NOHAR

Geste de savoir

Qu'est-ce que le Software Craftsmanship ?

27 mai 2021

Table des matières

Il y a quelques années, @SpaceFox a publié dans cette tribune un billet à propos du célèbre texte Software Disenchantment . Dans les commentaires de ce billet, une notion particulière a retenu mon attention: celle de *Software Craftsmanship*. Je me rappelle à l'époque avoir cherché rapidement des ressources pour comprendre ce mouvement, mais n'avoir pas creusé beaucoup plus loin que la Page Wikipedia .

Depuis quelques mois, je traverse une période de grande sérénité professionnelle. En effet, **j'ai fini de rembourser ma dette technique**[1]: mon code est testé avec un coverage important, ses tests tournent dans une CI qui automatise sa publication, il est déployé au moyen de commandes simplissimes (en deux mots : `make deploy`), et il est entièrement observable au moyen de métriques et de logs que je peux exploiter avec une facilité enfantine dans Grafana. Tout est sous contrôle.

Pour *améliorer encore* le système sur lequel je travaille, je dois maintenant descendre beaucoup plus profond que cela: je dois *améliorer la façon dont je travaille dessus*, de manière à ne plus avoir besoin de contracter à nouveau une dette technique, et produire à l'avenir **le meilleur travail dont je suis capable** en livrant le plus efficacement possible les fonctionnalités dont mes utilisateurs ont besoin. C'est dans cet état d'esprit que j'ai entrepris d'étudier une poignée de livres liés au *Software Craftsmanship*.

Ces lectures m'ont permis de revivre toutes les expériences que j'ai vécues dans ma carrière avec un œil nouveau, et d'en tirer de nombreux enseignements pour aborder les années qu'il me reste à exercer ce métier.

Dans ce billet, je vais résumer et partager avec vous quelques concepts importants que mes réflexions m'ont amené à revoir.

1. Je vois déjà des sourcils se lever. Sachez que ce n'est **pas** un exploit. Le projet sur lequel je travaille est un projet "*green field*", démarré depuis 0 il y a un an et demi. Ce n'est pas tant le fait que j'ai réussi à la rembourser entièrement, que le fait que cette dette *ait existé* qui devrait retenir votre attention.

1. Que signifie « craftsmanship » ?

1.1. Le « Craftsman » de Richard Sennett

La notion de *craftsmanship* sur laquelle ce billet repose est celle de Richard Sennett dans son livre *The Craftsman*, et celle-ci est particulièrement large: c'est le fait de **considérer le travail bien fait comme une fin en soi**. D'aucuns seraient tentés de traduire le terme en français, en partant du principe qu'un *craft* est un *artisanat*. Ce serait faire fausse route ici.

Si vous tenez vraiment à traduire le terme *craftsmanship*, alors le mot le plus proche que j'aie à vous proposer dans notre langue, c'est *professionalisme*. Mais ce n'est pas tout à fait ça non plus.

Le *Craftsman* de Richard Sennet désigne aussi bien :

— le maître ébéniste qui enseigne le métier à ses apprentis dans son atelier,
— la technicienne de laboratoire qui fait le choix de rester tard le soir pour disséquer des cobayes qui n'auraient pas dû décéder et comprendre si c'est la procédure qui est mauvaise ou si elle a été mal appliquée, alors qu'elle pourrait se contenter du rapport qu'elle a déjà envoyé à sa hiérarchie,
— la violoniste qui répète ses gammes encore et encore, heure après heure, jour après jour, année après année, afin de faire pénétrer et garder dans ses mains la justesse du ton et la finesse du jeu que l'on attend d'elle,
— les développeurs d'un projet Open Source comme le noyau Linux, qui échangent, affûtent leurs compétences et leurs pratiques en partageant leurs découvertes avec une communauté dont le travail collectif garantit une qualité logicielle avec laquelle peu d'entreprises peuvent rivaliser.

Il ne s'agit donc pas que des métiers manuels ou artisanaux. Richard Sennett parle aussi bien dans son livre des exemples que je viens de donner que des docteurs et des infirmières, des architectes et des urbanistes, des ingénieurs, des cuisinières, des souffleuses de verre, des briquetiers ou des forgerons. À son sens, l'éducation des enfants est également un *craft*.

Le livre s'attarde entre autres sur:

— l'attitude de ces professionnels et leur fonction sociale,
— leur relation avec leur atelier (ou plus généralement leur lieu de travail) au travers de l'histoire,
— l'entraînement, la spécialisation, le fonctionnement de leurs "mains",
— les écueils du perfectionnisme,

Il défend sur la question un point de vue *matérialiste* et *pragmatique*.

Tout peut se déduire logiquement de cette idée centrale: **le travail bien fait est une fin en soi**.

1.2. Ce n'est "qu'un" mot

Ce n'est qu'une idée. Une définition particulière du *professionalisme*. Ce n'est pas un culte. C'est juste une opinion.

Mais c'est une opinion que je partage.

1.3. Passion et discipline

Il y a vingt ans, alors que je traversais une tempête d'incertitudes à propos de mes études et mon avenir professionnel, mon père m'a sorti cet aphorisme:

«*Pour faire ce que tu aimes, tu dois aimer ce que tu fais.*»

Cette phrase m'a très longtemps hanté. J'ai toujours senti qu'elle énonçait une vérité bien plus profonde que ne le suggère sa formulation triviale. Mon père était cardiologue et il exerçait son métier en véritable *craftsman*. Il se tenait à jour de l'avancée des recherches, participait à des congrès, prenait souvent du temps jusque tard le soir pour rédiger sa correspondance avec ses consœurs et confrères, et avait à cœur d'agir, à tout instant, en professionnel dans l'intérêt de ses patients. Il suffisait que l'on déclare à table "j'ai comme un point de côté depuis hier soir" pour voir immédiatement son regard changer: c'était le père de famille qui *switchait* en mode médecin. En bref, mon père était quelqu'un de **passionné** par son métier, et cette phrase venant de lui n'avait rien d'anodin.

De loin, cette phrase fait écho à une notion plutôt à la mode en psychologie du travail: le *sens au travail*. En résumé, on ne peut s'épanouir au travail qu'à partir du moment où celui-ci a *un sens pour nous*. Sans trop entrer dans le détail de cette notion complexe en psychologie, cela signifie notamment que notre travail est aligné avec nos intérêts et nos valeurs professionnels. Et c'est bien naturel: pour s'infliger ce travail quotidiennement pendant la plupart de la journée, la plupart de la semaine, la plupart de notre vie, il est indispensable qu'il ait pour nous *un sens* qui dépasse la seule compensation financière. Si certains éléments (une relation toxique avec un collaborateur, des spécifications ou des normes complètement débiles, des processus qui accaparent tout notre temps, le manque de pouvoir pour arranger les choses…) viennent à se dresser entre nous et notre vision d'un travail utile et bien fait, alors celui-ci perd son sens et peut devenir une source de souffrances.

Dans le cas du *craftsmanship*, cela va encore plus loin que le sens du travail : notre métier est *bien plus qu'un travail* ; c'est une passion. Lorsque l'on est passionné par le développement, on ne compte pas les heures à faire de la veille: *ça nous intéresse à la base*, que l'on soit au bureau ou non, de découvrir des nouvelles techniques et technologies, de partager avec diverses communautés, d'améliorer nos pratiques. On se préoccupe et on en prend soin de notre code, car il y va de notre réputation, et parce qu'à nos yeux les problèmes que nous résolvons sont *passionnants*. Ce métier coule dans nos veines.

Cependant, **être passionné est une condition *nécessaire*, mais pas *suffisante*** pour se comporter en *craftsman*, car le travail bien fait exige que notre passion soit canalisée par une **discipline** tout aussi grande. Cette discipline, on l'exhibe notamment lorsque l'on s'entraîne en codant sur des projets-jouets pour nous former à une nouvelle *stack* ou de nouveaux outils, lorsque l'on ne fait pas l'impasse sur la qualité de notre code ni sur ses tests malgré la pression et le stress, ou encore, lorsque l'on ne joue pas au héros en revoyant nos estimations à la baisse pour

éviter un conflit avec notre manager, mais que l'on négocie avec lui des solutions alternatives pour livrer le meilleur travail possible à nos clients, à la date voulue.

Cette discipline, c'est *celle qui nous impose également de chercher un équilibre sain entre travail et vie privée*, sans lequel on ne peut pas rester performant sur le long terme.

Si passion et discipline sont nécessaires, c'est parce qu'ils se complètent: la passion sans discipline résulte en un travail d'amateur auquel on ne peut pas se fier, mais la discipline nécessaire à devenir un excellent professionnel ne peut être atteinte que du moment que nous sommes passionné pour ce que l'on fait. Nous exerçons un travail hautement qualifié: pour acquérir nos compétences, il faut les vouloir! Pour écrire du code toute sa carrière, *il faut aimer ça*.

Ramené à l'aphorisme de mon père: pour «faire ce que l'on aime», un travail dont on tire du sens et de la fierté, on doit «aimer ce que l'on fait», en nourrissant pour notre métier une passion telle qu'elle contrebalance le temps et les efforts que l'on concède à l'affûtage de notre discipline professionnelle.

Ce sont essentiellement cette passion et cette discipline que nous transmettons aux jeunes professionnels qu'il nous est donné d'encadrer et de former pendant notre carrière.

1.4. Pragmatisme

Le pragmatisme dont il est question ici s'oppose aux deux principaux écueils qui font obstacle au travail bien fait: le **dogmatisme** d'une part, et l'**obsession perfectionniste** d'autre part.

Le **dogmatisme**, c'est le fait de camper sur des vérités que l'on ne remet jamais en question, et que l'on considère comme absolues, à la manière de dogmes religieux ou idéologiques. Chez nous, développeurs, il se révèle au travers de phrases comme:

— *Tout nouveau système doit être architecturé en microservices et pas autrement*,
— *Les variables globales, c'est le mal*,
— *[C#/Java/Go/Python/C++/Rust/…] est un bien meilleur langage que [C#/Java/Go/Python/C++/Rust/…]*,
— *Si tu ne fais pas de [TDD/pair programming/…], alors tu ne peux pas faire du bon travail*.

Le danger, ici, n'est pas tellement de penser "des choses fausses".

Le danger, c'est de les penser pour les "mauvaises raisons".

Le pragmatisme nous impose de raisonner **dans un contexte**, en ayant l'esprit assez ouvert pour remettre en question nos croyances établies, de manière à ce que lorsque nous nous retrouvons face à une manière de faire objectivement meilleure que la nôtre, nous soyons *en mesure de l'adopter*. Aussi, si le mouvement du *software craftsmanship* promeut à l'heure actuelle les pratiques de la méthodologie XP (*pair programming*, TDD, *simple design*, …) c'est parce qu'à l'heure actuelle, la majorité des personnes qui s'identifient avec ce mouvement ne *demandent qu'à faire mieux* qu'avec ces pratiques dont ils font la promotion: si demain on nous montre une façon plus efficace d'écrire du code fiable que le TDD sur un projet donné, alors l'attitude *professionnelle* est de s'y essayer et d'être en mesure de l'adopter sur le projet si c'est utile.

L'**obsession perfectionniste**, de son côté, est plus traître. On peut facilement la confondre avec la passion, ou le goût du travail bien fait.

On est perfectionniste quand on ne sait pas quand il faut s'arrêter.

Ou quand on refuse de travailler avec du code *legacy*, parce qu'on n'a pas envie d'être tentés de le récrire entièrement: c'est perdre de vue l'intérêt de nos clients car la quantité d'effort que l'on dépenserait à récrire ce code qui existe déjà, pour ne rien faire de plus ni de mieux, n'est pas justifiable du point de vue de l'utilisateur. Comme le dit un dicton célèbre: *le mieux est l'ennemi du bien.*

L'une des façons dont le pragmatisme et la discipline s'opposent au perfectionnisme dans notre métier est par exemple la *règle du boy scout*: «*il faut toujours laisser le code un peu plus propre qu'on ne l'a trouvé*»[2]. Cela s'applique particulièrement au code *legacy*: si cette masse de code est moche et pas testée, plutôt que de la reconcevoir entièrement à partir de zéro, notre challenge est déjà d'extraire la partie du code *avec laquelle nous avons besoin d'interagir dans l'immédiat* pour la couvrir de tests *et* la refactoriser, comme une partie intégrante du ticket sur lequel nous travaillons. C'est peu, mais c'est déjà quelque chose, et faire *quelque chose* qui va dans le bon sens est toujours mieux que ne rien faire du tout, car ces petites touches d'amélioration s'accumulent avec le temps. Les chances que ce code ne soit pas testé *car il n'a pas été écrit pour être testable* sont relativement élevées, mais est-ce vraiment pour nous laisser vaincre d'avance par ce genre de défis que nous faisons ce boulot?

2. Se regarder dans le miroir

> Many Agile projects are now, steadily and iteratively, producing crap code.
>
> *Sandro Mancuso*[4]

Aucun développeur ne se lève le matin en se disant : «Quelle belle journée pour écrire plein de bugs et saloper mon projet !». Aucun manager ne se lève non plus le matin en se disant : «Quelle belle journée pour emmerder le monde et reprocher à mon équipe de ne pas avoir réussi l'impossible !». Et pourtant, les développeurs qui écrivent des bugs, les projets qui finissent salopés, les managers qui emmerdent le monde et qui demandent l'impossible à leurs équipes, il suffit de taper dans le premier arbre venu pour en être recouvert!

Cette situation est absolument regrettable. Pour espérer la changer, il nous faut la comprendre, et c'est une situation complexe. Si les projets finissent salopés, ce n'est pas *uniquement* la faute des développeurs, ni *uniquement* celle de leur hiérarchie. Les responsabilités sont partagées, et il importe que **tout le monde** prenne le temps de réaliser un examen approfondi dans le miroir. Ce n'est que de cette manière que chacun peut se mettre à assumer ses responsabilités et contribuer à relever la barre.

Au minimum, il importe que **nous autres**, développeurs, réalisions cet examen de conscience, afin de cesser de gesticuler comme des amateurs, et garder en tout temps la satisfaction et l'assurance de nous être comportés en professionnels irréprochables, même si cela veut dire mettre fin à une collaboration, ou encore plus extrême, de tenir face à un employeur qui nous

2. Chez les scouts la vraie règle est «il faut toujours laisser le camp un peu plus propre qu'on ne l'a trouvé». Et cela fait écho au discours d'adieu de Baden-Powell : «puissiez-vous laisser le monde un peu meilleur que vous ne l'avez trouvé».

menace de nous licencier[3]. De toute façon, pour paraphraser Sandro Mancuso[4], dans le contexte économique actuel, «seuls les mauvais développeurs ont peur pour leur job».

2.1. Sommes-nous traités comme des professionnels ?

Imaginons que vous soyez admis aux urgences suite à un accident. Disons, juste pour l'exemple, que vous vous êtes fait une entaille profonde qui nécessite quelques points de suture. Est-ce que cela vous viendrait à l'esprit de d'ordonner aux infirmières d'arrêter de se laver les mains pour vous recoudre le plus vite possible? Est-ce que vous ne trouveriez pas ça **encore plus dingue** si c'était le médecin urgentiste, plutôt que vous, qui demandait aux infirmières d'arrêter de perdre du temps et de se dispenser de stériliser votre blessure?

Ou encore imaginons que vous fassiez construire une maison. Est-ce que vous trouveriez normal de marchander le bout de gras avec le conducteur de chantier pour qu'il réduise son estimation des délais en sautant le temps de séchage de la dalle de béton qui constitue les fondations?

Non, bien sûr, personne ne ferait ça: les infirmières, les médecins, les conducteurs de travaux et les artisans qui construisent nos maisons sont **des professionnels**. Ils connaissent leur métier mieux que quiconque, et ce n'est certainement pas à nous de leur dire comment faire leur travail.

En tout cas ma mère était infirmière, et *personne ne lui parlait sur ce ton.*

Comment expliquer, dans ce cas, que j'aie déjà eu un manager qui m'a demandé, un nombre incalculable de fois:

— De ne pas perdre de temps à "faire propre et refactoriser" parce que c'est "pour la semaine prochaine, pas dans un mois"?
— De ré-estimer une tâche en excluant l'écriture des tests?[5]
— De "faire au plus vite", comme si je ne faisais pas déjà de mon mieux d'ordinaire pour livrer le plus tôt possible?

Comment expliquer que j'aie eu un N+2 qui s'enorgueillissait d'être "un pragmatique dans l'âme", et qui se mêlait des outils dont se servent les développeurs ("ok je veux bien que vous utilisiez gitlab à condition que toutes les équipes l'utilisent"), et qui nous **interdisait** ouvertement de pratiquer le TDD ou le *pair programming* ?

Tout cela porte un nom: c'est du micro-management, et si cela ne devait signifier qu'une seule chose, ce serait que **nos supérieurs ne nous traitent pas comme des professionnels**.

— Un manager ou un client n'a pas à imposer leurs outils de travail à des développeurs. Ce sont **nos** outils. C'est **nous** que ça concerne. C'est à nous d'utiliser ceux grâce auxquels nous sommes les plus efficaces.

3. Il faut prendre ce passage avec des pincettes. **Tout le monde** n'est pas en mesure de tenir tête à un employeur abusif. Par contre, *subir* ce genre de violence et s'y plier de la part d'un employeur, c'est l'assurance que cela recommencera, et donc que vous vous trouvez dans un poste particulièrement toxique que vous n'allez pas pouvoir garder longtemps. Ou alors au détriment de votre santé.

4. *The Software Craftsman : Professionalism, Pragmatism, Pride*, Sandro Mancuso, éditions Prentice Hall.

5. Comme je regrette aujourd'hui de ne pas avoir pratiqué le TDD plus tôt ! J'aurais adoré voir sa tête, quand, avec tout le sérieux et la bonne foi du monde, je lui aurais annoncé 20% de délai supplémentaire pour finir la fonctionnalité *sans écrire les tests.*

— Un manager ou un client n'a pas à imposer ni interdire à des développeurs d'adopter une pratique ou une autre: *cela ne le regarde pas*, ce sont les développeurs qui font le boulot. Nous sommes responsables de nos pratiques. C'est à nous de choisir celles qui nous permettent de *bien* faire notre boulot.
— Un manager ou un client n'a pas à imposer à ses développeurs de court-circuiter les tests qui garantissent la qualité du logiciel : c'est notre responsabilité, pas la sienne, et c'est un **manque de professionnalisme** vis-à-vis de notre équipe, parce que quand le logiciel pétera en production[6], ce sera à nous d'assumer en écrivant quand même, mais cette fois dans l'urgence, les tests sur lesquels il nous avait demandé de faire l'impasse. **Ce sont ceux qui assument quand ça foire qui sont responsables.**

Un professionnel ne se plie jamais à ce genre de caprices. Le contraire serait aussi *irresponsable* de sa part qu'il serait *criminel* pour un médecin d'accepter de ne pas se laver les mains entre deux patients. La qualité logicielle, en dépit de ce que peuvent nous dire certains managers peu scrupuleux, n'est *pas une option*: elle est systématiquement attendue, coûte que coûte, par les utilisateurs. Un manager qui exige cela de nous est en train de *tirer une balle dans le pied de toute la boîte*, et nous avons le devoir de ne pas le laisser nous nuire collectivement: ni à nos clients/utilisateurs, ni à nous, ni à lui.

Dans une telle situation, il n'y a qu'une seule chose à faire: *ne pas céder*. Rester fidèles à notre discipline, et faire comprendre à notre manager qu'il dépasse une limite et que ses demandes ne sont ni raisonnables, ni acceptables. En d'autres termes, lui faire comprendre que *ce qu'il demande n'arrivera jamais*, et lui proposer soit de repousser ses *deadlines*, soit de négocier ensemble, avec le client, un scope réaliste. Ou encore, lui répondre un **non** ferme et définitif, puis chercher avec lui des alternatives. Nous partageons un même but: celui de satisfaire le même client.

… Malheureusement, face à ces demandes surréalistes, il nous arrive parfois de pousser un soupir et céder: «ok, je vais essayer», et c'est alors le début de la fin. Mais là, ce n'est plus seulement le manager qui est en cause.

2.2. Nous comportons-nous comme des professionnels ?

Il y a long à dire sur ce «je vais essayer» que l'on a tous répondu au moins une fois dans notre carrière. Des gens comme 'Uncle Bob' Martin[7] et Sandro Mancuso[4] l'ont déjà fait bien mieux que je n'en suis capable. Je me bornerai donc à dire que ce «je vais essayer» est la chose **la moins professionnelle** à répondre à un manager qui nous met la pression, nous cajole, nous *bullshite* en nous promettant une refacto après la release, et essaye tous les stratagèmes possibles pour faire entrer nos deux mois d'estimations dans un délai de trois semaines.

Cette réponse, c'est celle de quelqu'un qui joue les héros. *Je vais éclater ce truc en deux fois moins de temps que prévu, ça va sauver le projet et ça sera bon pour mon avancement.* Foutaises!

Ce que cette réponse envoie comme signal au manager ou au client, c'est qu'on en avait "gardé sous la pédale" jusqu'à maintenant, et que l'on ne travaille pas déjà, d'ordinaire, en faisant de notre mieux. Ce par quoi elle se traduit dans la réalité, c'est *une gigantesque pression*. Ce

6. Et s'il vous manque les tests, vous le savez comme moi, ça va se casser la figure en production.
7. *The Clean Coder : A Code of Conduct for Professional Programmers*, Robert C. Martin, Pearson Education

sont des heures supplémentaires à la pelle que nous nous infligeons, à travailler dans un état de fatigue trop avancé pour que l'on puisse écrire autre chose que du code pourri.

Et si, par un jour d'éclipse, on arrivait à tenir cette deadline impossible, vous croyez que le manager nous donnerait une claque dans le dos, et qu'il demanderait à toute la boîte de s'arrêter un instant pour nous applaudir ? *Pour avoir réussi à faire notre travail, pour une fois?*

Est-ce que tenir une *deadline* ne devrait pas *aller de soi*, pour vous? Si seulement *on vous en donnait les moyens?*

Prenons le temps de méditer sur ces mots de maître Yoda :

FIGURE 2.1. – Do, or do not. There is no try.

Si nous nous *engageons* sur une date, alors nous refusons de *ne pas* la tenir. Dans ce cas, il vaut mieux pour tout le monde que nous ne nous engagions *jamais* sur un résultat que nous ne sommes pas **absolument certains** de tenir. Quand nous nous engageons, nous livrons. C'est une question de fiabilité.

Si nous savons que *cette tâche ne peut pas tenir* dans le délai que l'on nous demande, c'est qu'elle ne *peut pas*, le prix à payer pour la faire tenir dedans, il est *de notre poche*, et nous ne voulons pas nous tirer cette balle dans le pied.

Soyons honnêtes avec nous-mêmes deux minutes.

— Si vous embauchez un avocat pour qu'il vous aide à régler vos problèmes légaux, quelle tête feriez-vous s'il vous répondait «OK, mais en plus de mes honoraires je veux que tu m'achètes un ou deux livres de droit.» ?
— Si vous subissiez une opération à cœur ouvert, que diriez-vous si le chirurgien se mettait à crier des « *WHAT THE FUCK ?!* » alors qu'il vient de vous ouvrir, ou encore se mettre à râler, peut-être en balançant des ustensiles au travers du bloc opératoire, après sa hiérarchie qui fait n'importe quoi, après la deadline à laquelle il doit avoir fini de vous opérer, et après l'état de votre valve mitrale qu'il faudrait retaper entièrement après l'avoir "cramée au lance-flammes" parce qu'il n'y a plus rien à en tirer?
— Si vous embauchez un couvreur pour qu'il vienne refaire votre toiture, est-ce que vous accepteriez qu'il vous en recouvre un bout avec les dernières tuiles à la mode (qui ne vont pas du tout avec le reste de la toiture) "parce qu'il a envie d'essayer la techno" ?

Alors pourquoi les gens devraient-ils accepter ce genre de comportement de la part des développeurs ?

Que notre employeur nous forme sur les technologies dans lesquelles il a investi, c'est *normal*. Mais est-ce que ce sont sur *ces* technologies que nous voulons nous former, *nous*?

Auriez-vous envie de nous retrouver dans une situation où vous devriez demander à votre employeur de vous former *pour que vous puissiez partir de chez lui*? Si vous voulez quitter cet employeur, ce serait quand même *dommage* de dépendre de lui pour ça, vous ne pensez pas?

C'est pour ça que nous nous tenons à jour. C'est pour piloter notre carrière.

Chaque fois que nous nous reconnaissons dans les exemples que je viens de citer, nous nous souvenons d'une situation où nous ne sommes pas comportés en professionnels. Admettons que **nous ne nous serions pas traités nous-mêmes comme des professionnels** à la place d'un client ou d'un manager.

Alors, quand certains supérieurs se mettent à nous micro-manager, c'est aussi en partie **parce qu'on leur tend le bâton pour se faire battre**: si vous n'entrez pas dans le détail à parler de *pair programming* avec votre manager (que cela ne concerne pas), il n'y a aucune chance qu'il vous interdise de le pratiquer et ça devient **votre problème**, tant que ça vous aide à livrer du bon travail en temps et en heure.

De la même manière, si vous considérez que les tests font partie du ticket et que leur existence va tellement de soi qu'ils ne méritent même pas d'être un item dans une *checklist*, alors personne ne viendra vous dire de faire l'impasse dessus ou de les remettre à plus tard. Pourquoi tenter le diable?

Si nous ne voulons pas être micro-managés, alors *comportons-nous comme des pros* et **gardons ce niveau de détails pour nous**. Nous-mêmes, plus que quiconque, comprenons les bénéfices de l'encapsulation dans notre code: c'est *la même chose* ici.

2.3. La "gueule de bois Agile"

J'ai une fois aidé une société pour laquelle je travaillais à adopter une méthodologie Agile. Lorsque j'ai démissionné un an et demi plus tard, j'en étais arrivé à **haïr SCRUM**.

J'étais perplexe: qu'est-ce qui a bien pu foirer?

On avait une pièce dédiée avec un grand tableau blanc sur lequel on faisait bouger des étiquettes colorées. On faisait religieusement notre standup tous les jours. On faisait tout *by the book*, comme il fallait. On faisait tous de notre mieux pour limiter les réunions au strict nécessaire, pour que nos communications soient efficaces.

Et on s'enlisait quand même.

Les échecs s'accumulaient. On ratait des deadlines, on continuait à prier pendant les déploiements. L'environnement de développement était devenu un marécage radioactif truffé de modifications manuelles. On remplaçait le code *legacy* par du code qui devenait *legacy* encore plus rapidement. Le code mort s'accumulait. On avait le goût de l'échec dans la bouche en nous brossant les dents le matin.

Alors on les serrait. On bougeait encore plus frénétiquement nos étiquettes colorées sur notre grand tableau blanc. Mes collègues démissionnaient les uns derrière les autres. SCRUM nous avait trahis en rendant les choses encore pires.

C'est ce que l'on appelle la "gueule de bois Agile" (*the Agile hangover*[4]).

Ce qui a foiré, c'est que SCRUM n'est qu'une méthodologie, pas un remède miracle et qu'Agile ne se résume pas à des méthodologies: Agile est une philosophie. On ne *fait pas* "du Agile". On *est* Agile ou on ne l'est pas.

Ce qui a foiré, c'est que pour qu'une méthodologique comme SCRUM fonctionne, **il y a des prérequis**. Et l'un de ces prérequis est l'excellence de l'équipe: SCRUM fonctionne à merveille quand chaque développeur de l'équipe est traité comme professionnel et se comporte en tant que tel, quand l'équipe *a le pouvoir* (et *l'exerce dans la pratique*) de s'améliorer, de s'entraider, de s'engager. Si le planning se conclut par un "*on va essayer*" de la part de l'équipe, le sprint est **mort-né**.

Si les développeurs ne sont pas responsables de leurs outils et de leurs pratiques, alors :

— ils ne peuvent plus garantir la qualité du code,
— il ne peuvent plus s'engager sur des deadlines,
— ils ne peuvent plus améliorer le code en continu et maintenir son hygiène,

… et ils livrent, itérativement et progressivement, du code de merde.

Autrement dit: **avant de jeter du SCRUM sur nos problèmes, commençons par nous comporter en professionnels**, sans quoi SCRUM ne fera qu'empirer les choses.

Il y aurait encore des tas de choses sur lesquelles je pourrais écrire. Notamment sur ce que cela implique en termes de recrutement, sur les pratiques que l'on peut adopter…

Mais vous, qu'est-ce que vous pensez de tout ça, déjà?