Chapitre 13

Gestion des tests et de la qualité

Lorsqu'on travaille avec une équipe en offshore, on doit naturellement prévoir d'y placer une équipe pour assurer les tests. L'objectif est d'améliorer la communication avec l'équipe de développement et de réduire le coût de ces tâches dévoreuses de temps. La qualité de la production du code doit être vérifiée et contrôlée par les membres de l'équipe de test, qui fonctionnent de pair avec les développeurs. Ils vérifient notamment les aspects fonctionnels et techniques du code (performances, sécurité et respect des règles méthodologiques) afin de juger de la qualité des fonctions et du produit final.

On considère communément que l'objectif de l'équipe de test est de trouver les anomalies d'un produit. On pourrait aussi définir son rôle comme consistant à déterminer l'état de qualité d'un livrable ou d'un produit complet de façon objective et continue. Il s'agit ici de mesurer le risque que l'on prend à livrer un produit. Cela implique tout autant d'identifier les anomalies connues que d'estimer la qualité des parties du produit sur lesquelles on n'a jamais effectué de test ou dont les tests remontent à une version antérieure.

Il s'agit d'un travail complexe, où il est souvent plus important de mesurer les risques introduits par les zones d'ombre que de suivre les anomalies déjà détectées à travers un workflow.

On commence par tester le comportement fonctionnel. Le produit livré correspond-il aux fonctionnalités décrites dans les documents de spécification ? On s'intéresse ensuite au respect de certaines normes visibles par l'utilisateur, comme les règles relatives à l'IHM (interface homme-machine), dont on veut vérifier la bonne application. On s'inquiète enfin des exigences techniques, notamment des performances et de l'architecture, et de la consommation des ressources (processeur et mémoire).

La sécurité est toujours considérée comme une préoccupation importante si l'application est accessible par Internet ou par un grand nombre d'utilisateurs. Les objectifs de sécurité sont rarement exprimés d'une façon technique. On souhaite le plus souvent que le niveau de sécurité soit suffisant, sans vraiment définir ce que cela signifie. L'équipe de test doit cependant s'assurer que le niveau de sécurité est mesuré.

Sans faire formellement partie des engagements de livraison, la vérification des procédures et règles méthodologiques apporte un confort supplémentaire pour gérer les évolutions et asseoir le contrôle sur le produit. Il arrive que certaines procédures soient érigées

en règles incontournables, comme de refuser de tester des programmes qui ne vérifient pas certaines règles.

Pour disposer d'une vision juste de l'état de qualité d'un produit, l'équipe de test doit disposer de nombreuses informations. Elle doit aussi gérer son temps de façon à comprendre précisément où se trouvent les gisements d'incertitudes sur la qualité et à trouver les meilleures façons d'en avoir une vision claire. De plus, l'équipe de test doit en permanence être tenue au courant des attentes du client afin de concentrer ses actions sur les domaines les plus utiles.

Les tests commencent aussi tôt que possible, dès l'apparition des premiers livrables. Tous les livrables peuvent être recettés. Plus tôt on connaît les erreurs et les anomalies sur le produit, moins la correction est coûteuse et moins leurs effets sur les autres équipes de développement sont perturbateurs. Attendre d'avoir un ensemble suffisant de fonctionnalités pour commencer les tests est un leurre, qui cherche souvent à masquer une finition insuffisante des premières livraisons, et donc probablement un retard. Les tests correspondent à la confrontation avec la réalité des livrables.

Les tests unitaires

Les tests unitaires constituent la charnière entre l'équipe de développement et l'équipe de test. Leur fonction est de vérifier le fonctionnement de chaque élément d'une fonction selon sa description dans les spécifications détaillées. Pour que le développeur puisse communiquer son code à l'équipe de test, il faut qu'il fonctionne raisonnablement bien.

C'est avec les tests unitaires que l'on garantit que les rôles sont effectivement respectés entre les équipes de test et de développement. Le développeur accompagne ses productions jusqu'à un point où leur fonctionnement est correct à ses yeux. Pour y parvenir, il doit vérifier lui-même par des tests unitaires que sa réalisation respecte les spécifications détaillées.

Il est inutile que les livrables soient échangés plusieurs fois entre les équipes de développement et de test avant de parvenir à couvrir les fonctionnements de base décrits dans les cas d'utilisation. Ces échanges répétés entraînent une perte importante de productivité et des tensions entre les équipes de développement et de test. Les plannings dérapent, et les équipes de développement voient leur production renvoyée pour correction un grand nombre de fois. Ces échanges créent de surcroît des tensions entre les collaborateurs, qui voient leurs plannings glisser et réalisent qu'ils ne pourront tenir leurs engagements.

L'équipe de développement doit effectuer elle-même les tests unitaires visant à vérifier que le produit qui sera livré aux équipes de test correspond aux fonctionnalités attendues. Pour ce faire, elle dispose de cahiers de test unitaire comportant des jeux de données stables et adaptés, précisant la nature des tests qui doivent être réalisés. Le développeur ne considère sa production livrable aux tests que si elle a passé avec succès les tests unitaires. Le fait de les réaliser lui-même lui permet de corriger rapidement les dysfonctionnements.

L'équipe de test qui reçoit un livrable s'attend à un niveau de qualité assez élevé. Il faut que le développeur soit réellement motivé pour livrer la meilleure qualité qu'il peut

atteindre. Certaines équipes en offshore donnent des bonus aux développeurs selon le faible nombre d'anomalies détectées par l'équipe de test afin de favoriser une qualité optimale entre les tests et les développements. Le développeur reçoit une prime pour la qualité de sa livraison, et le testeur pour sa capacité à trouver les anomalies.

Le résultat des tests est consigné dans un tableau indiquant quels tests ont été exécutés et leurs résultats. La première tâche du testeur est d'exécuter les tests unitaires sur sa plate-forme de test afin de vérifier que le développeur a atteint le niveau de qualité minimal permettant de commencer les tests.

Si le client choisit de ne pas monter une équipe de test chez le prestataire et d'assurer lui-même les tests localement, la formalisation des tests unitaires qui était déjà fortement souhaitable, devient incontournable.

EN RÉSUMÉ

Tests unitaires et productivité

Les tests unitaires vérifient le comportement d'une fonction. Pour assurer une meilleure efficacité des équipes de test et de leurs échanges avec les équipes de développement, il faut convenir d'un niveau de qualité minimal des éléments livrés par les développeurs aux testeurs. Les développeurs assurent la réalisation des tests unitaires avant de transmettre leur production aux équipes de test.

Les tests unitaires sont décrits formellement par des cas de test préparés par les équipes de test ou par les chefs de produit et sont accompagnés d'un jeu de test représentatif de tous les cas à tester. On garantit ainsi que le niveau minimal de qualité attendu par les testeurs est assuré par les équipes de développement.

Les tests fonctionnels

Les tests fonctionnels ne peuvent être correctement réalisés que si l'on dispose de spécifications détaillées, complètes et à jour. Les modifications des spécifications et les ajouts doivent faire l'objet d'une mise à jour immédiate des documents de référence.

L'IHM est également construite à partir des documents de référence, souvent à l'aide d'une boîte à outils permettant d'appliquer une normalisation.

Les tests sont construits sous forme de plans de test que l'on peut demander aux équipes en offshore d'élaborer en s'appuyant sur les spécifications ou les règles de l'IHM. On essaie ainsi de couvrir toutes les actions et de vérifier le comportement du produit, d'abord avec des tests positifs (le produit assure-t-il le service qu'il est censé rendre ?) puis avec des tests négatifs (le produit gère-t-il toutes les erreurs qui peuvent se produire ?). Les tests sont naturellement placés sous la supervision du chef de produit, qui en apprécie la qualité et l'exhaustivité.

À chaque itération, un point complet est effectué sur les livraisons. L'état d'avancement du produit est entièrement testé afin d'évaluer sa qualité. On peut aussi essayer de viser une intégration continue avec un build quotidien, bihebdomadaire ou hebdomadaire.

L'équipe de test s'inquiète en permanence du niveau de qualité du produit. À chaque build, le testeur cherche à savoir quel est l'état de qualité de chaque fonction, puisqu'elle est potentiellement modifiée.

Si les développeurs n'ont pas modifié le code source de la fonction depuis qu'elle a été testée, on peut supposer que les résultats antérieurs sont toujours valides, sans toutefois pouvoir en être certain. Il se peut que cette fonction fasse appel à d'autres services, qui ont eux été modifiés ou qui contiennent depuis peu des anomalies. On peut aussi avoir fait une correction mineure, théoriquement bénigne et parfaitement circonscrite, qui engendre des anomalies majeures inattendues dans d'autres parties du produit. Il se peut encore que la fonction ait subi des régressions sans que les causes en soient évidentes.

On ne peut vraiment connaître l'état de qualité d'une fonction qu'en la testant entièrement. Plus les tests sont anciens, et moins les résultats obtenus lors de ces tests sont fiables quant au dernier build.

Le test d'une fonction est long et répétitif, et il est pratiquement impossible de réaliser manuellement tous les tests à chaque intégration, surtout si celles-ci ont lieu très fréquemment. Il faut recourir aux tests automatisés pour qualifier les builds quotidiens ou bihebdomadaires.

Les tests minimaux, ou MAT

Il est recommandé de séparer les tests fonctionnels en deux niveaux de test, les tests profonds, qui correspondent à des tests complets de toutes les fonctionnalités, et les tests rapides, ou MAT (Minimal Acceptance Test), qui couvrent succinctement une série de fonctions essentielles.

Il faut généralement plusieurs jours à l'équipe de test pour réaliser les tests complets, alors que les MAT sont davantage limités dans le temps. L'idéal est de les exécuter en deux heures. S'ils sont automatisés, on peut aller plus loin dans leur exécution que s'ils sont réalisés manuellement.

Les MAT n'étant pas exhaustifs, ils ne peuvent garantir pleinement le fonctionnement du produit. Ils se concentrent sur les scénarios d'utilisation les plus importants et les plus utilisés par les clients. Ces tests légers peuvent être réalisés rapidement lorsqu'on souhaite s'assurer d'un niveau de qualité minimal ou qu'il n'y a pas de régression majeure évidente.

Un MAT est construit de façon à couvrir l'effet d'une action au minimum sur tous les modules et tous les aspects techniques de l'application. Au moins une utilisation de tous les composants est testée, ainsi qu'au moins un accès à tous les outils de middleware, de façon à détecter, par exemple, l'absence de déploiement d'un fichier.

Le MAT est utilisé chaque fois que l'on doit vérifier rapidement que le produit fonctionne correctement et qu'on n'a pas le temps de le tester totalement. On utilise le MAT pour valider que le livrable reçu par l'équipe de test fonctionne, par exemple quand on doit livrer un patch ou des Hot Fix. Il permet en outre de vérifier qu'un déploiement est correct, puisqu'il est censé utiliser au moins une fois tous les aspects techniques du produit, et donc tous les fichiers, tous les éléments du middleware et tous les types d'interactions avec l'extérieur.

Les MAT sont une série de tests très importants, que l'on a tout intérêt à automatiser afin d'en augmenter l'étendue. Ils doivent être exécutés le plus souvent possible, parfois sur une même version mais sur des plates-formes différentes. Ils peuvent alors faire office de tests de validation de déploiement et servir de référence tout au long du travail avec l'équipe distante.

EN RÉSUMÉ

Tests profonds et tests minimaux

Les tests profonds permettent de juger de l'état de qualité d'un produit dans son intégralité selon un plan de test complet. Ces tests très longs ne peuvent être effectués sur tous les builds. Ils ne conviennent donc pas lorsqu'on doit livrer une version rapidement.

Les tests minimaux, ou MAT (Minimal Acceptance Test), couvrent succinctement tout le produit, en tentant de se concentrer sur les scénarios les plus importants. Les MAT sont généralement automatisés. Utilisés pour toutes les recettes rapides visant à estimer l'état de qualité du produit ou à détecter les régressions majeures, ils peuvent également servir à valider le déploiement sur une plate-forme.

Juger de l'état de qualité

Les MAT peuvent être exécutés sur tous les builds, donnant ainsi une première idée de l'état de qualité et de stabilité d'une livraison. Cet aperçu est toutefois peu profond, et il ne permet pas de juger de l'état du produit ni de sa progression dans le temps vers un produit stable, utilisable et de qualité.

Pour suivre l'état de qualité, on peut utiliser un tableau, dit *Acceptance Sheet*, permettant de suivre le résultat et l'ancienneté des tests pour chaque fonctionnalité (*feature*). On visualise de la sorte sur un même document les résultats des tests les plus récents et l'état de connaissance que l'on a des campagnes de tests plus anciennes. Un modèle de tableau de ce type est fourni en annexe.

Le tableau 13.1 indique les principes de fonctionnement de l'Acceptance Sheet.

Tableau 13.1. Exemple d'Acceptance Sheet

Fonctionnalité	Campagne de test 12	Campagne de test 13	Campagne de test 14	Campagne de test 15	Campagne de test 16
Fonctionnalité 1	Bogue mineur CR0034, CR0035	Bogue mineur CR0034			OK
Fonctionnalité 2	OK			Bogue majeur CR0056	
Fonctionnalité 3	OK				
Fonctionnalité 4	Bogue majeur CR008	OK	Bogue mineur CR0087	OK	

Le tableau montre que l'on ne connaît de façon sûre que l'état de la fonctionnalité 1, les autres fonctionnalités n'ayant pas été testées sur la dernière campagne de test. On ne sait pas vraiment quel est l'état de la fonctionnalité 3, dont les derniers tests sont anciens. Pour les fonctionnalités 2 et 4, on dispose d'une connaissance vraisemblable de l'état de qualité, puisque les informations remontent à la campagne de test précédente. L'état de chaque colonne peut bien sûr être codé par des couleurs représentant visuellement l'état

de qualité. Lorsqu'une anomalie est identifiée, on place son code dans la case qui la représente de façon à vérifier son état ou sa priorité plus facilement.

L'analyse de l'Acceptance Sheet permet de se faire une idée du risque que l'on prend à livrer une version. On peut juger de la qualité du produit global, des régressions (on passe du vert au jaune ou au rouge), du risque que l'on prend en ne retestant pas une fonctionnalité d'après l'ancienneté du dernier test passé avec succès, et l'on dispose du détail des anomalies connues sur chaque fonctionnalité.

Si le produit est destiné à plusieurs plates-formes de déploiement, par exemple, si l'on a une portabilité sur plusieurs systèmes d'exploitation ou plusieurs bases de données, on peut prévoir d'effectuer une rotation des plates-formes de test et de noter la plate-forme utilisée pour les tests pour chaque campagne de test.

Ce tableau est particulièrement utile pour les développements en offshore car il permet de connaître la stabilité d'un produit en cours de construction. Cela évite que les équipes de validation ne travaillent sur des fonctionnalités dont les anomalies sont connues. Il permet également de donner des priorités de réalisation aux tests. Il sert alors d'outil de décision sur les tests à réaliser et évite de travailler sur des fonctionnalités connues pour être boguées de façon à se concentrer sur celles qui sont réputées saines.

Si l'on dispose d'une intégration continue et si des tests automatisés sont exécutés quotidiennement, on voit le produit non seulement se construire pas à pas, mais se stabiliser. On dispose d'une source d'information permettant de juger de la réalité de l'information fournie par les équipes sur la progression du développement.

Les tests fonctionnels transversaux

Les procédures décrites précédemment cloisonnent fortement les tests par fonction et par module, toute l'organisation étant structurée par composant. De son côté, l'utilisateur s'intéresse avant tout à des tests métier, qui sont transversaux et suivent son cheminement à travers des séries d'actions faisant appel aux différents modules de l'application.

Il se peut que deux modules soient communément utilisés ensemble, par exemple, la mise à jour d'une information doit être reprise dans un tableau synthétique faisant partie d'un autre module. Le cloisonnement par module peut faire que l'on ne teste jamais ces croisements de mises à jour. Les tests transversaux visent à combler ces défaillances. Ils sont souvent organisés sous forme de scénarios métier, dans lesquels on accompagne les actions d'un utilisateur final à travers un cheminement sur plusieurs modules.

Si nous insistons ici sur un sujet qui peut paraître évident c'est qu'il est une source récurrente de problèmes avec l'offshore lorsqu'on met en place une méthodologie industrielle pour la première fois. On a alors tendance à se concentrer sur des tests très structurés, et l'on en oublie la vision d'ensemble.

Lorsqu'on met un produit testé par fonction entre les mains d'un utilisateur final, des anomalies majeures sont souvent trouvées dès les premières minutes d'utilisation, qu'il était impossible de détecter lors des tests de fonction. La confiance s'effondre alors rapidement, surtout si l'on a affirmé que le produit était bien testé, fort des nombreux tests cloisonnés effectués.

Comme les tests par module et par fonction, les tests transversaux doivent être assurés et renseignés dans des Acceptance Sheets. Les scénarios sont beaucoup plus difficiles à réaliser, car ils peuvent être très nombreux. Les chefs de produit sont particulièrement

sollicités pour donner une réalité métier à ces tests. Lorsque les utilisateurs fi nals rencontrent de nouvelles anomalies, ces cas de test sont ajoutés aux tests transversaux. Les tests transversaux sont structurés et documentés. Ils sont en outre reproductibles.

Les tests intuitifs

Les tests intuitifs sont des tests qui ne sont pas formalisés. On y fait ce que l'on veut pour détecter des anomalies selon son intuition ou son expérience. Les tests formalisés ne peuvent en effet tout prévoir. Par exemple, il est possible que l'on ne prévoie pas de tester formellement la couleur du fond d'écran, bien qu'elle soit définie dans la charte graphique du projet. Une erreur sur la couleur de l'écran est pourtant immédiatement visible de tout utilisateur.

C'est un des défauts que l'on attribue très souvent aux équipes en offshore. On dit qu'elles suivent les procédures à la lettre mais ratent les erreurs les plus flagrantes, sous prétexte qu'elles ne sont pas détectées dans les scénarios de test.

Le testeur qui connaît les développeurs et leurs erreurs peut anticiper les domaines où ils créent le plus souvent des anomalies et peut les rechercher directement. Bien sûr, ces tests ne sont pas reproductibles et ne sont pas documentés, mais ils sont d'une remarquable efficacité.

Il est recommandé d'allouer un certain temps sur chaque itération pour ces tests intuitifs.

Enrichissement des tests

Lorsque des anomalies sont détectées par les utilisateurs, il est important de comprendre où se trouve la faille dans les processus de test et de la combler, dans la mesure du possible, afin d'éviter que cette famille d'anomalies ne perdure.

Comme les équipes de test sont distantes, il est essentiel que les anomalies détectées en dehors d'elles leur soient transmises et qu'elles soient considérées non comme des fautes de leur part, mais comme des failles à combler.

Si l'on ne prend pas soin d'avertir les équipes de test des anomalies, les testeurs ne peuvent améliorer les processus de test. Du côté du client, on pensera que les testeurs ne sont pas très efficaces puisqu'ils travaillent en dehors de la réalité. Il faut donc toujours communiquer aux équipes en offshore le niveau de satisfaction des utilisateurs. Cette communication n'est jamais naturelle. Il est du ressort du chef de projet du client de faire comprendre aux équipes distantes ce que pensent les utilisateurs du produit.

Si les équipes en offshore estiment que le produit est apprécié alors que l'utilisateur se plaint d'une grande quantité d'anomalies, l'hiatus de perception peut vite devenir intolérable et est compris comme un signe de dysfonctionnement majeur de l'offshore. Il est facile de mettre en place un retour d'information sur la satisfaction des clients et sur la vie du produit. Les retours positifs sont de surcroît des sources de motivation. Quant aux retours négatifs, ils motiveront la concentration sur les faiblesses constatées.

Grâce à une communication efficace, l'équipe de test comprend les problèmes des utilisateurs et propose des moyens d'y remédier. Elle se trouve plus impliquée dans la vie du projet, et il y a tout lieu d'espérer que la recherche de la qualité s'en trouve renforcée.

EN RÉSUMÉ

Satisfaction des utilisateurs et motivation

Les équipes en offshore doivent être informées du niveau de satisfaction des utilisateurs finals. Une différence de perception de ce niveau devient vite intolérable. Une bonne connaissance de la satisfaction des utilisateurs peut être une source de motivation et de concentration sur les sujets les plus importants pour ces derniers.

Les tests techniques

Les tests techniques n'attirent vraiment l'attention que lorsque le projet va mal ou que les performances ne sont pas au rendez-vous. On en trouve une grande variété, depuis les tests de performance jusqu'aux tests de charge, en passant par les tests des services techniques.

Si le client travaille sur un projet important et qu'il ait choisi d'automatiser certains tests, on peut mettre en place plusieurs types de tests techniques en s'appuyant sur cette automatisation. On pourra ainsi détecter les régressions sur les temps de réponse et identifier la modification qui en est la cause.

Bien souvent, l'équipe qui s'occupe des services techniques jouit d'une situation particulière. Sa production n'est pas directement testée par les équipes de test puisque ces dernières ne peuvent aisément tester des bibliothèques de fonctions sans interface utilisateur. Cette production est cependant recettable et testable, et il n'y a pas de raison que les services techniques ne soient pas testés comme les autres. Le testeur représente ici le client de ces services, c'est-à-dire les développeurs fonctionnels.

Les testeurs des services techniques construisent de petites applications qui les utilisent. Les scénarios de tests peuvent être formalisés. Le meilleur test est encore la mise en condition, consistant à appliquer la nouvelle version des services techniques aux réalisations fonctionnelles existantes. L'équipe de test technique peut vérifier ce fonctionnement sans que les équipes chargées des développements fonctionnels soient affectées par les périodes de stabilisation. Les nouveaux services techniques sont livrés après que les développements fonctionnels ont été vérifiés.

Bien souvent, une production défaillante de services techniques peut retarder des livraisons, alors que les équipes fonctionnelles n'y sont pour rien. Pour que les équipes chargées des développements fonctionnels s'engagent pleinement sur leurs itérations, il faut éviter dans la mesure du possible que ces défaillances extérieures ne leur donnent l'occasion de se déresponsabiliser. Si l'on parvient à isoler les livraisons des services techniques, on fait plus clairement endosser leurs responsabilités par les équipes.

Couverture des tests

Certains produits de test permettent de tester la couverture des tests en marquant les lignes de code exécutées lors des séquences de test. Ces produits ne sont malheureusement pas disponibles pour tous les langages de programmation.

Lorsqu'on peut les utiliser, ils se révèlent très utiles pour repérer des pans entiers du code source qui n'auraient pas été testés. On peut dès lors revoir les scénarios de test ou estimer les tests partiels suffisants.

Ce type d'outil permet d'éliminer une forte incertitude sur les anomalies cachées dans le produit. Si l'on connaît précisément les portions de code testées régulièrement, et surtout les fonctions qui ne font pas encore l'objet de tests structurés, on peut estimer assez justement le risque que l'on prend en considérant l'application comme testée. On ne connaît certes pas la quantité d'anomalies qui persistent, mais on a une bonne idée du risque que l'on prend si l'on décide de ne pas tester ces parties de code source.

Comme les tests en offshore sont le plus souvent fortement structurés, on peut savoir précisément quelle est la proportion du code source qui est contrôlée régulièrement.

Il est recommandé de lancer une campagne de détection de la couverture des tests lorsque le produit est terminé environ aux trois quarts. Cela permet de décider des actions correctives à engager en ayant encore le temps de les réaliser.

Vis-à-vis du client comme des utilisateurs finals, il est particulièrement valorisant de pouvoir quantifier la quantité de lignes de code testées de façon automatisée, à la fois régulièrement et occasionnellement. Cela n'excuse sans doute pas d'éventuelles anomalies majeures mais fait la preuve de l'engagement de moyens sur les tests.

Les tests de déploiement

Situés à la charnière entre l'offshore et le client, l'intégration et le déploiement sont des activités délicates. Les plates-formes chez le client sont souvent différentes de celles qu'on utilise en offshore, et les problèmes de déploiement sont très courants.

Comme pour la plupart des autres activités, le livrable, c'est-à-dire l'application déployée, doit être recetté. Pour ce faire, on peut exécuter un scénario de test à la main ou exécuter le MAT, ou une version réduite du MAT, de façon automatisée. Une exécution sans erreur du MAT ou d'une version réduite est indicative d'un bon déploiement.

Tests et mesure du risque

Lorsque les équipes de test sont proches des utilisateurs, elles sont perçues assez naturellement comme étant chargées de limiter les risques d'insatisfaction. Avec des équipes de test en offshore, les utilisateurs se montrent toutefois moins tolérants aux erreurs qu'ils détectent et sont prompts à déclarer les testeurs éloignés des réalités.

Les équipes de test ne doivent jamais perdre de vue qu'elles représentent les utilisateurs dans l'équipe en offshore. De ce point de vue, elles sont les plus importantes. Elles doivent en être conscientes, et l'importance de leur rôle doit être reconnue de tous les autres collaborateurs. Les développeurs ont parfois l'impression qu'une hiérarchie par la compétence s'établit naturellement. Donner une grande valeur aux tests bouscule cette hiérarchie et affirme le rôle des testeurs.

Lorsque les équipes de test se mettent dans la peau de l'utilisateur final, elles doivent mesurer les dysfonctionnements du produit comme le ferait cet utilisateur. Pour cela, elles doivent comprendre exactement comment le produit sera utilisé.

Les fonctionnalités du produit qui sont utilisées en permanence doivent être parfaitement testées car l'utilisateur ne tolère aucun défaut. Les mauvais alignements, les lenteurs d'exécution, les fautes d'orthographe, etc., lui sont insupportables. Il est en revanche beaucoup plus tolérant à l'égard des fonctionnalités qu'il utilise rarement et qui peuvent comporter certaines anomalies, même plus graves, pour peu qu'elles ne créent pas de gêne importante. Les testeurs doivent donc prendre de l'altitude par rapport aux procédures pour comprendre par eux-mêmes l'importance relative des différents tests.

Les régressions sur un produit créent des réactions particulièrement vives de la part des utilisateurs. S'ils peuvent comprendre qu'un cas ne fonctionne pas, ils considèrent comme de la pure négligence qu'une anomalie apparaisse sur des fonctionnalités qui donnaient satisfaction auparavant.

La mesure du risque ne peut être juste que si l'on essaie de l'estimer tel qu'il est perçu par le client. Les équipes de test pourraient passer un temps infini pour assurer qu'un produit ne contient pas d'anomalies et se comporte comme on le souhaite. Il convient donc de vérifier régulièrement que le temps passé à contrôler un produit est rentable. Une telle estimation est toutefois délicate. Plus le temps passe, plus les anomalies sont difficiles à détecter, et plus leur coût de détection devient important.

Si les tests sont bien organisés, ils doivent permettre de se concentrer d'abord sur les anomalies le plus importantes. Le moment arrive ensuite où l'on préfère mettre le produit en production et gérer les anomalies détectées par les utilisateurs plutôt que de continuer à détecter d'autres anomalies.

La figure 13.1 illustre le rythme de détection des anomalies build après build comparé au coût des détections et au nombre d'anomalies non détectées dans le produit.

Le risque est difficile à estimer, car on ne sait pas vraiment mesurer les anomalies restées cachées dans le produit. On prend les décisions par rapport au nombre d'anomalies répertoriées et à la difficulté à détecter de nouvelles anomalies. C'est pour éviter que cette dernière mesure ne soit faussée que l'on recommande d'exécuter des tests intuitifs en dehors de toute procédure. Cela permet de ne pas appliquer les mêmes scénarios de test mécaniquement, car ceux-ci découvrent de moins en moins d'anomalies lors des builds successifs jusqu'à ne détecter que les anomalies de régression lorsque tous les scénarios ont été vérifiés avec succès au moins une fois.

EN RÉSUMÉ

Mesure du risque et coût des tests

Le risque doit être estimé par rapport aux engagements du client sur la livraison du produit et surtout par rapport à la satisfaction des utilisateurs. Les équipes de test sont les plus proches représentants du client et des utilisateurs chez le prestataire. Il est important qu'elles essaient de mesurer en permanence le risque en se mettant à la place de l'utilisateur et en estimant son niveau de satisfaction. Elles doivent se faire les avocats des utilisateurs.

Les décisions doivent être prises en visant la satisfaction des utilisateurs finals afin de déterminer comment équilibrer les tests pour limiter au mieux le risque d'insatisfaction. On détermine ainsi le pourcentage des tests structurés (scénarios de test réalisés régulièrement), automatisés, techniques et intuitifs. L'équipe de test doit être une force de proposition pour optimiser le risque, tant auprès du client que des autres équipes chez le prestataire.

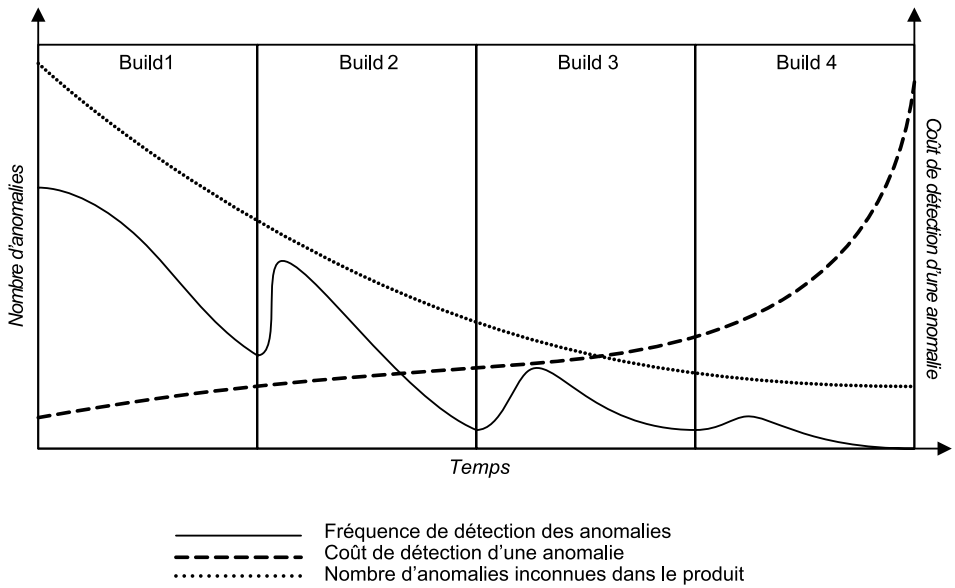

Figure 13.1. Coûts et efforts de détection d'anomalies

Conclusion

Bien souvent, les équipes de test sont considérées comme moins importantes que les équipes de développement. Les services qu'elles rendent sont souvent mal considérés, et elles sont tenues pour responsables de nombreux maux dont elles ne portent pourtant qu'une responsabilité limitée. Toute anomalie qui serait détectée après leur travail est considérée comme inacceptable.

En réalité, les testeurs assurent avant tout des tâches planifiées et documentées, dont on peut juger la bonne ou mauvaise réalisation de façon assez simple. C'est leur première mission, mais il ne faut surtout pas qu'ils s'arrêtent là. Ils doivent représenter le client et l'utilisateur du produit chez le prestataire et faire tout leur possible pour limiter les risques sur le produit et accroître la satisfaction des utilisateurs.

Le travail des équipes de test est extrêmement difficile, car elles sont les coupables désignés de tous les maux, ou presque. Comme elles assurent les dernières tâches avant livraison, elles sont toujours pressées par le client pour terminer leurs tâches de test au plus vite. C'est d'autant plus difficile que, bien souvent, elles ont reçu les livrables en retard de la part des équipes de développement. Lorsqu'une anomalie grave est découverte, l'équipe de test est immédiatement montrée du doigt pour ne pas l'avoir découverte

avant l'utilisateur, même si cette anomalie ne fait pas partie des scénarios de test habituels.

Les équipes de test ont un travail beaucoup moins créatif et motivant que la plupart des autres équipes en offshore. Leur travail est répétitif, et il est difficile de leur conserver une forte motivation.

Les équipes de test doivent s'aligner en permanence sur les attentes du client, ce qui leur demande un effort d'écoute continu, que les autres équipes n'ont pas besoin d'effectuer dans de telles proportions. C'est d'autant moins facile qu'il s'agit d'une écoute à distance du client pour comprendre sa façon de réaliser la recette et les sujets qui lui tiennent le plus à cœur. Les équipes de test sont bien les meilleurs représentants du client chez le prestataire et doivent être reconnues comme tels, si l'on souhaite rendre leur rôle plus motivant et influent.

Une équipe valorisée et bien organisée apporte un confort important dans la livraison des exécutables et permet de stabiliser beaucoup plus tôt les livrables en leur faisant atteindre un niveau de qualité satisfaisant aux exigences du client. Elle évite en outre les échanges inutiles entre le client et le prestataire pour stabiliser le produit.

Lorsqu'on parvient à créer une telle équipe de test, le travail en offshore devient beaucoup plus simple, et l'on évite de nombreux heurts. La satisfaction est au rendez-vous, chez le client comme chez le prestataire.