LIVRE BLANC | AVRIL 2016

Gestion des données de test

La seule voie vers une livraison continue

Huw Price
CA Technologies

ca technologies

Table des matières

Section 1

Introduction au concept de livraison continue

La livraison continue est un terme devenu à la mode dans le monde du développement logiciel, ce qui explique que nombre de fournisseurs promettent d'en faire une réalité. Pour ce faire, ils proposent des outils qu'ils présentent comme autant de remèdes aux causes habituelles de retard et d'échec des projets. L'idée ainsi véhiculée est que ces outils permettraient aux organisations d'innover en continu, pour livrer des logiciels de qualité dans les temps et les budgets impartis.

L'attrait de la livraison continue est compréhensible. À l'ère de l'économie des applications, les organisations dépendent des logiciels pour fournir de la valeur à leurs clients. Leurs besoins métier et IT sont par conséquent plus alignés, et leur positionnement sur les marchés dépend de leur capacité à fournir de la valeur aux consommateurs au quotidien. Leurs équipes informatiques doivent être en mesure de réagir rapidement à l'évolution des marchés et des attentes des consommateurs, en développant des logiciels qui répondent à des besoins métier critiques et changeants, tout en réduisant les délais et les coûts des phases de test de ces logiciels.

Si de nombreux outils prétendent promouvoir une livraison continue, il s'agit le plus souvent d'outils logistiques, qui ne peuvent être déployés que tardivement dans le cycle de développement logiciel. Ils ne visent que le côté strictement opérationnel de l'approche DevOps, prenant en charge uniquement le développement et les tests de régression de logiciels déjà conçus. En conséquence, les problèmes systémiques qui surviennent en amont du développement, dans la phase de définition des besoins, restent sans solution et provoquent des goulots d'étranglements coûteux et des retards de projets qui rendent impossible la mise en œuvre d'une livraison continue.

Plus particulièrement, les outils proposés négligent la gestion des données de test (Test Data Management, TDM) et la mise à disposition des données de test appropriées au bon emplacement et dans les temps, alors que ces aspects sont essentiels. Lorsque la conception des plans de test et le provisioning des données sont de mauvaise qualité, les équipes manquent des données nécessaires pour réaliser des tests complets sur les systèmes. Pour pouvoir livrer des logiciels à temps et dans les limites de son budget, l'organisation sacrifie la qualité.

Section 2

Une gestion des données de test médiocre, frein à une livraison continue

Pour la plupart des organisations et fournisseurs, la gestion des données de test commence et finit avec la copie, le masquage et, parfois, la création de sous-ensembles de données de production. Purement logistiques, ces méthodes de gestion se concentrent sur les données à déplacer et ignorent les données au moment de la phase de détermination des besoins. Une fois les données de production copiées, puis transférées dans un environnement de développement, de test ou d'assurance qualité, elles sont souvent désignées sous le terme de « copie maître » : un ensemble de données de test réputé parfait, grâce auquel les testeurs peuvent exécuter tous les tests nécessaires pour tester l'intégralité d'un système.

Les opérations de masquage et de création de sous-ensembles contribuent à résoudre les problèmes traditionnellement liés aux coûts d'infrastructure et de mise en conformité des organisations. Cependant, les opérations de masquage et de création de sous-ensembles sur une base de données de production soulèvent également beaucoup de problèmes inhérents à l'exploitation des données de production proprement dites. Des problèmes qui rendent la mise en œuvre d'une livraison continue impossible.

Perte de temps

Comme la plupart des organisations ne disposent généralement pas d'un service de provisioning ou d'une équipe centrale chargée de la gestion des données de test, ce sont des équipes locales qui doivent trouver ou créer ces données. Résultat : les testeurs consacrent parfois jusqu'à 50 % de leur temps à rechercher des données, et les temps d'attente accumulés peuvent représenter jusqu'à 20 % du cycle de développement logiciel. Cela entraîne des goulots d'étranglement, qui sont eux-mêmes des freins à l'atteinte d'une véritable agilité ou d'une livraison continue. Un jour, des collègues de Grid-Tools (qui fait désormais partie de CA Technologies) ont ainsi rencontré une équipe de testeurs qui était supposée réaliser un ensemble de sprints en trois semaines, mais qui en avait déjà passé quatre rien que pour préparer les données !

Avant de masquer ou de créer des sous-ensembles à partir des données de production, les équipes de test doivent rechercher elles-mêmes les données à utiliser ; c'est une tâche fastidieuse, dont la complexité est encore accentuée par des méthodes de stockage des données incohérentes, dans des feuilles de calcul non contrôlées. Par exemple, vous pouvez avoir une base de données dans laquelle un opérateur a choisi d'ajouter une colonne supplémentaire afin d'insérer des informations personnelles sur un utilisateur, car la base ne comporte que trois colonnes pour les cartes de crédit alors que cet utilisateur en détient quatre. De plus, les références des données doivent être intactes. Plus les données sont complexes, plus la tâche est difficile, et le temps et les efforts que les testeurs passent à masquer de telles données dépassent souvent, au final, ceux qu'ils auraient consacrés à créer ces mêmes données à partir de rien.

Cependant, la création manuelle de données peut aussi prendre beaucoup de temps. En outre, comme les données sont généralement créées avec des plans de test spécifiques en tête, elles ont tôt fait de devenir obsolètes et de perdre leur pertinence, dans un monde réel en constant changement. Un exemple fréquemment cité est celui des schémas de trading et des taux de change des devises, des données qui deviennent obsolètes au quotidien. Lorsque l'obsolescence arrive, les données concernées doivent soit être mises à jour (une tâche qui prend elle aussi du temps), soit supprimées (ce qui arrive le plus souvent). Même chose lors d'un changement de version, d'une demande d'actualisation des données, ou lorsque l'organisation doit créer des données à chaque fois qu'un environnement virtuel change.

Il est rarement possible de partager et réutiliser des données créées manuellement. Cette incapacité à exploiter les efforts déjà accomplis pour produire des données génère du travail supplémentaire et force l'organisation à faire un choix cornélien : doit-elle s'attacher à réduire ses coûts et délais de mise sur le marché, ou à livrer des applications logicielles de qualité et comportant toutes les fonctionnalités requises ?

Contraintes liées aux dépendances

Un testeur perd beaucoup de temps à attendre que les données deviennent disponibles ou soient provisionnées. Les organisations et fournisseurs voient souvent dans le cycle de développement logiciel un enchaînement d'étapes linéaires, dans lequel chaque équipe exécute sa tâche avant de transférer le résultat à l'équipe responsable de l'étape suivante. Dans ce mode de fonctionnement, les équipes situées plus loin dans le cycle doivent attendre que les équipes en amont leur envoient des données « adaptées à leur finalité », et peuvent se retrouver dans l'impossibilité d'utiliser ces dernières tant qu'une autre équipe travaille dessus.

Ce mode de fonctionnement va à l'encontre des principes de développement agile parallèle au cœur du concept de livraison continue. Selon ces principes, chaque unité du temps et de la capacité dont dispose un individu pour innover doit être utilisée à sa valeur maximale. Autre inconvénient majeur : tout changement apporté par une seule équipe dans le cycle peut affecter les données de telle manière que lorsqu'une autre équipe les utilisera, ses tests d'application pourraient échouer sans raison évidente.

Si une équipe doit créer des données manuellement ou attendre des jours, voire des semaines, que ces dernières deviennent disponibles, il est logique qu'elle ne soit pas en mesure de réagir rapidement à des changements des besoins métier ni de livrer des logiciels entièrement testés.

Qualité médiocre

L'utilisation de données de production dans des environnements qui ne sont pas destinés à la production pose un problème au niveau de la qualité : dans la mesure où les données de production fournissent généralement une couverture fonctionnelle comprise entre 10 et 20 %, il sera impossible de trouver une méthode d'échantillonnage qui puisse fournir des données répondant aux spécifications de tous les plans de test requis pour construire un nouveau sous-système. Les données de production sont généralement très similaires, car elles sont issues de transactions courantes et « adaptées » afin d'éliminer les données qui risqueraient de provoquer des défaillances système. En conséquence, les tests réalisés sur ce type de données tendront à refléter un parcours idéal et ne permettront pas de tester des scénarios moins fonctionnels ou négatifs.

Or, les scénarios négatifs devraient représenter 80 % des efforts de test, car ce sont précisément ces cas hors norme qui entraînent des défaillances système. Tant que les équipes de test se concentreront sur des scénarios positifs ou idéaux, les défauts passeront inaperçus et se retrouveront en production, occasionnant des remaniements, des retards critiques, une escalade des coûts et des échecs potentiels. Les recherches le montrent : corriger un bogue détecté pendant la phase de test prend 50 fois plus de temps que de le détecter dès la phase de définition des besoins[1], et implique des retards et efforts qui empêchent la mise en place d'une livraison continue.

Il est possible de recourir à la création manuelle, pour compléter la couverture fonctionnelle des données, mais cette façon d'agir est à la fois imprécise et chronophage, et manque de fondement scientifique. De plus, elle n'offre aucun moyen de vérifier que la couverture atteinte est optimale ni de savoir si les références des données sont intactes.

Coûts élevés

Enfin, outre les coûts élevés liés aux remaniements et aux retards de livraison, le processus de copie des données de production affiche une lenteur et un coût extrêmes. D'après un rapport, certaines organisations se retrouvent avec jusqu'à 20 copies d'une même base de données, ce qui implique des dépenses en matériel, licences et assistance élevées.

Sans compter que le stockage de grands volumes de données peut revenir très cher. Une recherche indépendante souligne le fait que le transfert de données vers des emplacements moins onéreux (Cloud inclus) permet de réduire temporairement le coût du stockage, mais ne résout pas le problème de fond, qui est celui de la croissance des données. En d'autres termes, une politique de gestion des données de test qui se résume à migrer, répartir en sous-ensembles et masquer les données ne permet pas de répondre aux enjeux d'un stockage économique des données qui sont traitées par des applications composites modernes de plus en plus complexes.

À cela s'ajoute le risque coûteux de la non-conformité, que même le masquage des données de production ne permet pas de résoudre. Une étude indépendante a révélé que le coût moyen des violations de données avait augmenté de 15 % en 2014, pour atteindre 3,5 millions de dollars par violation, tandis que la nouvelle directive de l'Union européenne sur la protection des données, dont l'entrée en application est programmée en 2016, amènera le montant maximal de l'amende encourue en cas de violation de données à l'équivalent de 100 millions d'euros[2], ou 5 % du chiffre d'affaires global de l'entreprise reconnue coupable de l'infraction, si le montant correspondant est plus élevé.

Le masquage des données de production ne permet pas de garantir la conformité, car le facteur humain reste le vrai danger : plus de 50 %[3] des violations de données peuvent ainsi être imputées à des personnes internes à l'organisation. Autre difficulté rencontrée dans le masquage des données : la nécessité de préserver l'intégrité référentielle des données, sous peine de faire échouer les tests applicatifs. Or, plus les données sont complexes, plus elles sont faciles à déchiffrer car plus il y a de parties d'informations qui peuvent être corrélées.

Le seul vrai moyen d'échapper à des coûts d'infrastructure astronomiques, et d'empêcher la fuite de données sensibles, consiste à ne pas utiliser de données de production. Bien que ces dépenses ne soient pas caractéristiques des organisations souhaitant implémenter une approche de livraison continue, elles vont à l'encontre des principes de cette dernière : elles ne permettent pas aux organisations de livrer des logiciels de qualité, permettant de répondre rapidement et dans les limites budgétaires à l'évolution de leurs besoins métier.

Section 3

Une approche TDM fondée sur les exigences métier

Une organisation qui souhaite implémenter une approche de livraison continue doit repenser ses processus de test et de développement, et non simplement les remanier. Elle doit revoir son approche de la gestion des données de test pour la remplacer par une autre approche, de bout en bout et axée sur les exigences, qui lui permettra d'exécuter les tests plus tôt dans le cycle de développement logiciel et de réduire son risque et la présence de défauts, pour livrer des logiciels de qualité plus rapidement et à un moindre coût.

Plutôt que de fonder sa réflexion sur les plans de test individuels, elle doit penser les données en termes de décisions de conception : elle doit partir des exigences, et concevoir des plans de test auxquels les données sont directement liées. Personnaliser les données de test en fonction des exigences permet de garantir que ces données seront « adaptées à leur finalité » ; en outre, la capacité à les provisionner rapidement aux équipes de test permettra à celles-ci de répondre rapidement à l'évolution des besoins métier.

À cet égard, le principe de livraison continue ne peut pas intervenir tard dans le cycle de développement logiciel : il doit commencer par une idée (issue elle-même des exigences) sur laquelle les équipes de test et de développement basent leur travail, pour réagir rapidement à l'évolution des besoins métier. Ces équipes doivent être en mesure de valider et vérifier facilement les exigences, pour en déduire directement des plans de test liés aux résultats attendus et données virtuelles. Ces tests qui interviennent de manière précoce dans le cycle de développement logiciel rendent la livraison continue possible : ils permettent de condenser tout le travail du cycle de développement dans l'étape de définition des besoins, la suite du travail se déroulant alors de manière plus fluide, même si les exigences évoluent par la suite.

Meilleure définition des exigences

Les exigences elles-mêmes doivent contenir l'ensemble des informations de qualité nécessaires à des fins de test, afin de permettre aux testeurs de créer des scénarios d'utilisation et des plans de test sur cette base, et d'en effectuer le suivi. Ce suivi sera nécessaire pour leur permettre de mettre à jour leurs tests rapidement, au fil de l'évolution des exigences.

Une modélisation basée sur les outils adéquats permettra de produire un organigramme simple, contenant l'ensemble des informations qualitatives du système nécessaires pour le tester. CA Agile Requirements Designer (anciennement Grid Tools Agile Designer) permet de créer un organigramme comportant la logique fonctionnelle nécessaire pour dériver automatiquement le plus petit nombre possible de plans de test afin de garantir une couverture maximale, tout en impliquant les équipes métier, contrairement aux autres méthodes de modélisation (causes et effets et Pairwise)[2].

Par ailleurs, le format organigramme permet de garantir la clarté et l'intégrité. Il remplace les diagrammes « bavards » et encombrés, en découpant les exigences en petites parties d'informations plus digestes, des processus qui reflètent la logique de cause et d'effet d'un système, sous forme d'une série de déclarations « Si… alors ».

Cette méthodologie permet non seulement de réduire les défauts liés aux ambiguïtés présentes dans les exigences (56 % en moyenne[4]), et force l'équipe chargée de la définition des besoins à penser en termes de modélisation du système, en tenant compte de ses contraintes, restrictions et limites. L'organigramme créé sur la base de ces données permet d'identifier chaque chemin possible à travers le système, et de dériver des plans de test offrant une couverture fonctionnelle complète : les tests couvrent donc chacun des chemins possibles dans le système, y compris les chemins négatifs et entraînant des résultats inattendus[3].

Les plans de test peuvent également être reliés à des métriques de complexité, des données virtuelles, des données de test, des scripts d'automatisation, des résultats attendus et une liste d'exigences, pour permettre de concentrer les efforts du cycle de développement dans la phase de définition des besoins.

Provisioning de données « adaptées à leur finalité »

En automatisant les processus de conception des plans de test au moyen d'outils tels que CA Agile Requirements Designer, les testeurs peuvent générer les plans dont ils ont besoin pour tester les systèmes de manière optimale et pour leur entière satisfaction. Cela nous ramène à l'argument central de ce document : l'intérêt d'une meilleure gestion des données de test pour mettre en place une livraison continue. Pour atteindre une couverture fonctionnelle de 100 %, les testeurs ont besoin de pouvoir accéder à des données « adaptées à leur finalité », au bon emplacement et au bon moment.

Qualité

Comme nous en avons discuté auparavant, les données de production ne permettent pas de disposer de la couverture nécessaire pour tester l'intégralité d'un système. En revanche, la génération synthétique de données de test permet de produire de petits ensembles riches de données, couvrant tous les scénarios possibles (y compris des événements qui ne sont jamais survenus auparavant). Ce mode de fonctionnement permet de penser chaque scénario du monde réel comme un point de données, et de créer des données synthétiques y compris pour des scénarios nouveaux ou futurs. CA Test Data Manager (anciennement CA Data Finder ou Data Maker, une solution de Grid-Tools) fait appel à des techniques de profilage de données intelligentes afin de capturer une image précise d'un modèle de données, générant des données riches et sophistiquées qui offrent une couverture fonctionnelle complète.

Dans le cadre d'une approche basée sur les exigences, les données générées sont mises en correspondance avec un plan de test, ce qui permet de garantir qu'elles répondront aux besoins de chaque testeur. Cette « génération de données de plan de test » permet d'accroître les chances des testeurs de détecter les défauts dès le début, et d'éviter des remaniements fastidieux qui empêcheraient une livraison continue.

Temps

Le temps consacré à créer ou manipuler manuellement les données, ou à attendre qu'elles deviennent disponibles, est le principal obstacle à la mise en place d'une livraison continue telle que nous venons de la décrire.

Or, des outils automatisés de création de données tels que CA Test Data Manager peuvent travailler directement avec des SGDBR ou des API d'ERP, pour permettre aux utilisateurs de générer des données aussi rapidement que leur puissance de traitement le leur permet. Les scripts de regroupement peuvent multiplier le volume de données de l'organisation par deux, aussi rapidement que son infrastructure de bases de données le permet. Cette méthodologie permet de garantir la création de « données adaptées à leur finalité » en l'espace de quelques heures plutôt que de semaines, pour permettre aux testeurs d'accéder aux données dont ils ont besoin quand ils en ont besoin.

La puissante fonctionnalité de correspondance de tests assure l'identification et la récupération des données à partir de sources aussi multiples que disparates, puis leur mise en correspondance et leur allocation dans les plans de test appropriés. Il a été prouvé que cette fonctionnalité permet de réduire le temps passé à trouver et provisionner les données de 95 % par rapport aux processus manuels, en plus de permettre aux équipes de reconnaître les tests destinés à échouer du fait de problèmes au niveau des données avant l'exécution.

Le stockage de données de test dans un entrepôt de données centralisé peut également aider à éviter les goulots d'étranglement qui se créent lorsque les équipes attendent que les données deviennent disponibles en aval. Le portail CA Test Data Manager On Demand Test Data propose de construire des formulaires dynamiques dans lesquels les utilisateurs peuvent sélectionner le type de données qu'ils souhaitent sur la base de critères spécifiques (le type de carte de crédit ou l'emplacement géographique, par exemple). Le formulaire alloue ensuite les données nécessaires pour le plan de test ou crée de nouvelles données, le cas échéant. Cet outil permet aux ingénieurs de disposer de plus de temps pour corriger les défauts dévoilés par les tests plutôt que de passer 50 % de leur temps à attendre les données. De son côté, l'organisation peut centraliser la propriété des données au sein de son équipe de sécurité informatique, en fournissant les données sensibles uniquement aux équipes autorisées qui les demandent.

Coût

De meilleures règles de gestion des données de test contribuent à réduire le risque d'escalade des coûts. Grâce à la détection et à la résolution précoces des défauts, l'organisation peut voir le nombre de ces derniers baisser jusqu'à 95 %[5] et réaliser des économies de plus de 50 000 dollars[5] par défaut. En plus, le fait d'utiliser des sous-ensembles de données plus petits et plus riches permet de réduire les coûts d'infrastructure de 50 000 dollars[5] par base de données ; comme l'organisation réalise un nombre de tests de qualité moins important, mais de meilleure qualité, elle revoit du même coup considérablement à la baisse, ses temps et coûts de tests. Le risque de non-conformité coûteuse est également éliminé : comme l'organisation utilise des données synthétiques, les informations sensibles n'ont pas besoin de sortir des environnements de production.

Section 4

Résumé

Copier les données de production dans des environnements qui ne sont pas dédiés à la production empêche la mise en œuvre efficace d'une livraison continue. Les équipes de test risquent de manquer de données « adaptées à leur finalité » au moment de tester les logiciels et seront en conséquence incapables de réagir rapidement à l'évolution des besoins métier. Si les outils de création de sous-ensembles et de masquage peuvent résoudre une partie des problèmes, les données de production n'affichent pas un niveau de qualité suffisant pour permettre la détection et la résolution de tous les défauts.

Adopter une approche du développement logiciel de bout en bout, basée sur les exigences, et traiter les données de test comme une ressource de valeur et réutilisable sont les deux clés à l'implémentation d'une livraison continue. Grâce à cette approche, les équipes sont en mesure de réagir rapidement aux changements des besoins métier, en utilisant des données « adaptées à leur finalité », livrées au bon emplacement et dans les temps, pour tester entièrement les logiciels, en un nombre de cycles aussi réduit que possible.

Grâce à cette approche, l'organisation peut générer des efficiences dans son provisioning de données de test, mais aussi réduire le risque de retards, les remaniements et une escalade des coûts qui rendraient une livraison continue impossible. Elle peut ainsi mettre en place une livraison continue de logiciels de valeur et répondant aux besoins métier critiques, en temps opportun et pour un coût moindre.

Section 5

Références

1 http://www.softwaretestingclass.com/why-testing-should-start-early-in-software-development-life-cycle/

2 http://www.agile-designer.com/resources/test-case-generation-bloor-market-report/

3 « Primer of A Critique of Testing for a conceptual, mathematical treatment of the advantages of flowchart modelling », Llyr Wyn Jones, http://www.agile-designer.com/resources/critique-testing-primer/

Section 6

À propos de l'auteur

Cofondateur de Grid-Tools et Vice-président de CA Technologies

Élu « Directeur informatique de l'année 2010 » par QA Guild, Huw Price a été l'architecte technique de plusieurs entreprises logicielles américaines et européennes. Spécialisé dans les outils d'automatisation des tests, il a lancé de nombreuses solutions innovantes qui ont contribué à refondre les modèles de tests en place dans le secteur des logiciels.

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 la performance 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 en savoir plus sur nos programmes de succès client, rendez-vous sur le site **ca.com/customer-success**. Pour en savoir plus, rendez-vous sur **ca.com/fr**.

1 Ponemon Institute : rapport « 2014 Cost of Data Breach »

2 Projet de réglementation de l'Union européenne sur la protection des données

3 « 2013 Data Breach Investigations Report »

4 « Bender Report: Requirements Based Testing »

5 Métriques issues de l'expérience d'implémentation de Grid-Tools

CS200-196723_0416