Trois problématiques d'agilité auxquelles les développeurs doivent faire face

Les entreprises sont soumises à une pression constante pour accroître la vélocité de leurs développeurs. Sur des marchés où la concurrence est particulièrement vive, les équipes de développement doivent en effet produire du code, mais également construire, tester et déployer de nouvelles fonctionnalités à un rythme soutenu pour satisfaire et fidéliser les clients. Au-delà de cette nécessité existentielle de maintenir la cadence, le choix des outils et l'architecture organisationnelle ont des conséquences involontaires qui pénalisent certaines équipes et parfois empêchent les développeurs de donner vie à leur code.

Ces erreurs ne se contentent pas de retarder le lancement de produits : elles peuvent également avoir un coût humain inattendu. En effet, de nombreux développeurs préfèrent jeter l'éponge plutôt que de travailler pendant des mois sans arriver à leurs fins. Il m'est personnellement arrivé de décliner une proposition d'emploi où j'aurais pourtant travaillé sur une technologie intéressante dans un environnement agréable ; mon enthousiasme est retombé lorsque j'ai entendu le responsable me dire en soupirant : « Nous n'avons pas lancé de code en production depuis six mois ». Dans un secteur rendu célèbre par sa volonté « d'aller vite et de tout casser », quel développeur peut avoir envie de plancher sur un projet sans voir ses efforts déboucher sur un résultat concret avant six mois, voire davantage ?

Si plusieurs facteurs influencent la vélocité des programmeurs, l'actuel environnement de développement se heurte à trois obstacles majeurs que nous allons présenter ci-après, tout en essayant de comprendre comment les équipes peuvent les surmonter et se remettre sur la bonne voie.

Adapter et orchestrer les différentes équipes

Ce problème a été mis en lumière dans un récent tweet où Joe Magerramov affirme que le principal obstacle à la vélocité fonctionnelle est d'ordre architectural, notamment lorsque « le lancement de toute fonctionnalité nécessite la participation de plusieurs équipes, surtout si elles sont dispersées sur le plan organisationnel ». Je ne suis pas convaincu qu'il s'agisse du seul, ni même du principal obstacle, mais Joe Magerramov n'a pas totalement tort. Il est important d'évaluer les équipes de développeurs complexes et de les rationaliser pour éviter qu'un organigramme même complexe retarde le lancement d'une nouvelle release. En bref, la ligne droite sera toujours le chemin le plus rapide.

Des outils de rollback limités

J'ai la conviction que des tests en cours de production permettent d'accroître la vélocité, mais également que nous devrions développer des outils efficaces pour tester plus souvent en production.

En concevant nos systèmes pour avoir la possibilité de revenir à une version antérieure lorsque certains déclencheurs (triggers) sont atteints, nous permettrons aux développeurs de passer une bonne nuit, même s'ils ont déployé un nouveau code dont ils ne sont pas sûrs à 100 %.

La problématique des émulations locales

Il y a bien longtemps, les développeurs pouvaient exécuter la totalité de leur interface Web (front-end) sur leur PC, de sorte que la moindre modification apportée au code était testée localement avec une précision de 99 % par rapport à son fonctionnement en production. Mais avec l'avènement de l'approche « sans serveur » (serverless), nous travaillons depuis quelques années dans des environnements où une telle approche est pratiquement impossible. Au moment où ce virage a eu lieu, nous aurions dû créer les outils nécessaires pour émuler localement de nouveaux environnements. Tant que nous n'aurons pas rattrapé notre retard et inventé de telles solutions, la vélocité des développeurs en pâtira.

En deux mots, augmenter la vélocité des développeurs n'est pas une question binaire impliquant un changement d'outils ou une refonte de l'organisation. Il s'agit davantage de modifier l'état d'esprit pour accepter la publication de codes imparfaits comme la norme et non plus l'exception, en sachant qu'une augmentation minime des erreurs se traduira par une augmentation considérable de la vélocité des développeurs et la fourniture de nouvelles fonctionnalités à des clients satisfaits. De tels changements peuvent être minimes. Vos analyses post-mortem seront peut-être plus fréquentes et plus ludiques et n'aboutiront pas nécessairement à la recherche d'un coupable !

Prenez une seconde pour observer vos équipes de développeurs. Quels sont les évènements susceptibles de les ralentir ? Si vous accélérez le rythme de publication de codes légèrement imparfaits, votre équipe progressera-t-elle beaucoup plus rapidement ?