SQuORE et la qualité des développements logiciels

Boris Baldassari,
Université de Lille 3, Lille, France
boris.baldassari@gmail.com

1 Introduction

La qualité des développements logiciels est un sujet récurrent pour les praticiens et utilisateurs du logiciel. Objet d'intérêt et de discussions, elle est cependant régulièrement mal comprise ou dénigrée, et a souvent mauvaise presse auprès des développeurs ou chefs de projet. Aujourd'hui encore, l'assurance qualité est souvent considérée comme une activité coûteuse et un investissement hasardeux.

La non-qualité logicielle a un coût, pourtant. Ou plusieurs : ceux qui se voient et se chiffrent facilement, tel qu'un plantage en production, et ceux qui sont dissimulés dans la maintenance du produit, plus difficiles à estimer. Les premiers sont connus et impressionnants : l'explosion d'Ariane 5 (500M$ en pure perte), le crash du réseau AT&T (60M$ de perte de revenue, retracé jusqu'à un break manquant), ou la destruction du module Mars Climate Observer (durée des tests insuffisante, le robot s'écrase suite au remplissage de sa mémoire). Les seconds sont moins visibles mais plus courants, et portent un impact bien plus grand : chaque projet logiciel en porte sa part, et ils sont récurrents.

Dans cet article, nous proposons quelques pistes et définitions pour aider à mieux formaliser les attentes et objectifs de la qualité, et comment mettre en place un processus de mesure et d'analyse fiable avec l'outil d'évaluation de la qualité des développements SQuORE[1].

2 A propos de la qualité

Définir la qualité

Quelque soit le domaine considéré, donner une définition de la qualité qui soit généralement admise par tous est difficile. Appliquées au développement logiciel, les principales définitions de la qualité rencontrées dans la littérature sont la satisfaction du client[a], l'adéquation au besoin[b], et la conformité aux exigences[c]. Dans la pratique le concept varie principalement en fonction du rôle des intervenants – commanditaire, développeur, ou utilisateur final – et du domaine d'application – les exigences ne sont pas les mêmes pour un logiciel critique embarqué et pour un fournisseur de services de bureautique.

Partant de ce constat, et suivant les perspectives de la qualité définies par Garvin[d], il ressort que la qualité doit être définie, ou redéfinie, pour chaque contexte afin de prendre en considération les particularités du domaine et les attentes spécifiques de la démarche de qualité.

1 http://www.squoring.com/fr/produits

Mesurer la qualité

Une autre problématique redondante est la pertinence des mesures de caractéristiques logicielles : i.e. comment s'assurer que l'on mesure réellement ce que l'on pense mesurer, et quel impact la méthode a-t-elle ? Plusieurs auteurs (Fenton[e], Kaner[f], Pfleeger[g]) se sont ainsi penchés sur l'intégrité du processus de mesure, démontrant l'inaptitude de certaines mesures et proposant des solutions pour éviter les pièges les plus communs.

Ainsi, il n'est pas trivial de mesurer la productivité des développeurs : le nombre de lignes de code est parfois utilisé, mais finit par générer des fichiers artificiellement longs et de faible qualité. De même, la maintenabilité ou la fiabilité ne se mesurent pas à la taille ou à la complexité du code, bien qu'elles soient plutôt corrélées. La première dépend typiquement de la population de développeurs et des conventions en vigueur. La seconde ne se mesure pas à l'aune du nombre de bugs, car il n'y a pas de relation démontrée entre le nombre de bugs enregistrés et le nombre de problèmes réellement présents dans le produit. Si un produit fournit un moyen efficace pour rapporter les crashs et autres problèmes aux développeurs, un grand nombre de problèmes pourront être identifiés et corrigés, ce qui améliore la fiabilité du produit. A l'inverse un produit non testé contiendra peu de rapports de bugs, mais sera intrinsèquement peu mature.

La principale approche utilisée aujourd'hui est l'analyse multi-dimensionnelle : les attributs de qualité sont estimés grâce à un ensemble de caractéristiques du développement logiciel, pondérés et agrégés. Ces mesures doivent être comprises par tout le monde, et dans leurs modes de calcul et dans leurs conséquences sémantiques. Pour cela des approches telles que la méthode GQM (Goal-Question-Metric) apportent une aide précieuse pour la mise en place d'un processus de mesure.

Modèles de qualité

Lorsque l'on souhaite mesurer un certain nombre d'éléments d'un processus ou d'un produit, il est nécessaire de décomposer la première définition de la qualité en sous-éléments, appelés *attributs de qualité*. Ils représentent les aspects du développement que l'on souhaite mesurer et améliorer : e.g. la réactivité du support, la prédictibilité des livraisons, la lisibilité ou la stabilité du code. L'image 1 illustre un exemple de modèle de qualité dérivé de l'ISO 9126, de sa caractéristique principale à sa méthode de calcul.

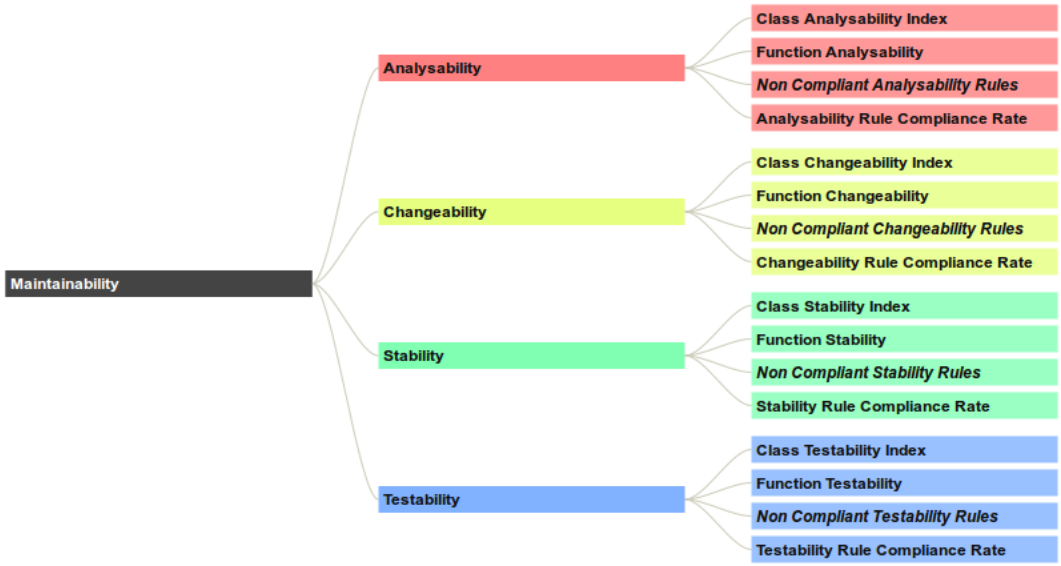

Illustration 1: Modèle de qualité orienté produit (SQuORE)

Au sein d'une même entreprise, certains départements auront des exigences de qualité particulières, ou mesureront des entités différentes : e.g. nombre de produits livrés, chiffres de ventes, retours clients. Plusieurs modèles de qualité peuvent ainsi co-exister dans l'organisation, différant par des poids variables sur les attributs de qualité ou dans leur structure même.

En considération de tous ces éléments, un certain nombre de standards ont vu le jour pour formaliser les différentes facettes du problème. On peut citer l'ISO 15504 (SPICE) ou le CMMi pour la qualité des processus, et SQUALE, l'ISO 9126 ou l'ISO 250xx (SQuaRE) pour la qualité des produits. Ces standards ont l'avantage de proposer des définitions claires et précises, une organisation normée des concepts, voire une méthode ou des recommandations de mise en œuvre, et forment ainsi un socle fiable pour l'élaboration du processus de mesure et d'analyse. En contribution à la maturité du domaine, il existe un écosystème abondant autour des standards majeurs, sous forme d'articles (critique, amélioration, implémentation...) ou de services d'audit ou de conseil.

3 L'approche SQuORE

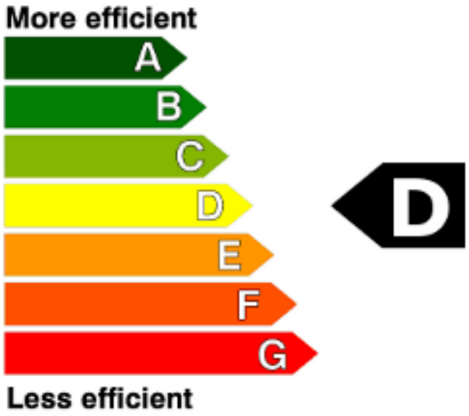

Illustration 2: La notation SQuORE

SQuORE a pour vocation de proposer un framework de mesure de la qualité prêt à l'emploi, mais entièrement configurable pour adapter le processus de mesure et d'analyse à chaque contexte[h]. Plusieurs modèles de qualité sont proposés par défaut, utilisant la structure de l'ISO 9126 ou les techniques classiques de dette technique. En s'appuyant sur ces bases de travail standardisées, il est possible ensuite d'affiner le modèle de qualité et d'ajouter aisément de nouveaux points de mesure pour adapter le processus au contexte d'analyse.

Modèles de qualité

Les modèles de qualité sont décrits au moyen de fichiers XML, modulaires et réutilisables, pour définir les attributs de qualité et les méthodes de calcul souhaités. Il est ainsi possible d'introduire une nouvelle caractéristique (e.g. la réactivité du support, en sous-caractéristique de la qualité du processus) et sa méthode de calcul (e.g. basée sur le temps moyen de réponse aux requêtes et un questionnaire de satisfaction). Un ensemble d'opérateurs permet d'agréger les métriques à plusieurs niveaux (e.g. fonctions, fichiers, application, service ou département) pour tenir compte de la culture d'entreprise et des spécificités du domaine d'application. La figure ci-contre présente un modèle de qualité orienté projet, configuré pour trois axes personnalisés : produit, processus, communauté.

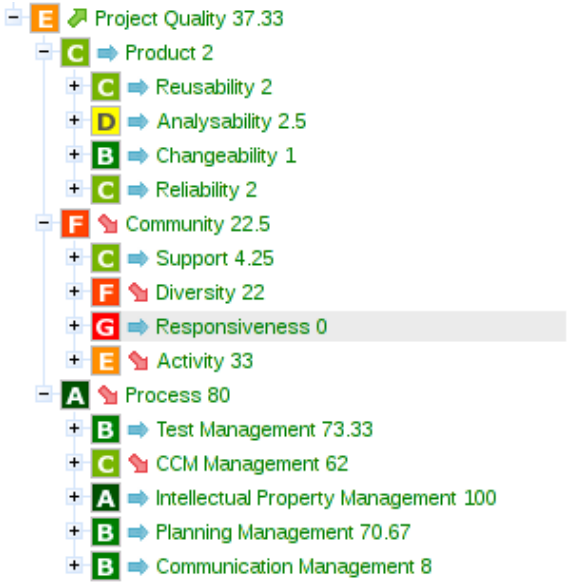

Illustration 3: Exemple de décomposition de la qualité projet

Données d'entrée

SQuORE embarque un analyseur multi-langage pour l'analyse du code source, et génère toutes les métriques classiques : taille, complexité, mesures de Halstead, profondeur de l'arbre d'héritage, etc. Mais la véritable force de SQuORE réside dans sa capacité à s'intégrer avec l'environnement extérieur. Ainsi SQuORE peut exécuter un outil de vérification de règles (CheckStyle, PMD, FindBugs, Coverity®, Polyspace®, etc.) ou un script personnalisé, et en lire le résultat en sortie. Tous les types de fichier classiques sont également supportés en standard (CSV, XML, bases de données, JSON...). Cette flexibilité permet d'inclure dans le processus de mesure des informations de natures variées, telles que les résultat des tests, les exigences, ou le nombre de produits livrés.

Les points d'action

Les points d'action (action items) sont un ensemble de règles associant plusieurs métriques à des valeurs seuil, afin de détecter des motifs complexes et générer des avertissements. Par exemple, un fichier peut avoir un grand nombre de lignes, un faible taux de commentaire et beaucoup d'anomalies (violations de règles) ; pris individuellement chacun de ces éléments peut être admis, mais leur cumul est dangereux. Pour chaque point d'action, l'outil donne une explication du problème, indique les seuils tolérés et propose des actions correctives.

Visualisation

Enfin il convient d'expliciter correctement les résultats de l'analyse aux utilisateurs finaux, pour qu'ils puissent se les approprier, comprendre la mécanique du calcul de qualité et les raisons de l'évaluation. Les images valent souvent plus que de longs discours pour cela, et il existe un graphique adapté à chaque objectif d'information : les séries temporelles pour montrer l'évolution pathologique d'une mesure (figure 4, gauche), les nuages de points pour identifier des valeurs extrêmes, ou un graphe de contrôle pour visualiser la complexité d'une fonction (figure 4, droite).

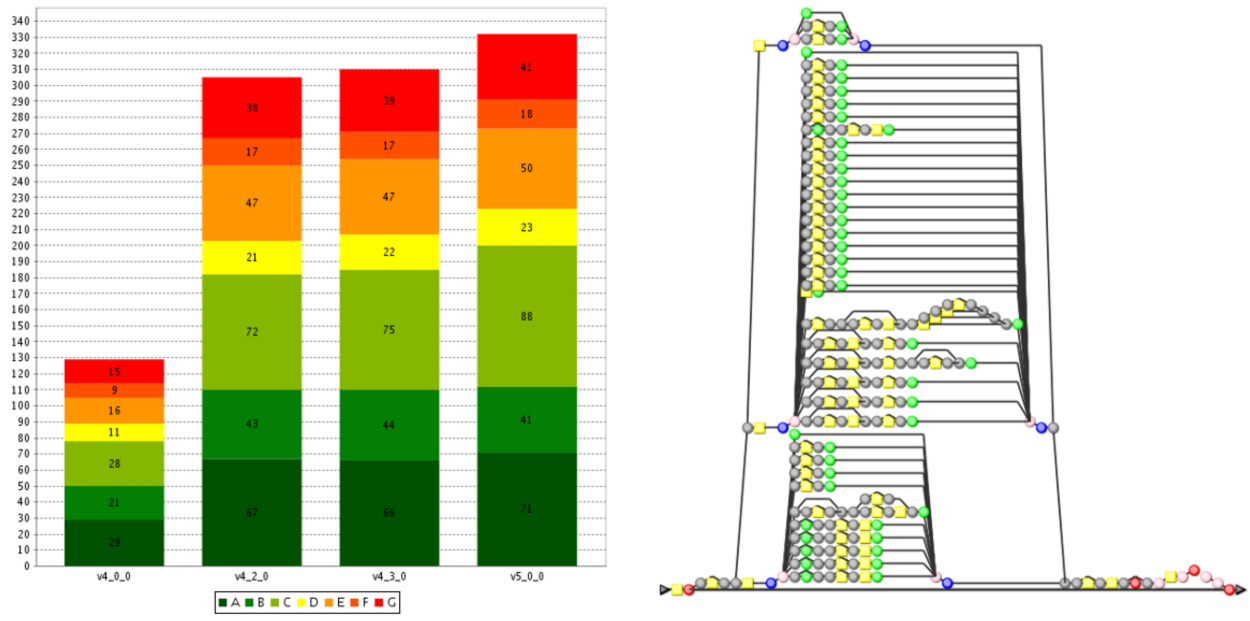

Illustration 4: Exemples de visualisation (SQuORE)

A destination des utilisateurs, premièrement : ils doivent comprendre ce que les notations signifient, et savoir quelles actions permettraient de les améliorer. Au titre du suivi de projet, ensuite, pour comprendre et prévoir le fonctionnement interne du projet.

Ce dernier maillon de la chaîne est d'une importance fondamentale, car l'ensemble du processus est inutile si les intervenants ne savent pas interpréter et utiliser ces résultats, et s'il n'y a pas d'amélioration concrète et de résultats visibles sur la qualité. Un autre aspect important est de rendre visible l'amélioration du processus ou du produit : pour justifier l'investissement, et accroître la confiance dans le système de mesure.

4 Conclusion

Nous avons présenté dans cet article les notions fondamentales de la qualité logicielle, de la mesure à la construction d'un modèle de qualité adapté. Les principaux problèmes connus ont été abordés, ainsi que les méthodes proposées pour les résoudre. Nous avons également vu les principes d'approche de l'outil SQuORE, et comment il peut être appliqué à un processus de mesure et d'analyse spécifique.

Il en ressort qu'il n'existe pas de définition universelle ou de panacée pour la qualité logicielle : les concepts doivent être étudiés et ajustés à chaque situation pour être pleinement utilisables. Cependant, en tenant compte des exigences et spécificités de l'organisation ou du projet, en adaptant les points de mesure aux informations disponibles, et en impliquant les utilisateurs du processus de mesure, il est non seulement possible d'obtenir une évaluation pertinente des projets, mais également d'aider pro-activement à l'amélioration de leur qualité.

a Deming, W. E. (1988). Out of the crisis: quality, productivity and competitive position. Cambridge University Press.
b Juran, J. M., Godfrey, A. B., Hoogstoel, R. E., & Schilling, E. G. (1999). Juran's Quality Handbook (Vol. 2). McGraw Hill New York.
c Crosby, P. B. (1979). Quality is free: The art of making quality certain (Vol. 94). McGraw-Hill New York.
d Kitchenham, B., & Pfleeger, S. (1996). Software quality: the elusive target. IEEE Software, 13(1), 12–21.
e Fenton, N., & Pfleeger, S. (1991). Software metrics.
f Kaner, C., & Bond, W. P. (2004). Software engineering metrics: What do they measure and how do we know? Methodology, 8(6), 1–12.
g Fenton, N., & Pfleeger, S. (1998). Software metrics: a rigorous and practical approach. Brooks/Cole Pub Co.
h Baldassari, B. (2012). SQuORE : une nouvelle approche de mesure de la qualité des projets logiciels. Génie Logiciel, 103, 9–14.