Quel gain de productivité pour les développeurs grâce à l'IA ?

27 août 2025, 14:37

Pascal Pollet

La véritable contribution de l'IA au développement de logiciels selon une étude de l'université de Stanford

L'un des principaux domaines d'application de l'IA générative dans le développement de logiciels est l'aide apportée aux programmeurs pour écrire du code. Il existe désormais de nombreux outils d'IA spécialisés dans l'écriture de code. Certaines entreprises affirment même qu'elles pourront bientôt remplacer une partie de leurs développeurs de logiciels par l'IA. Mais que peut-on réellement attendre de l'IA pour la programmation ?

Dans le cadre d'une étude à grande échelle, **l'université de Stanford** a examiné l'impact des assistants d'IA sur la productivité dans l'écriture de code logiciel. Et la méthodologie utilisée en fait une étude très particulière. Au lieu de travailler avec des tests artificiels, les chercheurs se sont basés sur des données réelles provenant de plus de 600 entreprises qui leur ont donné accès à leurs référentiels de code, les « repositories ».

Les chercheurs ont ainsi eu accès au travail de plus de 100 000 développeurs, ce qui leur a permis

d’analyser des milliards de lignes de code et de mesurer, entre autres, la fréquence des modifications (« commits ») apportées aux référentiels. La productivité n’a pas été mesurée au nombre de lignes de code écrites, mais à la fonctionnalité du logiciel fourni.

Quel est le véritable impact de l’IA pour le développement de logiciels ?

L’étude montre que l’**écriture de code assistée par l’IA** augmente dans un premier temps le rendement d’environ 30 à 40 %. Toutefois, ce gain de productivité ne se maintient pas durablement : l’utilisation de l’IA introduit également des erreurs que les programmeurs doivent ensuite détecter et corriger. En fin de compte, ce travail supplémentaire réduit le gain de productivité moyen à environ 15 à 20 %.

Dans quels cas l’IA fonctionne-t-elle et ne fonctionne-t-elle pas ?

L’étude montre que l’impact de l’**IA dans le développement de logiciels** varie considérablement en fonction de divers facteurs :

1. La complexité de la tâche de programmation

L’IA est nettement plus performante pour les tâches de programmation simples. Pour les tâches plus complexes, les gains sont moins importants et, dans certains cas, l’IA peut même entraîner une baisse de la productivité.

2. La maturité du code base

L’IA fonctionne plus efficacement avec les nouveaux projets (« greenfield ») que sur les codes bases existants (« brownfield »). Quelques chiffres tirés de l’étude :

- Tâches greenfield simples : gains de 30 à 35 %
- Tâches greenfield complexes : gains de 10 à 15 %
- Tâches brownfield simples : gains de 15 à 20 %
- Tâches brownfield complexes : gains de 5 à 10 %

3. La popularité du langage de programmation

L’efficacité de l’IA dépend également du langage de programmation utilisé. Pour les langages moins populaires, comme Cobol, l’IA n’apporte presque aucun avantage, même pour des tâches simples. En présence d’un langage de programmation peu courant et d’une complexité élevée, l’IA peut même avoir un impact négatif sur la productivité.

En revanche, pour les langages populaires tels que **Python** et **Java**, les gains sont bel et bien au rendez-vous, car les systèmes d’IA sont beaucoup mieux entraînés grâce aux volumes importants de code disponible.

4. L’ampleur du code base disponible

À mesure que la taille du code base augmente, les avantages de l’IA diminuent sensiblement. Cette information est particulièrement pertinente quand on sait que la plupart des codes bases modernes comptent au moins plusieurs milliers de lignes de code.

Cette caractéristique de l’IA s’explique en partie par le fait que les performances des **grands modèles de langage (Large Language Models, LLM)** pour les tâches d’écriture de code diminuent à mesure que la longueur du contexte augmente. La longueur du contexte est la quantité maximale d’informations (exprimée en jetons) qu’un modèle de langage peut traiter simultanément, à partir tant de l’invite saisie que de la sortie générée.

L’IA est une solution puissante, pas une baguette magique

L’IA générative permet indéniablement une augmentation considérable des performances des développeurs de logiciels. Mais il convient de nuancer cette affirmation. Les **gains de productivité** ne sont pas un phénomène nouveau dans le monde des logiciels. Par le passé, des gains de productivité encore plus importants ont souvent été obtenus grâce à l’adoption de nouveaux langages de programmation. Ainsi, les langages d’écriture de scripts tels que Python offrent un gain de productivité d’environ 30 à 70 % par rapport aux langages procéduraux tels que Fortran ou C++. Si on les compare aux langages d’assemblage, ces langages modernes sont au moins cinq fois plus rapides pour les tâches de développement.

L’IA générative est donc un puissant accélérateur pour le développement de logiciels, mais pas la panacée pour une plus grande efficacité. Le contexte qui va du langage de programmation à la nature de la tâche détermine en grande partie les gains de performances. Il est donc important pour les organisations d’**intégrer intelligemment l’IA dans leurs pratiques de développement**, plutôt que de compter sur une hausse « magique » de la productivité.

Sources

- Https://www.youtube.com/watch?v=tbDDYKRFjhk
- Softwareengineeringproductivity.stanford.edu

Auteurs

Pascal Pollet