Qualité du logiciel et de la maintenance

La qualité du logiciel et de la maintenance représente l'un des défis majeurs pour un développeur. En effet, sur des projets de grande envergure, maintenir la cohérence d'un code de qualité est un travail de précision et de rigueur. Quant à la maintenance, cette tâche constitue la partie la plus importante du cycle de vie d'un logiciel ou d'une application.
Ainsi, l'introduction de l'IA générative, réputée pour sa réactivité et sa précision, apparaît en surface comme la solution miracle face aux défis majeurs que sont la qualité et la maintenance.
Mais cette image d'une intelligence artificielle infaillible est une illusion : l'outil intelligent peut présenter des lacunes et n'est pas un outil pouvant remplacer la rigueur et la réflexion humaine.
Une problématique se pose : comment s'assurer qu'un code généré en très peu de temps par une IA reste maintenable sur tout le cycle de vie d'un logiciel ou d'une application ?

Dans un premier temps, nous allons analyser comment l'utilisation de l'IA générative a dégradé la qualité globale de la structure logicielle. Ensuite, nous étudierons sa fiabilité et le phénomène d'hallucination. Enfin, nous terminerons cette étude par la sécurité et l'évolutivité à long terme.

Bien que l'intelligence artificielle accélère considérablement la phase de développement, son intégration lors de cette phase impacte profondément la structure même du produit, favorisant une résolution immédiate plutôt qu'une architecture logicielle durable.

Tout d'abord, le code churn, indicateur de code de mauvaise qualité, qui désigne du code ajouté puis supprimé ou modifié en moins de deux semaines, ne fait qu'augmenter depuis l'émergence des modèles IA.
D'après l'étude de GitClear en 2024, on constate que 46% des modifications apportées au code correspondaient à l'ajout de nouvelles lignes, alors que le nombre de lignes copiées-collées a dépassé celui des lignes déplacées. Sachant que le déplacement est considéré comme un signe de qualité, témoignant d'une réorganisation de code, on peut en déduire que l'impact de l'intelligence artificielle se fait plus intense et tend vers la propagation de mauvaises pratiques dans l'écriture de code.

De plus, les pratiques de Clean Code, la lisibilité du code et ses standards, sont en baisse.
En produisant du code généré par différents prompts, la structure globale du code logiciel devient de plus en plus incohérent. L'intelligence artificielle produit du code dépourvu de design patterns, et fournit un travail difficile à relire pour un humain, pollué par des code smells qui contredisent les principes fondamentaux de la maintenabilité du code logiciel.

Ainsi, l'IA générative marque la dégradation de la structure logicielle par son code dépourvu des bonnes pratiques de Clean Code.

Au-delà des problématiques purement structurelles, l'utilisation de l'assistance intelligent soulève un enjeu majeur quant à l'exactitude des solutions proposées, un risque qui prend toute son ampleur lors de l'implémentation d'algorithmes complexes.

En effet, le phénomène lié à l'IA générative nommé « l'illusion de l'exactitude », qui survient lorsque l'outil ne possède pas la solution à un problème complexe, se met à « halluciner », propose une solution à partir d'informations inventées.
Dans l'étude des chercheurs de l'Université de Purdue en 2023, il est démontré que 52% des réponses de ChatGPT sont incorrectes et 77 % sont verbeuses. L'intelligence artificielle donne avec aplomb des solutions qui échouent silencieusement sur des cas limites, comme dans notre cas de l'algorithme de Voronoi l'alignement de plusieurs points. Ce manque de fiabilité est bien trop négligé par les développeurs qui font une confiance aveugle à l'IA générative.

Mais plus encore, cette confiance qu'ont les développeurs envers l'intelligence artificielle appuie l'effet de « Boite noire » et la perte de compétences.
La délégation systématique de tâches complexes à l'intelligence artificielle mène à la transformation du cœur du fonctionnement d'un logiciel en une « boite noire ». L'équipe de développement est alors privée du niveau de connaissance nécessaires afin de pouvoir corriger les bugs en production. Cela mène fatalement à la refonte des systèmes créés par l'outil IA, annulant les bénéfices recherchés liés à son utilisation (vitesse de réaction, précision), ou pire encore, mener vers une dépendance de l'équipe de développement envers l'intelligence artificielle.

Donc l'IA générative se retrouve être un outil peu fiable lors de la génération de solutions complexes sur le long terme, fiabilité beaucoup trop passée sous silence par rapport à son taux d'utilisation qui s'exprime par millions.

Enfin, si un code généré par IA peut sembler parfaitement opérationnel au moment de sa création, sa viabilité sur l'ensemble du cycle de vie du logiciel est fortement menacée par des enjeux critiques liés à la sécurité, à la gestion des dépendances et à la fiabilité des tests.

L'intelligence artificielle, lorsqu'elle est mal utilisée, représente un danger majeur dans la sécurité d'un logiciel, donnant un « faux sentiment » de sécurité.
L'université de Stanford à travers une étude en 2022 a démontré que les développeurs utilisant des intelligences artificielles ont tendance à intégrer davantage de vulnérabilités et de dépendances obsolètes, que des développeurs n'utilisant pas d'intelligence artificielle. Elle donnerait un « faux sentiment » de sécurité, donnant un excès de confiance aux développeurs l'utilisant, en leur donnant l'illusion d'écrire un code sécurisé et augmentant au contraire les vulnérabilités.

Ce problème de sécurité a par logique un impact direct sur l'évolutivité d'un logiciel à long terme.
L'IA générative va jusqu'à proposer des tests unitaires trompeurs, dits « tautologiques », qui se contentent de valider la logique erronée qu'elle vient elle-même d'implémenter, faussant la robustesse du code qui rend la maintenance future complètement aveugles des vrais problèmes de fond.
Alors comme démontré, l'utilisation aveugle de l'intelligence artificielle représente un réel

danger pour la sécurité et la maintenance sur le long terme.

Pour conclure, bien que l'IA générative puisse arborer une image de sécurité, de précision, et de connaissance solide, elle s'avère faillible : responsable de la dégradation globale de la structure logicielle, n'ayant souvent pas les connaissances adéquates pour apporter de vraies réponses, et représentant un risque de sécurité tant dans immédiat que sur le long terme.
Cet exercice sur le diagramme de Voronoi illustre très bien toutes les vulnérabilités évoquées précédemment. Quels que soient les cas d'usages, on pouvait observer les limites de ces modèles sur la qualité de code et la maintenance applicative. Une recherche profonde sur ce diagramme de Voronoi fut obligatoire, qui, sans cela, serait impossible car l'intelligence artificielle seule ne faisait que créer et corriger des erreurs algorithmiques dont elle seule en avait la compréhension.
Pour nos projets futurs, ce travail de sensibilisation devient une confirmation quant à l'utilisation des IA génératives, qui ne doit jamais être utilisé au détriment d'un travail de réflexion et de rigueur.

Sources :

- Etude de l'Université de Stanford : https://dl.acm.org/doi/epdf/10.1145/3576915.3623157
- Article de ICTJournal : https://www.ictjournal.ch/etudes/2025-02-25/limpact-negatif-du-code-genere-par-lia-se-confirme#:~:text=Parmi%20les%20nombreux%20indicateurs%20de,dépassé%20celui%20des%20lignes%20déplacées.
- Article de ACM Digital Library : https://dl.acm.org/doi/10.1145/3688671.3688783