LES COMMENTAIRES EN INFORMATIQUE

Introduction

Chapitre 1

Définitions

1.1 Commentaires et documentation

1.1.1 Commentaires

Pour tous ceux qui ont déjà développé le moindre petit programme informatique, les commentaires n'ont pas de secret. Et c'est tout à fait normal, car ils n'ont pas vocation à en avoir. Pour définir ce qu'est un commentaire, commençons par regarder la définition qu'en donne Wikipedia :

> " Les commentaires sont, en programmation informatique, des portions du code source ignorées par le compilateur ou l'interpréteur, car destinés en général à un lecteur humain et non censés influencer l'exécution du programme. "

Comme le décrit cette définition, un commentaire n'est pas interpreté. Il n'a donc aucune incidence sur la compilation, l'interprétation et l'exécution d'un programme. Il est donc le meilleur outil pour conserver une trace écrite quelconque dans un fichier source sans qu'elle n'ait de conséquences. Les commentaires peuvent être du texte expliquant un processus, du code à analyser ou à corriger, des informations à propos du fichier ou de son contenu. Ils sont très utilisés durant les différentes étapes de développement : conception, programmation, test, maintenance, ...

Avec la professionnalisation du développement et l'avènement de la programmation en équipe, il est devenu vital de créer un logiciel de qualité suffisamment clair pour qu'utilisateur, comme développeur, sachent ce qu'ils peuvent faire et comment le faire.

1.1.2 Documentation

> " La documentation logicielle est un texte écrit qui accompagne le logiciel informatique. Elle explique comment le logiciel fonctionne, et/ou comment on doit l'employer. "

En plus de cette définition, l'article Wikipedia décrit cinq types de documentation :

— L'expression du besoin
— Architecture / Conception
— Technique
— Utilisateur
— Marketing

Lors du développement d'un logiciel, l'étape de documentation représente la finalisation du projet. Elle intervient après que le logiciel ait été testé et avant de le donner aux clients. Et malgré son intrication avec le code, elle en est séparée. C'est pourquoi certains développeurs ont conçu des outils pour créer la documentation directement à partir des fichiers source. Or chaque entreprise définit sa propre manière de développer. Comme il aurait été quasiment impossible d'adapter le code de toutes les entreprises sur un même modèle, ils ont choisi de définir un modèle de commentaire à la place.

Depuis lors, en utilisant des outils comme Doxygen ou Javadoc, il est possible d'automatiser la documentation en détournant les commentaires de leur utilisation primaire.

1.2 Qualité logicielle

Définir la qualité logicielle de manière précise prendrait énormément de temps. Je vais simplement me restreindre à définir le concept principal ainsi que le lien entre les commentaires, la documentation et la qualité.

La qualité logicielle représente un champ du génie logiciel dont le but est d'établir des critères d'évaluation pour les logiciels. Un logiciel de qualité a une durée de vie plus longue, une communauté d'utilisateur plus importante et est, la plupart du temps, plus performant. Plusieurs normes ont été définis pour qualifier un logiciel : en France, la plus connue est la norme ISO-9126 (complétée par l'ISO 25000) et elle définit six caractéristiques divisées en vingt-sept sous catégories :

1. Capacité fonctionnelle
2. Fiabilité
3. Facilité d'utilisation
4. Rendement / Efficacité
5. Maintenabilité
6. Portabilité

Pour ce qui nous concerne, je vais occulter tous ce qui n'est lié qu'au code pur pour ne conserver que les sous-caractéristiques intéressantes de notre point de vue :

— **Facilité d'utilisation** - Facilité de compréhension
— **Facilité d'utilisation** - Facilité d'apprentissage
— **Facilité d'utilisation** - Facilité d'exploitation
— **Maintenabilité** - Facilité d'analyse

Étant donné que les commentaires n'ont pas d'incidence sur le programme, tout comme la documentation, ils n'ont pour but que de facilité certains processus. Mais ces processus sont vitaux pour la vie d'un projet. Si vous concevez un logiciel que personne ne peut comprendre, personne ne pourra reprendre votre travail pour l'améliorer. S'il est hardu pour de nouveaux développeurs de rentrer dans le projet, vous vous retrouverez dans la même situation avec personne pour continuer votre projet.

1.3 Couverture commentaire

En informatique, le terme de couverture prend une multitude de sens en fonction du contexte : on peut parler du champ d'action d'un logiciel, du nombre de besoins clients couvert, ...

En génie logiciel, les couvertures représentent des metriques servant à évaluer la qualité d'un logiciel. Par exemple, les tests unitaires servent à analyser la couverture de code lors d'une exécution. Ces métriques sont un gages de qualité qui peux servir de comparateur entre deux logiciels.

Mais alors, comment mesurer la qualité des commentaires ?

La couverture commentaire représente le moyen d'évaluer un logiciel en observant ses commentaires et plus particulièrement si chaque portion de code est correctement commentée. Ce n'est pas quelque chose d'aisée à faire car les commentaires ne sont pas reglementés par une syntaxe (mise à part les balises qui ouvre et ferme un commentaire). Il est tout à fait possible d'écrire une ligne de commentaire qui va parfaitement expliquer les dix lignes de codes qui suivent, ou écrire dix lignes de commentaires pour simplement clarifier une ligne de code. Certains développeurs vont même jusqu'à ne pas utiliser de commentaires de manière volontaire.

Il existe peu de définitions de la couverture commentaire et encore moins de logiciel pour la mesurer. En C, la norme ISO 9126 définit la fréquence des commentaires comme étant le rapport entre le nombre de blocs de commentaires (avant et dans la fonction) et le nombre d'instructions de la fonction. Sonarqube, qui est le plus connu des logiciels évaluant la qualité d'un code source, définit deux métriques intéressantes à propos des commentaires : la première est le nombre de commentaire utile (non vide et non esthétique),

la seconde représente la densité de commentaires qui correspond au pourcentage de commentaires dans le code.

Chapitre 2

À l'intérieur du code

2.1 Absence de documentation

Si l'on considère une documentation qui respecte la qualité logicielle, il lui reste un défaut dont elle ne pourra jamais se séparer : elle est externe au code source. Je vais surement nourrir un cliché mais un développeur passe la plupart de son temps à regarder du code. Et devoir jongler entre le code et la documentation est une perte de temps qui croit exponentiellement avec les dimensions du logiciel. La documentation est importante, mais les commentaires à l'intérieur du code aideront les développeurs beaucoup plus rapidement. C'est pourquoi ils ne doivent pas être mis de côté au profit de commentaire pour la documentation.

Tous les langages définissent des balises de commentaires. D'un point de vue syntaxique, on peut réunir chaque commentaire dans un de ces trois groupes :

— **bloc** - commentaire sur plusieurs lignes encadré par deux balises.
— **ligne** - commentaire sur une ligne, utilisant une seule balise en début de ligne.
— **à la traine** - commentaire qui termine une ligne de code, encadré par deux balises ou utilisant seulement une balise de début de commentaire.

Néanmoins, certains langages n'utilisent que deux groupes : il existe des langages qui définissent un commentaire comme étant du texte compris entre deux balises uniquement. Il n'existe pas de commentaire de type ligne dans ces langages. Parmis ces langages, on trouve le C, le HTML, ... À l'inverse, certains langages ne définissent pas de commentaires bloc car ils n'utilisent qu'une seule et unique balise. Le PHP, le langage Latex ou le Bash sont des langages de ce types.

Pour simplifier l'analyse j'ai choisi de voir un commentaire comme étant une ligne du fichier et de définir alors trois types de lignes :

— Ligne de code
— Ligne de commentaire
— Ligne mix (contenant à la fois du code et un commentaire)

Grâce à cette simplification, il est possible d'analyser n'importe quel type de commentaire de n'importe quel type de langage.

2.2 Type de commentaires

En analysant plusieurs projets informatiques (personnels et professionels), on peut séparer les commentaires en plusieurs catégories en fonction de l'objectif visé. J'ai alors créé sept catégories de commentaires qui permettent de trier la totalité des commentaires :

1. **Esthétique** - commentaire sans importance pour le fichier. Ce sont, la plupart du temps, des lignes sans texte qui ont pour but de créer des séparations entre des portions de code.

2. **En-tête** - commentaire très rarement utilisé lors de développements personnels mais qui sont important dans le domaine professionnel. Il définisse le nom du fichier, son auteur, la date de création, de modification, ... Là encore, aucun intérêt pour le code vu qu'il s'agit d'une simple description du fichier.

3. **Documentation** - commentaire utilisé pour générer, de manière automatique, la documentation externe des fonctions de ce fichier. Ils peuvent donner quelques informations à propos de la fonction mais ne doivent pas servir de remplacements aux commentaires standards.

4. **Temporaire** - commentaire décrivant une portion de code qui est vouée à disparaître avant de livrer le code. Il sert à informer les autres développeurs d'une modification à faire.

5. **Problème** - commentaire décrivant une portion de code qui provoque, de manière continuelle ou spontanée, une erreur quelconque. On peut l'assimiler à un commentaire temporaire qui provoque une erreur. Un logiciel qui ne provoque pas d'erreur sera alors exempt de commentaires de ce type.

6. **Evolution** - commentaire qui décrit une optimisation ou un changement à effectuer dans un futur plus lointain. Ce type de commentaires peut se trouver dans un logiciel livré. C'est ce que l'on appelle, généralement, des commentaires TODO.

7. **Normal** - commentaire qui n'entre dans aucune catégorie. Et par défaut, un commentaire normal est un commentaire donc le but est de décrire un algorithme, un fonctionnement, le contenu d'un fichier, ... C'est ce type de commentaire qui est important pour un développeur qui cherche à comprendre un fichier.

2.3 Couverture commentaire