Non, un bug n'est pas normal !

Aurore Xemar

Lors d'un Gemba récent, la discussion s'étend sur les raisons d'un retard dans l'implantation d'une solution informatique chez un client. L'équipe met en évidence une erreur dans un script d'injection de numéros de série qui rend impossible l'activation du service. Une contre-mesure manuelle a été identifiée, aussi l'équipe technique refuse de corriger le défaut dans l'immédiat car elle a "d'autres priorités". L'un des ingénieurs finit par lâcher qu'avec un système aussi complexe, "c'est normal d'avoir des bugs".

Non, un bug n'est déjà pas normal en cours de développement, et ça l'est encore moins s'il est détecté en pre-production ou en production. Un "fix" n'est pas non plus une étape normale dans la construction d'une solution. Le client n'a pas pu utiliser le système pendant trois jours, et l'entreprise s'est vue privée du chiffre d'affaires correspondant. Il n'y a rien de normal dans tout cela.

Les équipes techniques expliquent inlassablement qu'elles s'améliorent puisqu'elles travaillent à corriger plus vite, et lorsque la situation devient critique elles en viennent à ajuster la taille des équipes. Ce n'est pas de l'amélioration, mais de la réaction.

Car le bug reste un défaut. Et comme tout défaut il entraine retards, désorganisation et surcoûts — voire des pénalités.

Mais le bug est aussi un formidable vecteur d'amélioration, dès lors que les équipes l'acceptent et cherchent à en découvrir l'origine, à le comprendre en profondeur, bref à apprendre. C'est une opportunité pour continuellement améliorer le produit, réduire les coûts de développement et de production. Cela reste un levier indéniable de performance, de travail collaboratif, de satisfaction pour les équipes et les clients. Mais pourquoi sommes-nous si loin de cet idéal en informatique ?

Peut-être s'agit-il d'une interprétation erronée du manifeste agile, et d'une importance démesurée donnée à la réactivité ? Il faut livrer vite, et la correction doit elle aussi être rapide. C'est aussi la conséquence du travail en silos, qui fait que les développeurs n'ont une compréhension que très réduite des conséquences de leurs décisions. Enfin, la pratique de la résolution de problème est peu répandue, avec une culture du "fix" très ancrée au détriment d'un "solve" apprenant et durable.

Il semble que ce qui a été durement acquis dans l'industrie n'en est parfois qu'à ses prémices dans les équipes informatiques. Prenons l'écoute du client. Dans les départements informatiques la satisfaction client est mesurée via le NPS au point de contact. Ce qui est mesuré ? La capacité à répondre professionnellement et efficacement à la demande de support, donc au défaut et à la réclamation. La capacité à ne pas exposer le client au défaut est rarement suivie, et encore moins remise en question. Ou encore certains défauts et réclamations restent tabous alors même que ces évènements donnent un accès direct à la connaissance du besoin client et offrent autant d'opportunités de progrès.

Comment changer les choses ? Par un changement dans l'attitude du manageur qui doit savoir accueillir l'erreur comme un levier d'amélioration et d'apprentissage. Dans sa capacité à générer continuellement un environnement qui encourage la remise en question. A véhiculer un changement de comportement et considérer le client tant interne qu'externe au-delà du

ticket, ou considérer le bug (et le rework) comme une non-valeur qui augmente les coûts totaux de développement et retarde in fine les livraisons. Un management qui refuse qu'un défaut ne puisse être poussé à l'étape suivante et encourage continuellement son équipe à toujours mieux détecter, comprendre leur origine pour mieux les résoudre. Apprendre à voir et à apprendre, en quelque sorte.

Le Jidoka est un des deux piliers fondamentaux du Lean, c'est la capacité à ne jamais laisser un défaut se propager à l'étape suivante. Un des moyens consiste à stopper l'opération automatiquement quand le problème apparait et l'empêcher de passer à l'étape suivante. Les tests automatiques permettent par exemple de développer une détection efficace et au plus tôt et un processus d'escalade et de résolution.

L'automatisation des tests est un premier pas vers le Jidoka, mais la véritable mutation réside dans la capacité à comprendre les causes qui ont provoqué ces anomalies et à apprendre à développer toujours mieux et donc plus vite aux yeux du client.

En référence à Toyota, réduire les coûts ce n'est pas acheter moins cher, mais c'est être capable de faire les choses pour moins cher, c'est-à-dire de viser à réduire toute non-valeur et non-qualité dans la chaine de développement, de production et de livraison. Bref, la recherche de compétitivité commence par chercher à ne jamais produire de défaut.