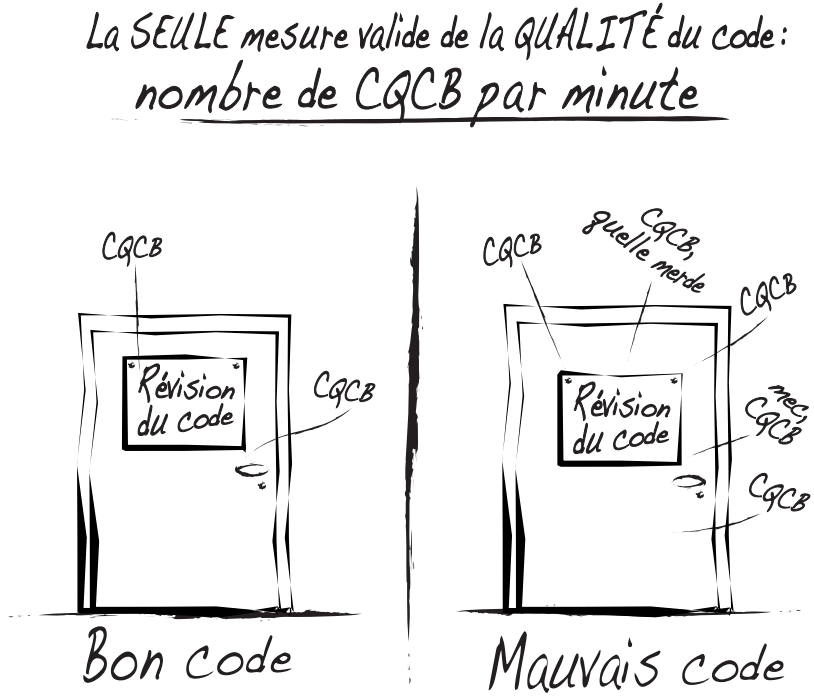
Coder proprement

Robert C. Martin

Michael C. Feathers Timothy R. Ottinger
Jeffrey J. Langr Brett L. Schuchert
James W. Grenning Kevin Dean Wampler
Object Mentor Inc.

Introduction



CQCB : C'est quoi ce bordel

Reproduction et adaptation avec l'aimable autorisation de Thom Holwerda (**http://www.osnews.com/story/19266/WTFs_m**). © 2008 Focus Shift

Quelle porte ouvre sur votre code ? Quelle porte ouvre sur votre équipe ou votre entreprise ? Pourquoi êtes-vous dans cette pièce ? S'agit-il simplement d'une révision normale du code ou avez-vous découvert tout un ensemble de problèmes désastreux peu après le lancement ? Procédez-vous à un débogage en urgence, en plongeant dans du code que vous pensiez opérationnel ? Les clients partent-ils en masse et les managers vous surveillent-ils ? Comment pouvez-vous être sûr que vous serez derrière la *bonne* porte lorsque les choses iront mal ? Les réponses tiennent en une seule : l'*art du métier.*

La maîtrise de l'art du métier englobe deux parties : connaissances et travail. Vous devez acquérir les connaissances concernant les principes, les motifs, les pratiques et les heuristiques connus de l'artisan, et vous devez également polir ces connaissances avec vos doigts, vos yeux et vos tripes, en travaillant dur et en pratiquant.

Je peux vous enseigner la physique qui permet de rouler à vélo. Il est vrai que les mathématiques classiques sont relativement simples. Gravité, frottements, moment angulaire, centre d'inertie, etc. peuvent être expliqués en moins d'une page d'équations. Grâce à cette formule, je peux prouver qu'il vous est possible de rouler à vélo et vous apporter toutes les connaissances dont vous avez besoin pour ce faire. Néanmoins, vous tomberez immanquablement la première fois que vous grimperez sur la bicyclette.

Écrire du code n'est pas si différent. Nous pourrions rédiger tous les bons principes d'écriture d'un code propre et vous faire confiance pour réaliser le travail (autrement dit, vous laisser tomber lorsque vous monterez sur le vélo), mais quelle sorte de professeurs serions-nous alors et quelle sorte d'étudiant seriez-vous ?

Ce n'est pas l'orientation que nous donnons à ce livre.

Apprendre à écrire du code propre est un *travail difficile*. Cela ne se limite pas à connaître des principes et des motifs. Vous devez *transpirer*. Vous devez pratiquer et constater vos échecs. Vous devez regarder d'autres personnes pratiquer et échouer. Vous devez les voir hésiter et revenir sur leurs pas. Vous devez les voir se tourmenter sur des décisions et payer le prix de leurs mauvais choix.

Vous devez être prêt à travailler dur au cours de la lecture de cet ouvrage. Il ne s'agit pas d'un livre que vous pourrez lire dans un avion et terminer avant d'atterrir. Il vous imposera de *travailler, dur*. Qu'est-ce qui vous attend ? Vous allez lire du code, beaucoup de code. Vous devrez réfléchir aux points positifs et aux points négatifs de ce code. Il vous sera demandé de nous suivre pendant que nous découpons des modules, pour ensuite les réunir à nouveau. Cela demandera du temps et des efforts, mais nous pensons que cela en vaut la peine.

Nous avons décomposé ce livre en trois parties. Les premiers chapitres décrivent les principes, les motifs et les pratiques employés dans l'écriture d'un code propre. Ils contiennent beaucoup de code et ne seront pas faciles à lire. Ils vous prépareront à la deuxième partie. Si vous refermez le livre après avoir lu la première partie, nous vous souhaitons bonne chance !

C'est dans la deuxième partie que se trouve le travail le plus difficile. Elle est constituée de plusieurs études de cas dont la complexité va croissant. Chaque étude de cas est un exercice de nettoyage : une base de code qui présente certains problèmes doit être transformée en une version soulagée de quelques problèmes. Dans cette section, le niveau de

détail est élevé. Vous devrez aller et venir entre le texte et les listings de code. Vous devrez analyser et comprendre le code sur lequel nous travaillons et suivre notre raisonnement lors de chaque modification effectuée. Trouvez du temps, car *cela demandera plusieurs jours*.

La troisième partie sera votre récompense. Son unique chapitre contient une liste d'heuristiques et d'indicateurs collectés lors de la création des études de cas. Pendant l'examen et le nettoyage du code dans les études de cas, nous avons documenté chaque raison de nos actions en tant qu'heuristique ou indicateurs. Nous avons essayé de comprendre nos propres réactions envers le code que nous lisions et modifiions. Nous avons fait tout notre possible pour consigner l'origine de nos sensations et de nos actes. Le résultat est une base de connaissances qui décrit notre manière de penser pendant que nous écrivons, lisons et nettoyons du code.

Cette base de connaissance restera d'un intérêt limité si vous ne faites pas l'effort d'examiner attentivement les études de cas de la deuxième partie de cet ouvrage. Dans ces études de cas, nous avons annoté consciencieusement chaque modification apportée, en ajoutant également des références vers les heuristiques. Ces références apparaissent entre crochets, par exemple [H22]. Cela vous permet de savoir dans quel *contexte* ces heuristiques ont été appliquées et écrites ! C'est non pas tant les heuristiques en soi qui ont de la valeur, mais *le lien entre ces heuristiques et chaque décision que nous avons prise pendant le nettoyage du code*.

Pour faciliter l'emploi de ces liens, nous avons ajouté à la fin du livre des références croisées qui indiquent le numéro de page de chaque référence d'heuristique. Vous pouvez les utiliser pour rechercher chaque contexte d'application d'une heuristique.

Si vous lisez la première et la troisième partie, en sautant les études de cas, vous n'aurez parcouru qu'un livre de plus sur la bonne écriture des logiciels. En revanche, si vous prenez le temps de travailler sur les études de cas, en suivant chaque petite étape, chaque décision, autrement dit en vous mettant à notre place et en vous forçant à réfléchir à notre manière, alors, vous comprendrez beaucoup mieux ces principes, motifs, pratiques et heuristiques. Ils ne seront plus alors des connaissances de "confort". Ils seront ancrés dans vos tripes, vos doigts et votre cœur. Ils feront partie de vous, de la même manière qu'un vélo devient une extension de votre volonté lorsque vous savez comment le conduire.