Allée de la Réflexion

Qualité et quantité

Logiciel élégant contre logiciel obèse

Dans son dernier ouvrage, intitulé « Le logiciel à valeur ajoutée », Yves Constantinidis décrit les six caractéristiques fondamentales qui expriment la valeur d'un logiciel.
La « fonctionnalité »,caractérise l'ensemble des fonctions remplies par le logiciel et les propriétés de chacune de ces fonctions.
Mais « plus » signifie-t-il « mieux » ? Dans ce domaine comme dans d'autres, la quantité semble nuire à la qualité. Démonstration.

Qualité ou quantité ?

Après les ingénieurs qualité, faudra-t-il un jour nommer des « ingénieurs quantité » ?

Cela semble une boutade. Pourtant, le logiciel, dans son ensemble, souffre d'une tendance à l'obésité. L'obésité du logiciel est due, avant tout, au fait que celui-ci est impalpable et immatériel. Les mégaoctets sont de moins en moins chers, les MIPS et les mégaflops[1] aussi, et toute l'industrie souffre d'une fuite en avant. On peut imaginer que cela ne pose aucun problème sur le plan de la qualité, de la valeur des logiciels. Si la mémoire et la puissance machine ne coûtent pas cher, pourquoi s'en priver ?

Mais le problème n'est pas là. Il ne s'agit pas de s'en priver, il s'agit au contraire de l'utiliser judicieusement. Compenser une « mauvaise programmation » par une consommation excessive de ressources est une erreur classique des étudiants en programmation. Mais si vous programmez mal, le moment arrivera où l'augmentation de la puissance ne compensera pas l'augmentation des besoins. Il y a une manière élégante de programmer, et c'est aussi la plus efficace.

Le logiciel obèse

Ce qui est valable pour la programmation (et que nous n'avons cité qu'à titre indicatif au paragraphe précédent) l'est également pour la construction du logiciel dans son ensemble.

Prenons un exemple bien connu : d'une version à l'autre, les logiciels de bureautique occupent une place de plus en plus importante sur le disque dur. Cela pourrait être sans conséquences. La capacité des disques durs n'est-elle pas de plus en plus importante, pour un prix et un poids de plus en plus faible ? Cependant le problème n'est pas dans la capacité du disque dur. Il est dans l'utilité des fonctions offertes par le logiciel, et par leur utilisabilité (aptitude à être utilisé, avec facilité et efficacité ?).

Pour nous en convaincre, essayons de mettre chaque fonction offerte par un logiciel dans une (ou plusieurs) des catégories suivantes :

- FEU : fonctions effectivement utilisées, par au moins 20 % des utilisateurs ;
- FFU : fonctions fréquemment utilisées, par 80 % des utilisateurs ou plus ;
- FPU : fonctions potentiellement utiles, qui seraient utilisées si elles étaient mieux documentées qu'elles ne le sont aujourd'hui ;
- FDE : fonctions déficientes, correctement ou incorrectement documentées, qui ne peuvent ou ne doivent pas être utilisées, parce qu'elles font prendre des risques aux utilisateurs, ou leur font perdre plus de temps qu'elles ne leur en font gagner ;

[1] *Mesures de la puissance d'un processeur. MIPS : million d'instructions par seconde. Mégaflop : million d'instructions en virgule flottante (*floating point operations*) par seconde.*

- FO : fonctions obsolètes, présentes dans les versions précédentes, et rendues inutiles par l'arrivée de nouvelles fonctions ;
- FR : fonctions redondantes, existant sous d'autres formes dans le logiciel, ou plus judicieusement présentes dans d'autres logiciels ;
- FDI : fonctions dissimulées, à la fois non documentées et inutiles ;
- FM : fonctions mortes, c'est-à-dire existantes, défaillantes ou non, mais enfouies dans le logiciel et inaccessibles à l'utilisateur.

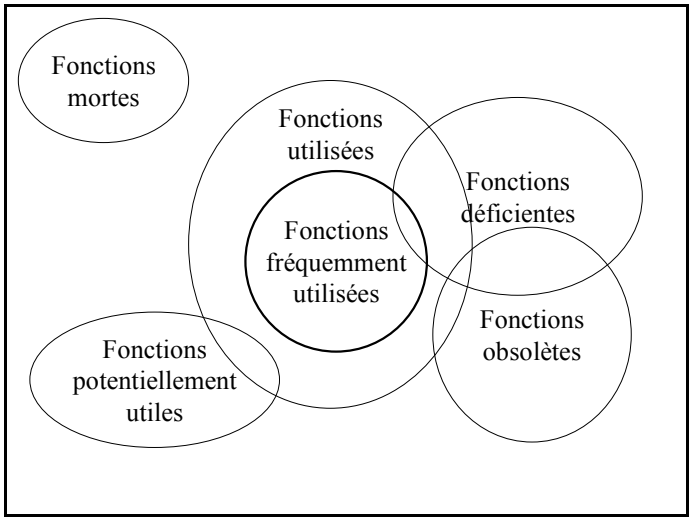

Figure 1. Obésité fonctionnelle

Lorsque le nombre total de fonctions est très largement supérieur au nombre de fonctions utiles, on pourra dire que le logiciel est obèse.

Une autre mesure de l'obésité pourrait être obtenue en faisant le rapport entre la complexité, selon McCabe et la fonctionnalité (en points de fonctions).

- Le problème n'est pas le volume occupé, mais la difficulté à gérer l'ensemble.
- Les fonctions obsolètes ou redondantes trompent ou perturbent l'utilisateur.
- Les fonctions mortes rendent plus difficile (alourdissent) la maintenance du logiciel..
- Les fonctions déficientes posent des problèmes de fiabilité.

Autres formes d'obésité

Chaque fois que l'on en fait « plus » au lieu de faire « mieux », la qualité en pâtit immanquablement. La confusion entre qualité et quantité est très répandue dans le monde du logiciel, à tel point que le terme même de qualité fait fuir ceux-là mêmes qui s'attellent à un véritable travail de qualité, avec le souci de l'utilisateur en tête. On peut citer, entre autres :

- l'obésité de la documentation utilisateur, et qui la rend totalement inutilisable ;
- l'obésité des documents méthodologiques à l'usage des développeurs, que personne ne pourra lire avec attention ;
- la tendance, tant de fois décriée, à augmenter la taille des équipes lorsqu'un projet de développement prend du retard, avec l'effet exactement inverse de celui escompté, et le risque de développer des fonctions redondantes.

Yves Constantinidis

Extrait de l'ouvrage *Le logiciel à valeur ajoutée*, Hermès, 2001