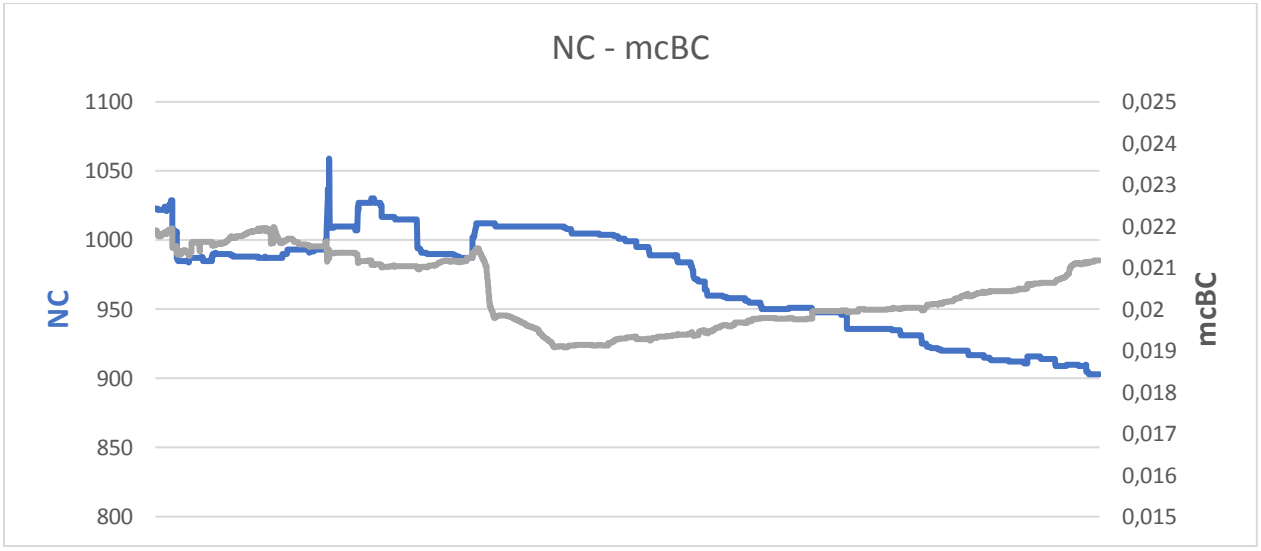
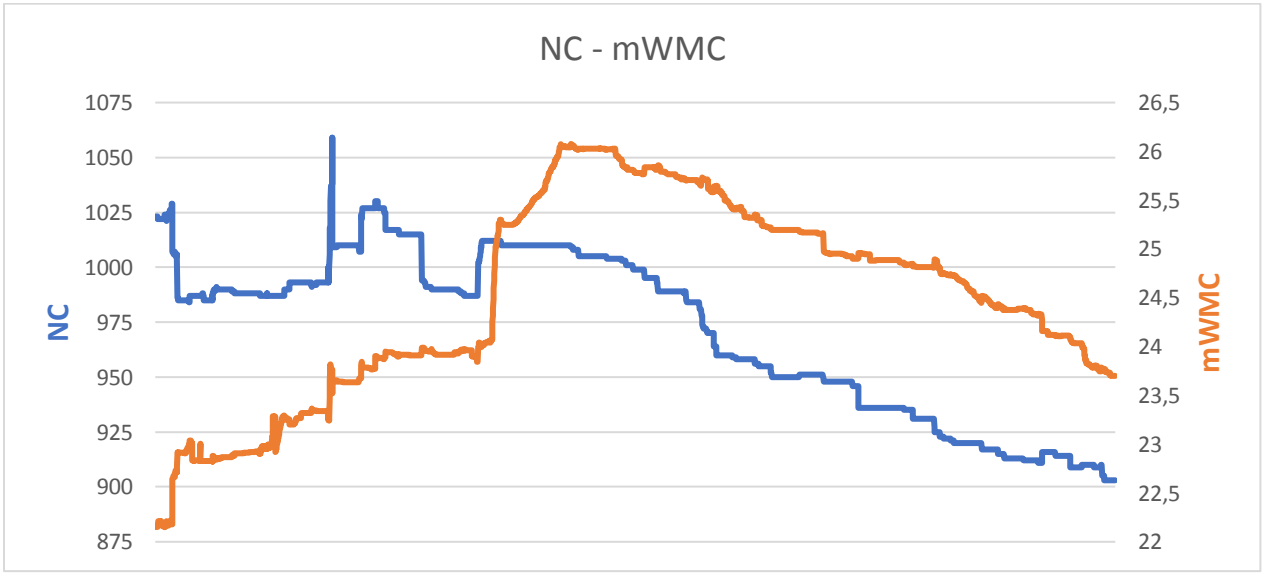
#4 Évolution des métriques représenté sous forme de graphiques :

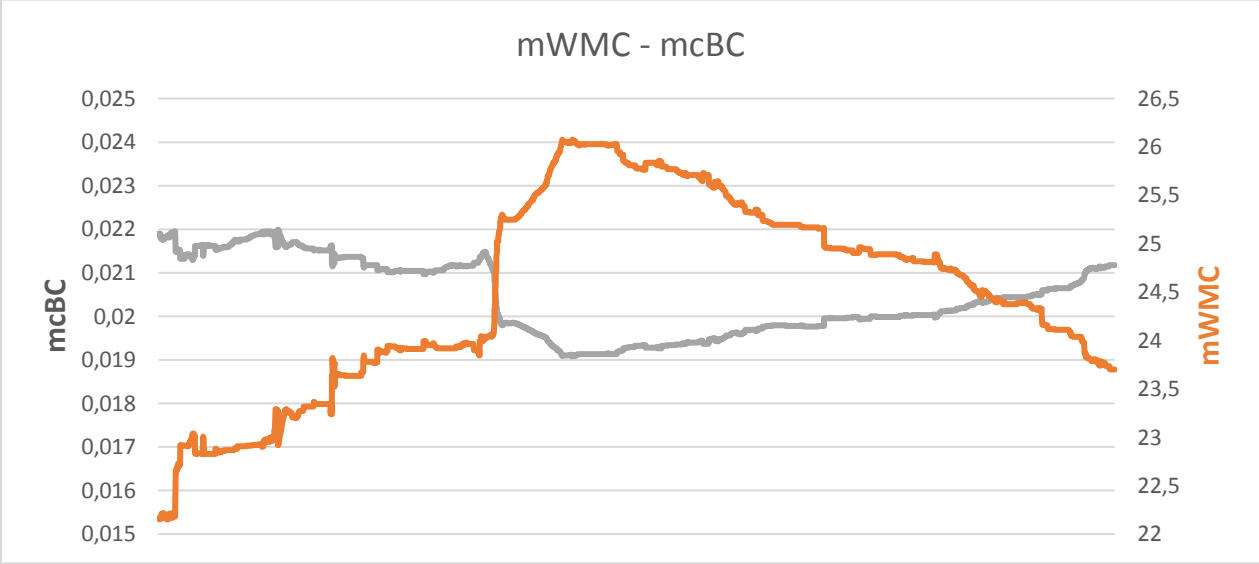


On voit qu'avec le temps, le nombre de classe [NC] diminue considérablement, probablement dû à de la refactorisation et de l'optimisation dans le code. Ensuite, on voit que le degré selon lequel les classes sont bien commentée [mcBC] augmente au fil du temps ce qui est probablement dû au fait que la documentation (et donc les commentaires) est améliorée au fil du temps.



Le graphique de la métrique mWMC est intéressant : il montre clairement qu'au début et pendant un certain temps suivant le commit initial, la complexité des classes augmente considérablement. Une explication plausible est qu'au début les développeurs ajoutent beaucoup de fonctionnalité en peu de temps. En revanche, après un certain temps, le projet semble maturer et donc vers la moitié du graphique, on voit que la complexité des classes

atteint un plateau avant de commencer à redescendre rapidement et de façon constante, ce qui est probablement dû à de l'optimisation.



#5

Relation entre NC et mcBC

À l'aide du graphique « NC – mcBC », on peut observer qu'à partir du tier du développement jusqu'à aujourd'hui, les métriques sont clairement inversement proportionnelles : plus le nombre de commentaire augmente, plus le nombre de classe diminue. L'hypothèse la plus probable est que cela serait dû à de la refactorisation du code ainsi qu'à de l'optimisation. Donc, avec le temps, chaque classe est de mieux en mieux documenté ce qui en facilite l'usage et le développement et cela aide à réduire le nombre de classes inutiles.

Relation entre NC et mWMC

Au début du développement, les deux métriques semblent être plutôt indépendante l'une de l'autre. Cependant, vers le milieu du développement, les deux métriques semblent évoluer exactement au même rythme : elles diminuent toutes les deux à la même vitesse. Donc, une fois passé le milieu du développement, le nombre de classe et la complexité de celles-ci diminue ensemble ce qui, encore une fois, est probablement dû à de l'optimisation et de la refactorisation dans le code.