Module Veille Technologique & Outils

Yannis El HAJAOUI

Table des matières

Introduction

Depuis 2022, l'intelligence artificielle générative a connu une progression remarquable dans le domaine du développement logiciel. Ce qui relevait encore il y a peu d'une curiosité expérimentale réservée aux chercheurs est devenu, en l'espace de deux ans, un outil du quotidien pour des millions de développeurs à travers le monde. GitHub Copilot dépasse ainsi le million d'abonnés payants dès sa première année d'existence, tandis que des outils comme Cursor, Windsurf ou Gemini Code Assist s'imposent progressivement dans les environnements de développement intégrés.

Ce mouvement de fond ne se limite pas à une simple amélioration de la saisie automatique. Il touche en profondeur la manière dont le code est écrit, relu, testé et documenté. Il interroge également les pratiques de formation des développeurs, les modèles de responsabilité en cas de bug, la propriété intellectuelle du code produit, et la souveraineté des données d'entreprise transmises à des services cloud.

C'est dans ce contexte que s'inscrit la présente veille technologique. Elle vise à dresser un état des lieux documenté et objectif de l'intelligence artificielle générative appliquée au développement logiciel, en s'appuyant sur des sources variées et complémentaires : rapports d'éditeurs, études académiques, analyses de cabinets spécialisés, et retours d'expérience de la communauté des développeurs.

La veille s'articule autour de trois axes thématiques : le panorama des outils disponibles, l'impact réel sur la productivité et la qualité du code, et les limites techniques ainsi que les enjeux éthiques et professionnels soulevés par cette évolution.

Présentation du thème

L'intelligence artificielle générative désigne un ensemble de modèles capables de produire du contenu original — texte, code, images — à partir de données d'entraînement massives et d'une instruction formulée en langage naturel, appelée prompt. Dans le domaine du développement logiciel, cette capacité se traduit par des assistants qui complètent automatiquement des fonctions, génèrent des tests unitaires, rédigent de la documentation ou détectent et corrigent des anomalies dans le code existant.

Les Large Language Models, ou LLM, constituent le moteur de ces outils. Entraînés sur des milliards de lignes de code et de texte, ils prédisent statistiquement la suite la plus probable d'une séquence. GPT-4 d'OpenAI, Claude d'Anthropic, Gemini de Google et CodeLlama de Meta représentent les modèles les plus utilisés dans ce contexte. Leur puissance tient à leur capacité à généraliser à partir d'exemples vus en entraînement, mais c'est aussi là que réside leur principale limite : ils peuvent produire des réponses confiantes mais incorrectes, un phénomène désigné sous le terme d'hallucination.

Définition des concepts clés

LLM — Large Language Model

Modèle de langage entraîné sur des milliards de tokens de texte et de code. Il fonctionne en prédisant la suite statistiquement la plus probable d'une séquence. GPT-4, Claude, Gemini et CodeLlama sont des exemples de LLMs spécialisés pour la génération de code.

IA Générative

Sous-domaine de l'intelligence artificielle capable de créer du nouveau contenu — texte, code, images — à partir d'une description en langage naturel, par opposition aux modèles discriminatifs qui classifient ou prédisent une valeur à partir de données existantes.

RAG — Retrieval-Augmented Generation

Technique qui enrichit dynamiquement le prompt envoyé au LLM avec du contexte récupéré localement : documentation du projet, base de code existante, historique des commits. Cela permet à l'outil de répondre en tenant compte du code réel du projet, réduisant ainsi les hallucinations et améliorant la pertinence des suggestions. Cursor utilise cette approche nativement.

Hallucination

Phénomène par lequel un LLM génère une réponse syntaxiquement correcte et formulée avec assurance, mais factuellement incorrecte : invocation d'une fonction inexistante, référence à une API obsolète, logique algorithmique erronée. Le danger est maximal lorsque le développeur accepte la suggestion sans la vérifier, car le modèle n'a pas conscience de ses propres erreurs.

Prompt Engineering

Art de formuler les instructions données au LLM pour obtenir des résultats optimaux. Comprend des techniques comme le chain-of-thought, qui consiste à décomposer le raisonnement en étapes, le few-shot prompting, qui fournit des exemples concrets, ou le role prompting, qui attribue un rôle spécifique au modèle. Cette compétence devient essentielle pour tout développeur souhaitant exploiter efficacement ces outils.

Méthodologie de veille

Pourquoi cette veille ?

En tant qu'étudiant en BTS SIO option SLAM, l'intelligence artificielle générative concerne directement ma formation et ma future carrière de développeur. En l'espace de deux ans, ces outils sont passés du statut d'expérimentation à celui de compétence attendue sur le marché du travail. Comprendre leur fonctionnement, mesurer leur impact réel et identifier leurs limites n'est plus une option, mais une nécessité professionnelle.

L'enjeu de cette veille est donc double. Il s'agit d'une part de comprendre comment ces outils transforment concrètement le quotidien du développeur, et d'autre part d'identifier les risques et les questions éthiques qu'ils soulèvent, afin d'adopter une posture critique et éclairée face à leur utilisation.

Comment a-t-elle été réalisée ?

Cette veille a été menée en plusieurs étapes méthodiques visant à garantir la complémentarité et la pertinence des informations collectées.

Dans un premier temps, j'ai défini les axes de recherche en centrant l'analyse sur trois thématiques principales : le panorama des outils d'assistance au développement basés sur l'IA, l'impact de ces outils sur la productivité et la qualité du code, et enfin les limites techniques ainsi que les enjeux éthiques et professionnels.

Dans un second temps, j'ai procédé à la sélection des sources en privilégiant leur fiabilité et leur représentativité du secteur. Ont été retenus des rapports d'entreprises technologiques telles que GitHub, JetBrains et Google, des études de cabinets d'analyse reconnus comme Gartner et McKinsey, des publications académiques et des compte-rendus de conférences de référence telles que JSConf et Google I/O.

La troisième étape a consisté en l'agrégation des contenus via l'outil Feedly, qui a permis de centraliser les flux RSS de toutes ces sources dans un espace unique et de suivre en temps réel les nouvelles publications. Les mots-clés utilisés pour paramétrer les alertes étaient les suivants : "AI coding assistant", "GitHub Copilot", "LLM code generation" et "developer productivity AI".

Enfin, les données collectées ont été analysées et confrontées, en recoupant systématiquement les informations issues de sources aux intérêts divergents. Les chiffres de productivité avancés par GitHub ont ainsi été mis en perspective avec les études académiques indépendantes et les retours terrain de la communauté des développeurs, souvent plus nuancés.

Outils utilisés

Feedly : pourquoi et comment ?

Feedly revêt une importance cruciale pour mener à bien une veille, puisqu'il permet de rassembler plusieurs sources d'information au sein d'un même espace de travail. Plutôt que de consulter les sites un à un, l'agrégateur facilite le suivi des flux RSS et permet d'accéder en temps réel aux nouvelles publications, dès leur mise en ligne.

J'ai paramétré Feedly de façon à suivre des sources spécialisées dans l'intelligence artificielle appliquée au développement logiciel : sites spécialisés, blogs d'experts reconnus, publications académiques et annonces officielles des éditeurs d'outils. J'ai également configuré des mots-clés directement liés à mes axes de recherche.

L'un des grands avantages de Feedly est la possibilité d'organiser les articles dans différentes catégories thématiques, facilitant ainsi leur analyse et leur exploitation. L'outil s'est révélé précieux pour structurer la veille et obtenir une vision globale et continue des tendances du marché.

Les principaux outils d'IA observés

GitHub Copilot

Lancé en disponibilité générale en juin 2022, GitHub Copilot est l'assistant le plus répandu sur le marché. Développé par GitHub, filiale de Microsoft, en partenariat avec OpenAI, il s'intègre directement dans Visual Studio Code, les IDEs JetBrains et Neovim. Il propose des complétions de code ligne par ligne ou bloc entier, en s'appuyant sur le contexte du fichier ouvert et des fichiers adjacents. Une étude conduite par GitHub en 2024 mesure un gain de rapidité de 55 % sur les tâches répétitives.

Cursor

Cursor se positionne comme l'IDE de nouvelle génération, construisant nativement l'intelligence artificielle au cœur de l'expérience de développement. Il s'agit d'un fork de Visual Studio Code qui conserve l'ensemble de l'écosystème d'extensions tout en ajoutant des fonctionnalités IA avancées absentes des simples plugins. Son mode Agent peut modifier plusieurs fichiers simultanément en fonction d'un objectif décrit en langage naturel. Il supporte GPT-4, Claude et Gemini au choix de l'utilisateur. En 2025, il est considéré par la communauté comme l'outil le plus avancé du marché.

Gemini Code Assist

Anciennement connu sous le nom de Duet AI, rebaptisé Gemini Code Assist lors de Google I/O 2024, cet outil se distingue par son intégration profonde dans l'écosystème Google Cloud et sa fenêtre de contexte exceptionnelle pouvant atteindre un million de tokens avec Gemini 1.5 Pro, ce qui permet théoriquement d'injecter l'intégralité d'une base de code dans le contexte du modèle. Il est particulièrement performant sur les projets Android et Flutter grâce à sa formation

sur les SDKs Google.

Autres outils notables

L'écosystème s'est rapidement densifié avec l'apparition d'alternatives comme Codeium, disponible gratuitement et compatible avec plus de soixante-dix langages de programmation, Tabnine, pionnier du domaine depuis 2019 et disponible en version auto-hébergée pour les entreprises soucieuses de la confidentialité, Amazon CodeWhisperer, optimisé pour les services AWS, et Windsurf, concurrent direct de Cursor lancé fin 2024.

Sources et classement

Les sources utilisées ont fait l'objet d'un tri rigoureux selon leur degré de fiabilité et de pertinence par rapport aux axes de recherche définis. Elles se répartissent en plusieurs catégories complémentaires.

Rapports et études chiffrées

Le GitHub Octoverse 2024 constitue la référence principale sur l'adoption de Copilot et les tendances générales du développement logiciel. Il fournit des données chiffrées directement issues de la plateforme, mais doit être lu en tenant compte du biais inhérent à la position d'éditeur de GitHub.

Le Stack Overflow Developer Survey 2025, conduit auprès de plus de soixante-cinq mille développeurs dans le monde, constitue une référence indispensable pour mesurer l'adoption réelle des outils d'IA par les praticiens. Ses données brutes sont accessibles en open data, ce qui permet une analyse indépendante.

Le JetBrains Developer Ecosystem 2024 apporte un éclairage complémentaire sur les pratiques et les outils des développeurs, avec une attention particulière aux IDEs et aux workflows. L'étude McKinsey Technology Trends 2024 fournit pour sa part des estimations économiques sur les gains de productivité induits par l'IA, compris entre 20 et 45 % selon les types de tâches concernées.

Éditeurs et cabinets d'analyse

Le Gartner Hype Cycle 2024 permet de situer l'IA générative sur le cycle de maturité technologique. Sa position au pic des attentes exagérées invite à anticiper une phase de désillusion partielle avant une adoption mature et stabilisée.

L'Agence Nationale de la Sécurité des Systèmes d'Information, l'ANSSI, a publié des recommandations spécifiques sur les risques liés au code généré par intelligence artificielle : introduction de vulnérabilités, exfiltration potentielle de code sensible vers des serveurs cloud, et dépendance aux services tiers. Ces recommandations constituent une référence essentielle pour les entreprises françaises.

Communauté et conférences

Google I/O 2025 et GitHub Universe 2024 ont fourni les annonces officielles des éditeurs concernant les évolutions prévues de leurs outils respectifs. Ces sources, bien que naturellement orientées vers la promotion des produits, permettent d'identifier les directions technologiques envisagées.

Les plateformes communautaires telles que Hacker News, Dev.to et Reddit ont permis de collecter des retours d'expérience terrain, souvent plus nuancés et critiques que les communications officielles. Ces témoignages de praticiens constituent un contrepoids nécessaire aux discours promotionnels des éditeurs.

Synthèse des résultats

Impact sur la productivité

Les études conduites sur l'impact de l'IA générative sur la productivité des développeurs convergent vers des conclusions globalement positives, mais avec des nuances importantes selon le type de tâche considéré.

Les gains les plus significatifs sont observés sur les tâches répétitives et bien définies. La génération de code boilerplate, c'est-à-dire les structures de code répétitives comme les opérations CRUD, les getters et setters ou les structures de données standards, bénéficie pleinement de l'assistance de l'IA. La rédaction de tests unitaires, souvent négligée par les développeurs sous pression, est également prise en charge efficacement, avec une réduction estimée entre 40 et 60 % du temps consacré à cette activité. La génération de documentation, commentaires et fichiers README constitue un autre domaine où l'IA apporte une valeur ajoutée mesurable.

En revanche, les résultats sont sensiblement plus mitigés sur les tâches complexes. La conception d'architectures logicielles distribuées, la gestion des compromis techniques, l'optimisation des performances ou la compréhension du contexte métier spécifique à une organisation restent des domaines où l'expertise humaine demeure irremplaçable. L'IA reproduit ce qu'elle a appris ; pour des problèmes véritablement nouveaux, elle tend à halluciner ou à proposer des solutions génériques inadaptées au contexte particulier.

Impact sur la qualité du code

La question de la qualité du code généré par l'IA est plus complexe qu'il n'y paraît. Si la vitesse de production augmente significativement, la qualité du résultat dépend fortement de la compétence du développeur qui valide les suggestions.

Une étude conduite par l'Université de New York en 2023 a analysé du code généré par GitHub Copilot sur des scénarios impliquant des problématiques de sécurité. Les résultats indiquent que 40 % des suggestions contenaient des vulnérabilités, parmi lesquelles des injections SQL, des problèmes de gestion des entrées utilisateur et des références à des dépendances obsolètes contenant des failles connues.

Par ailleurs, la tendance des LLMs à suggérer des versions de packages basées sur leurs données d'entraînement, potentiellement datées, représente un risque supplémentaire. Un développeur peu vigilant peut ainsi introduire dans son projet des bibliothèques affectées par des vulnérabilités identifiées et documentées, mais inconnues du modèle au moment de son entraînement.

Analyse critique

Les enjeux éthiques et professionnels

Au-delà des aspects techniques, l'adoption massive de l'IA générative dans le développement logiciel soulève des questions éthiques et professionnelles qui méritent une attention particulière.

La question de la propriété intellectuelle du code généré fait l'objet de plusieurs procédures judiciaires aux États-Unis, notamment l'affaire Doe contre GitHub initiée en 2022. Les LLMs ont été entraînés sur des milliards de lignes de code open source placées sous des licences diverses, et des chercheurs ont démontré que certains modèles peuvent reproduire à l'identique des extraits de code protégés. La question de la paternité du code produit et de la conformité légale reste largement non résolue.

L'impact sur la formation des développeurs juniors constitue un autre sujet de préoccupation majeur au sein de la communauté. En acceptant des suggestions sans les comprendre, un développeur débutant peut livrer du code fonctionnel tout en restant incapable d'en expliquer les mécanismes, ce qui se révèle problématique lors du débogage, de la maintenance ou de la revue de code. La question posée est fondamentale : comment apprend-on véritablement à programmer si l'outil écrit le code à notre place ?

La confidentialité du code transmis aux services cloud représente également un risque concret pour les entreprises. L'incident Samsung survenu en 2023, lors duquel des ingénieurs ont involontairement partagé du code propriétaire et des données confidentielles via ChatGPT, illustre concrètement ce danger. Si des solutions d'hébergement local existent, comme Tabnine Enterprise ou Codeium Enterprise, elles restent moins performantes que leurs équivalents cloud et nécessitent une infrastructure dédiée.

Les perspectives pour 2025-2026

Plusieurs tendances se dessinent pour les années à venir et méritent d'être suivies attentivement dans le cadre de cette veille.

L'émergence des agents IA autonomes constitue le prochain saut qualitatif annoncé. Au-delà de la simple complétion de code, ces agents sont capables de planifier, d'exécuter et de tester des tâches complexes de manière semi-autonome. Cursor Agent, GitHub Copilot Workspace et Devin, présenté comme le premier ingénieur logiciel entièrement autonome, représentent les premières manifestations concrètes de cette évolution.

Le fine-tuning sur code propriétaire devrait également se généraliser. Les grandes organisations vont de plus en plus entraîner leurs propres modèles sur leur base de code interne, leurs conventions de développement et leur documentation métier, pour obtenir des suggestions ultra-contextualisées et conformes à leurs standards internes.

Enfin, l'intégration progressive de l'IA générative dans les pipelines d'intégration et de déploiement continus est déjà en cours. La révision automatique de code, la détection de vulnérabilités et la génération de documentation à chaque commit représentent des cas d'usage concrets qui se déploient dès maintenant dans les équipes les plus avancées.

Conclusion

À l'issue de cette veille, plusieurs conclusions s'imposent avec clarté. L'intelligence artificielle générative ne remplace pas le développeur, mais elle reconfigure profondément son rôle. Les tâches mécaniques et répétitives sont prises en charge efficacement, libérant du temps pour des activités à plus forte valeur ajoutée : conception, architecture, revue de qualité, compréhension du domaine métier.

Les gains de productivité sont réels et documentés, mais ils sont asymétriques. Les développeurs expérimentés, qui savent évaluer, corriger et adapter les suggestions de l'IA, bénéficient davantage de ces outils que les développeurs débutants. Ces derniers risquent d'accepter du code sans le comprendre, introduisant potentiellement des bugs, des vulnérabilités ou une dette technique sans en avoir conscience.

Les enjeux éthiques et légaux soulevés par cette évolution sont réels et encore largement non résolus. La propriété intellectuelle du code généré, la responsabilité en cas de faille de sécurité, la confidentialité des données transmises aux modèles cloud et l'impact sur la formation des juniors constituent autant de questions ouvertes qui appellent une adoption réfléchie et encadrée de ces technologies.

La vraie question posée par cette veille n'est pas de savoir si l'intelligence artificielle va remplacer les développeurs. Elle est de savoir si les développeurs qui maîtrisent ces outils vont remplacer ceux qui ne les maîtrisent pas. En BTS SIO SLAM, apprendre à collaborer efficacement avec l'IA — tout en conservant une compréhension solide des fondamentaux algorithmiques, de la sécurité et de l'architecture logicielle — devient une compétence professionnelle à part entière.

Cette veille continuera d'évoluer au rythme d'un domaine qui se transforme à une vitesse sans précédent dans l'histoire du développement logiciel. Le suivi régulier des publications académiques, des annonces des éditeurs et des retours d'expérience de la communauté restera indispensable pour maintenir une vision à jour et critique de ces technologies.