défauts sont corrigés lorsqu'ils sont trouvés.
8. Il n'y a plus d'incident. L'outil de contrôle du code source, les tests unitaires automatisés qui favorisent et protègent les changements, fournissent les preuves et les nouvelles visualisations de changement et d'impact. La visualisation du code source « History of Python » vous montre un exemple de ce qui est possible.[6]

Tous ces nouveaux outils, ces pratiques, techniques et visualisations font évoluer la nature des tests. Nous avons une réelle opportunité d'améliorer notre produit de base : cette opportunité réside dans les informations que nous fournissons aux parties prenantes. Mais comment les tests peuvent-ils aider la prise de décision ?

Tests et prise de décision

Au cours de ma carrière, j'ai eu l'occasion de travailler sur de nombreux projets de tests, de grande et de petite envergure. Quel que soit le projet, la phase de test doit inévitablement aboutir à une prise de décision : que faire ensuite ? Passer à la phase suivante, produire, publier ? Repousser les délais ? Faut-il interrompre temporairement le travail, pour réviser nos objectifs ? Ou laisser tout tomber et mettre fin au projet ?

Aussi critiques que ces décisions paraissent, elles sont prises à de nombreux niveaux. Imaginons qu'un testeur détecte un problème et en discute avec un utilisateur et le développeur. Les décisions doivent être prises rapidement : est-ce un bogue ? Peut-il être corrigé ? Peut-il être corrigé rapidement ? Quel est l'impact du changement ? Y a-t-il une solution viable ? Est-ce que cela vaut la peine de corriger ce bogue ? A-t-il un impact sur l'utilisateur ?

Les tests soutiennent la prise de décision à tous les niveaux. Les parties prenantes aux tests (dirigeants, clients, utilisateurs, chefs de projets, équipe de production IT et développeurs) pouvant avoir de nombreux rôles, les informations de test dont elles ont besoin varient en fonction de leurs perspectives et de leur besoin de prendre des décisions. Il n'y a pas de formule magique pour décider des informations qui sont nécessaires : il convient de demander directement à nos parties prenantes quelles données elles trouvent les plus utiles, précieuses et rentables.

Mais il y a un problème. Comment quantifier les tests ? S'il est assez facile de calculer les coûts des tests, à partir de quel volume de tests pouvez-vous estimer que cela suffit ? Et quelle valeur ont les tests ? Voilà autant de questions difficiles auxquelles les testeurs doivent pouvoir répondre.

Grand amateur de physique, j'ai choisi de reprendre à mon compte quelques-uns de ses principes pour mon argumentaire.

Tests fonctionnels évolutifs

Lorsque nous testons la fonctionnalité de composants plus haut dans l'architecture, notamment au niveau de l'intégrateur et de l'application, nous pouvons être amenés à simuler des milliers ou des millions d'appareils sur le terrain. Le nombre de combinaisons et de permutations est difficilement calculable ou prévisible. Nos simulations génèrent en boucle les scénarios à tester, enregistrent les résultats et répètent potentiellement les simulations pour permettre une analyse ultérieure.

Les composants de plus haut niveau doivent pouvoir être testés. Pour ce faire, nous aurons besoin d'équipements tels que des gestionnaires d'exception, des utilitaires qui injectent des données, capturent et reproduisent ou répètent des scénarios. C'est ce que Cem Kaner appelle une automatisation des tests en volume (« High Volume Automated Testing »[7]), un bon point de départ.

Ces techniques peuvent également être appelées tests Big Data. Voici ce dont nous avons besoin pour mettre ces techniques en application :

- Trouver des données qui répondent à notre objectif
- Générer, marquer, modifier et « planter » (« seed ») des données afin de pouvoir retracer leur utilisation
- Disposer d'outils permettant de superviser l'utilisation des données marquées et de rapprocher les données de collecte, de stockage, d'utilisation et d'élimination
- Obtenir de nouveaux outils de visualisation des tests pour prendre en charge des fonctions de diagnostic et de débogage

Il s'agit d'un travail conséquent : que les tests spécifiques soient importants ou non, nous passerons beaucoup de temps à traiter des résultats, des visualisations et des prises de décision à grande échelle.

Le principe d'incertitude des tests

Si vous vous êtes déjà essayé à la planification d'un volume de tests qui vous semble « approprié », en fixant une date de fin, vous vous êtes sans doute rendu compte de la difficulté d'une telle tâche. Comment définir un planning d'exécution fiable quand vous ne connaissez pas sa variable la plus importante, à savoir le nombre de tests qui seront bloqués et le nombre de tests qu'il faudra renouveler ? Cela dépend du nombre de bogues que vous rencontrerez, et du niveau de difficulté lié à leur correction et à leur test. Vous serez incapable d'évaluer ces données avant d'avoir accompli une grande partie de vos efforts de tests, de correction et de nouveaux tests. C'est là un autre paradoxe des tests.

J'ai choisi de décrire le défi lié à la prédiction et à la planification des tests avec le principe suivant, que j'appellerai principe d'incertitude des tests :

- Il est possible de prédire le statut d'un test, mais pas la date à laquelle il sera achevé.
- Il est possible de prédire la date de fin d'un test, mais pas son statut à cette date.

Il est possible de définir un critère de sortie ou de fin (par exemple, quand tous les tests auront été exécutés avec succès), mais il n'est jamais possible de déterminer avec certitude la date à laquelle ce critère sera atteint. Combien de fois avez-vous défini des critères de sortie sans réussir à les tenir ? Il convient de traiter les critères de sortie comme des hypothèses de planification : si nous n'atteignons pas les critères, alors cela signifie que nos hypothèses et notre plan sont incorrects, et que nous devons replanifier ou redéfinir le périmètre du projet.

Nous pouvons définir une période fixe de test (« timebox ») pour garantir une date de fin, mais comment déterminer quel volume de tests nous pourrons intégrer dans cette période, et si cela suffira ?

La théorie de la relativité des tests

La valeur d'un test peut être affectée par de multiples facteurs. Un seul plan de test peut couvrir cinq lignes de code dans le test unitaire d'un composant, quand un autre en couvrira cinq millions dans un grand système. Évidemment, le deuxième aura plus de valeur. Mais qui peut dire quelle est cette valeur ?

La question doit être moins ambitieuse : le testeur peut avoir son opinion, mais il doit s'appuyer sur les parties prenantes pour évaluer la valeur des plans de test. À défaut d'attribuer une valeur à un test, les parties prenantes sont généralement en mesure d'identifier le test qui a le plus de pertinence pour elles. Il ne s'agit pas de définir la valeur absolue d'un test, mais de discerner sa valeur relative par rapport à un autre : lequel a le plus de valeur par rapport à l'autre ? Cela pourrait sembler être un inconvénient, mais ça ne l'est vraiment pas.

Le plus grand défi lié à la planification des tests réside dans la définition de leur périmètre. Comme nous ne pouvons pas tout tester, nous devons définir des priorités. En nous appuyant sur la valeur relative des tests, nous devrions être en mesure de sélectionner les plus pertinents d'entre eux et de mettre de côté les moins pertinents, dans un premier temps du moins.

Seules les parties prenantes pour lesquelles nous réalisons les tests peuvent juger de leur valeur.

Mais qu'en est-il des risques de défaillance ? La valeur d'un test n'est-elle pas conditionnée par les risques du mode de défaillance que le test a été conçu pour couvrir ? Il peut en être ainsi : l'évaluation d'un risque (quantifié ou non) est un facteur de valeur des tests. Mais il reste un problème.

Imaginons un test jugé comme pertinent. Un second test similaire à ce premier test aura-t-il systématiquement la même valeur ? Non. Pour en décider, nous avons besoin de faire appel à la théorie quantique.

La théorie quantique des tests

Lorsque nous réalisons un test, nous obtenons des résultats que nous comparons à des résultats attendus.

- Si les résultats obtenus correspondent aux résultats attendus, alors ils répondent de manière incrémentielle à un besoin ou une exigence utilisateur. Le test accroît de manière incrémentielle notre connaissance et notre confiance.
- Si les résultats obtenus ne correspondent pas aux résultats attendus, alors ils ne répondent pas, de manière incrémentielle, à un besoin ou une exigence utilisateur. Le test accroît de manière incrémentielle notre connaissance, mais il réduit notre confiance (jusqu'à ce que le système soit corrigé et retesté avec succès).

Chaque test génère un quantum discret de preuve : oui/non, vrai/faux, 0/1. Une fois tous les tests exécutés, nous agrégeons ces quanta de preuve pour obtenir une vue plus globale de la situation (voir le principe de boucle application de test - interprétation de résultat de l'article « A New Model for Testing »[8]).

Lorsque nous exécutons un test, celui-ci n'a de valeur que s'il nous apprend quelque chose que nous ne savions pas déjà. Si nous apprenons peu, voire rien, alors le test a une valeur faible ou nulle. Un test qui a de la valeur est important du fait qu'il renforce notre connaissance sur le système testé. Pour déterminer si deux tests valent mieux qu'un seul, nous devons évaluer la valeur potentielle du deuxième test et son importance.

Le deuxième test peut avoir de la valeur, mais son importance et sa valeur réelle peuvent se révéler faibles s'il intervient après un test quasi équivalent. L'importance d'un test est directement proportionnelle à l'augmentation incrémentielle de couverture qu'il fournit.

Le testeur, ou du moins la personne qui a conçu le modèle de test, est le meilleur juge de l'importance d'un test. Pour que la mesure de la couverture ait du sens, elle doit toujours être mesurée par rapport à un modèle de test.

Résumé

Que pouvons-nous conclure de cette discussion plutôt académique ?

DevOps, livraison continue, déplacement plus en amont des tests, toutes ces approches émergentes offrent une belle opportunité pour réduire les problèmes liés à des définitions des besoins de mauvaise qualité, intégrer les tests plus tôt dans le cycle de vie et localiser et résoudre les problèmes plus facilement. Toutefois, l'automatisation accrue facilite également la collecte de données sur les résultats des tests. Comment pouvons-nous tirer parti de cette opportunité ?

En créant des modèles de test qui sont pertinents pour nos parties prenantes, afin qu'elles reconnaissent la valeur des tests que nous leur proposons. En tant que testeurs et modélisateurs, nous sommes en mesure d'évaluer l'importance de ces tests (la couverture) par rapport aux modèles eux-mêmes, ce qui nous permet d'obtenir des tests de valeur tout en minimisant les duplicats.

Cet examen de la valeur et de l'importance des tests nous aide également à mieux juger de la valeur de l'automatisation de nos tests. J'ai souvent vu des difficultés survenir sur des projets d'automatisation de tests, lorsque l'automatisation portait sur un système complet. Lorsque les tests étaient exécutés comme des tests fonctionnels, ils avaient de la valeur, mais lorsqu'ils étaient ensuite exécutés de manière répétée, dans les phases de régression, alors leur importance (et donc leur valeur) se réduisait considérablement.

L'objectif de ce que nous devrions peut-être plutôt appeler « test d'anti-régression »[9] consiste à détecter un comportement indésirable après des modifications. Supposons que seuls 10 % des tests système suffisent à déclencher une alarme. Les autres 90 % couvrent le même aspect et ne sont donc pas importants.

Cet article devrait vous aider à entamer des discussions mieux informées lors d'une prise de décision et à déterminer le volume de tests suffisant, ainsi que la valeur de votre travail de testeur. Après tout, la théorie de la relativité et la théorie quantique pourraient vous être bien utiles.

À propos de l'auteur

Paul Gerrard est consultant, enseignant, auteur, webmaster, développeur, testeur, conférencier, entraîneur d'aviron et éditeur. Il a assumé de nombreuses missions de conseils concernant tous les aspects des phases de test et d'assurance qualité logiciels, et s'est spécialisé dans l'assurance qualité des tests. Il a animé des présentations et des didacticiels lors de nombreuses conférences sur le thème des tests, en Europe, aux États-Unis, en Australie et en Afrique du Sud, et a été récompensé à plusieurs reprises.

Diplômé des universités d'Oxford et Imperial College London, Paul a été lauréat en 2010 du prix d'excellence Eurostar European Testing, et en 2013 du prix The European Software Testing Awards (TESTA) Lifetime Achievement.

Il a écrit, en collaboration avec Neil Thompson, le livre « Risk-Based E-Business Testing », publié en 2002. En 2009, il a écrit « The Tester's Pocketbook ». En 2011, il a coécrit « The Business Story Pocketbook » avec Susan Windsor et a rédigé « Lean Python » en 2014.

Cette année-là, Paul a été également président de programme à la conférence EuroSTAR de Dublin.

Il est directeur de Gerrard Consulting Limited, directeur de TestOpera Limited et hôte du Test Management Forum.

Courriel : paul@gerrardconsulting.com
Twitter : @paul_gerrard
Site Internet : gerrardconsulting.com

Pour plus d'informations, rendez-vous sur la page **Développement et test** de CA Technologies.

Restez connecté à CA Technologies sur ca.com/fr

CA Technologies (NASDAQ : CA) fournit les logiciels qui aident les entreprises à opérer leur transformation numérique. Dans tous les secteurs, les modèles économiques des entreprises sont redéfinis par les applications. Partout, une application sert d'interface entre une entreprise et un utilisateur. CA Technologies aide ces entreprises à saisir les opportunités créées par cette révolution numérique et à naviguer dans « l'Économie des applications ». Grâce à ses logiciels pour planifier, développer, gérer les performances et la sécurité des applications, CA Technologies aide ainsi ces entreprises à devenir plus productives, à offrir une meilleure qualité d'expérience à leurs utilisateurs et leur ouvre de nouveaux relais de croissance et de compétitivité sur tous les environnements : mobile, Cloud, distribué ou mainframe. Pour plus d'informations, rendez-vous sur **ca.com/fr**.

Références

1 Paul Gerrard, « The Tester's Pocketbook », http://testers-pocketbook.com

2 Paul Gerrard, « Thinking Big: Introducing Test Analytics », octobre 2013, http://blog.gerrardconsulting.com/?q=node/630

3 Julian Harty, Antoine Aymer, « The Mobile Analytics Playbook », http://www.themobileanalyticsplaybook.com/

4 Cem Kaner, « The Insapience of Anti-Automationism », février 2013, http://context-driven-testing.com/?p=69

5 Paul Gerrard, «A New Model for Testing », 2012, http://gerrardconsulting.com/?q=taxonomy/term/281

6 Cory Goldberg, « History of Python », Gource development visualization, juin 2012, https://www.youtube.com/watch?v=cNBtDstOTmA

7 Paul Gerrard, « Regression Testing—What to Automate and How », janvier 2010, http://gerrardconsulting.com/?q=node/547

8 Paul Gerrard, «A New Model for Testing », 2012, http://gerrardconsulting.com/?q=taxonomy/term/281

9 Cory Goldberg, « History of Python », Gource development visualization, juin 2012, https://www.youtube.com/watch?v=cNBtDstOTmA

CS200-218519_0916