« Dette technique » lors de l'écriture de logiciels

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 25 août 2009. Dernière mise à jour le 9 mai 2012

https://www.bortzmeyer.org/dette-technique.html

Les développeurs de logiciels, lorsqu'ils suivent les cours d'un établissement d'enseignement sérieux, apprennent un certain nombre de bonnes pratiques qui, si elles étaient systématiquement utilisées lorsqu'ils programment plus tard, dans le monde réel, mènerait à du logiciel de bien meilleure qualité. Au lieu de cela, les logiciels contiennent des bogues, parfois énormes, des failles de sécurité et sont souvent très difficiles à maintenir par les successeurs des premiers programmeurs.

Pourquoi les bonnes pratiques ne sont-elles pas davantage utilisées ? Les programmeurs mettent souvent en avant le manque de temps : à l'Université, on a des heures ou des jours pour programmer un crible d'Ératosthène ou une suite de Fibonacci avec tous ses détails, alors que, dans le monde réel, le temps manque, le chef réclame que le logiciel soit livré à temps, on prend donc des raccourcis et tant pis pour la qualité.

Ont-ils raison ou tort de « bâcler » le travail ? Le débat se focalise souvent en des termes binaires, le camp des « vrais programmeurs qui codent » accuse celui des experts d'irréalisme, celui des experts accuse les « bricoleurs » de mal faire le travail. Par exemple, dans une excellente discussion <http://serverfault.com/questions/28915/advice-and-tips-for-a-junior-sysadmin-straight-out-of-col sur "*Server Fault*" <https://www.bortzmeyer.org/server-fault.html>, Laura Thomas écrit, en défense de l'approche « vite fait et pas parfaitement fait », « "*Another of my favorite mottos is "Don't let perfect get in the way of better." Sometimes people struggling for elegance, simplicity, or the "right way" will refuse improvement because it isn't good enough.*" » alors que Ben Dunlap répond « "*Filthy scripts have a tendency to punish your co-workers and successors, and to fail when unexpected circumstances arise, which they always do. So do get the job done, but find the right balance between expedience and ideals. Those ideals exist for a reason. ;-)*" ».

L'article de Martin Fowler, "*Technical debt*" <http://martinfowler.com/bliki/TechnicalDebt.html> (« dette technique »), évite au contraire de parler en terme de mal ou de bien, juste de compromis. Ne pas suivre complètement les pratiques idéales équivaut à s'endetter : il faudra payer un jour (rien n'est gratuit) mais, dans certains cas, s'endetter reste quand même une approche raisonnable. Il faut juste en être conscient.

Un autre excellent article sur ce même thème est celui de Anthony Ferrara, "*The Power of Technical Debt*" <http://blog.ircmaxell.com/2012/03/power-of-technical-debt.html>, où il compare la dette technique du programmeur à divers types de dettes qu'on trouve dans le monde financier (attention, les exemples sont plutôt empruntés au monde états-unien, les dettes ne fonctionnent pas forcément pareil dans les autres pays), analysant à chaque fois si cela peut être une bonne ou une mauvaise dette.