Les programmeurs ne sont pas des branleurs !

Le travail intellectuel des programmeurs souffrirait-il d'un manque de visibilité et de reconnaissance aux yeux d'une logique managériale qui cherche à mesurer le travail effectif avec des critères dépassés ? C'est ce que laisse entendre ce témoignage qui au détour d'une plaisante anecdote met l'accent sur un relatif malaise d'une profession qu'il est difficile de cerner de l'extérieur, et même de l'intérieur d'une entreprise.

Are Your Programmers Working Hard, Or Are They Lazy? article paru sur le blog de l'auteur

Traduction Framalang : sinma, goofy, KoS

Vos programmeurs travaillent-ils dur, ou sont-ils fainéants ?

par Mike Hadlow

Quand quelqu'un effectue une tâche physique, il est facile d'évaluer à quel point il travaille dur. Vous pouvez le voir physiquement en mouvement et en transpiration. Vous pouvez aussi voir le résultat de son travail : le mur de briques s'élève, le trou dans le sol devient plus profond. Reconnaître et récompenser un dur labeur est un instinct assez fondamental chez l'être humain, c'est l'une des raisons pour lesquelles nous trouvons les sports d'endurance si fascinants. Cette reconnaissance d'un dur travail physique devient un problème quand on l'applique à la gestion de personnes qui travaillent à des activités techniques ou créatives. Souvent, les travailleurs intellectuels efficaces n'ont pas l'air de travailler très dur.

En 2004, j'étais développeur junior dans une grande équipe qui s'occupait du système de fourniture et de facturation d'une entreprise de télévision par câble. Comme tous les grands systèmes, il était constitué de plusieurs unités relativement indépendantes dont s'occupaient des personnes ou des équipes différentes. Les départements de TV analogiques et numériques étaient presque entièrement séparés, pris en charge par des équipes différentes.

L'équipe de la TV analogique avait décidé de fonder son système autour d'une pré-version de Microsoft Biztalk. Quatre gars de chez nous et une équipe de Microsoft le développaient et le faisaient tourner en production. Ils avaient tous l'air de travailler très dur. On les voyait souvent travailler tard dans la nuit et pendant le weekend.

Chacun laissait tomber ce qu'il était en train de faire si un problème survenait en production, ils se réunissaient souvent autour du bureau d'un type, faisaient des suggestions pour régler ce qui n'allait pas, ou pour réparer quelque chose. L'activité était permanente, et tout le monde pouvait voir non seulement qu'il y avait un véritable esprit d'équipe, mais que tout le monde travaillait très très dur.

L'équipe chargée de la TV numérique était tout à fait différente. Le code avait été, en majorité, écrit par une seule personne que l'on appellera Dave. Il était développeur de maintenance junior dans l'équipe. Au départ, j'ai eu beaucoup de difficultés à comprendre le code. Au lieu d'une seule longue procédure dans un endroit précis où tout le travail se serait effectué, il y avait des tas de petites classes et méthodes avec seulement quelques lignes de code. Plusieurs collègues se plaignirent que Dave rendait les choses extrêmement compliquées. Mais Dave me prit sous son aile et me conseilla de lire quelques livres sur la programmation orientée objet. Il m'apprit les patrons de conception, les principes SOLID, et les tests unitaires. Le code commença alors à avoir du sens, et plus je travaillais dessus plus

j’appréciais l’élégance de sa conception. Il ne posait pas de problème en phase de production, il ronronnait dans son coin et faisait le boulot. Il était assez facile d’opérer des modifications dans le code, du coup l’implémentation de nouvelles fonctionnalités se faisait souvent sans aucun problème. Les tests unitaires permettaient de trouver la plupart des bugs avant la mise en production.

Le résultat de tout cela est que nous n’avions pas l’air de travailler très dur du tout. Je rentrais chez moi à 18h30, je ne travaillais jamais pendant les weekends, nous ne passions pas des heures attroupés autour du bureau de quelqu’un d’autre pour deviner le problème avec un quelconque système planté en production. De l’extérieur, notre tâche devait sembler beaucoup plus facile que celle des gens de la TV analogique. En vérité, les exigences étaient très similaires, mais nous avions un logiciel mieux conçu et implémenté, et une meilleure infrastructure de développement, notamment les tests unitaires.

La direction fit savoir que des augmentations seraient accordées sur la base de nos performances. Quand ce fut mon tour de parler au directeur, il expliqua qu’il était normal que les augmentations soient accordées à ceux qui travaillaient vraiment dur, et que l’entreprise ne semblait pas importer beaucoup à notre équipe, au contraire de ces héros qui lui consacraient leurs soirées et leurs weekends.

photo par nic’s events – CC BY-SA 2.0

Cette entreprise était l’un des rares laboratoires où l’on pouvait comparer directement les effets d’une bonne ou d’une mauvaise conception logicielle et le comportement d’une équipe. La plupart des organisations ne permettent pas une telle comparaison. Il est très difficile de dire si ce type, transpirant sang et eau, travaillant tard les soirs et weekends, constamment sur la brèche, fait preuve d’une grande volonté pour faire fonctionner un système vraiment très complexe, ou s’il est simplement nul. Sauf si vous pouvez vous permettre d’avoir deux ou plusieurs équipes concurrentes travaillant à résoudre le même problème, mais bon, personne ne fait ça, on ne saura donc jamais. Et au contraire, le type assis dans son coin, travaillant de 9 heures à 17 heures et qui semble passer beaucoup de temps à lire sur internet ? Est-il très compétent pour écrire un code stable et fiable, ou son boulot est-il plus facile que celui des autres ? Pour l’observateur moyen, le premier travaille vraiment dur, pas le second. Travailler dur c’est bien, la paresse c’est mal, n’est-ce pas ?

Je dirais qu’avoir l’air de travailler dur est souvent un signe d’échec. Le développement logiciel est souvent mal fait

dans un environnement sous pression et dans lequel on est souvent interrompu. Ce n'est généralement pas une bonne idée de travailler de longues heures. Quelquefois, la meilleure façon de résoudre un problème est d'arrêter d'y penser, d'aller prendre l'air, ou encore mieux, de prendre une bonne nuit de sommeil et de laisser faire notre subconscient. Un de mes livres favoris est « *A Mathematician's Apology* » (traduit sous le titre *L'apologie d'un mathématicien)* par G. H. Hardy, un des mathématiciens anglais les plus importants du XX^{e} siècle. Il y décrit sa routine : quelques heures de travail le matin suivies par un après-midi à regarder le cricket. Il dit qu'il est inutile et contre-productif d'effectuer un travail mental intensif plus de quatre heures par jour.

photo par sfllaw – CC BY-SA 2.0

J'aimerais dire aux *manageurs* de juger les gens en regardant leurs résultats, leurs logiciels qui tournent bien, et non en regardant si les programmeurs ont l'air de travailler dur. C'est contre-intuitif, mais il est sans doute préférable de ne pas vous assoir tout près de vos développeurs, vous pourrez ainsi avoir une meilleure idée de ce qu'ils ont produit, sans être affecté par des indicateurs conventionnels ou intuitifs. Le travail à distance est particulièrement bénéfique ; vous

devez apprécier vos employés pour leur travail, plutôt que par la solution de facilité qui consiste à les regarder assis à leur bureau 8 heures par jour, martelant de façon lancinante sur leur IDE, ou se pressant autour du bureau des autres pour offrir des suggestions « utiles ».