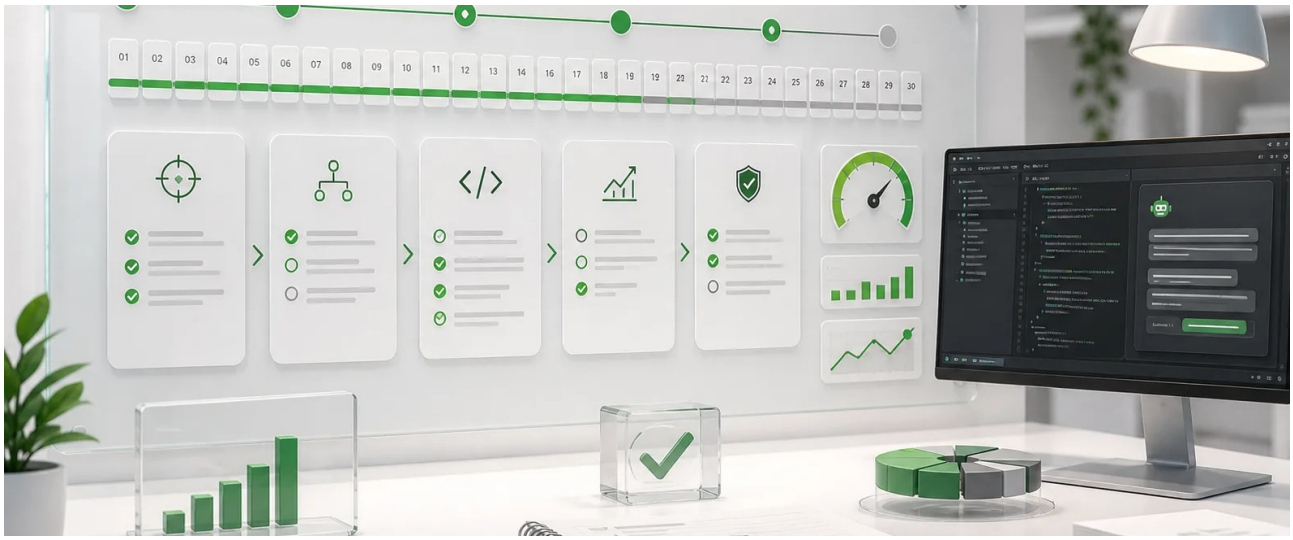
Notilis Tech

Agent de codage IA : concevoir un pilote ROI de 30 jours

Concevoir un pilote ROI de 30 jours pour décider d'arrêter, d'ajuster ou d'étendre l'usage d'un agent de codage IA.

2026-05-03 · Pilotage · ROI · Mesure



Résumé

Un agent de codage IA ne devrait pas être déployé sur la base d'une intuition, d'un effet de mode ou d'un simple enthousiasme utilisateur. Pour un décideur, la bonne question n'est pas de savoir si l'outil impressionne pendant une démonstration, mais s'il améliore réellement un workflow de développement dans le contexte précis de l'entreprise, avec un coût maîtrisé, une qualité au moins stable et un risque acceptable.

C'est tout l'intérêt d'un pilote ROI de 30 jours. Bien conçu, ce pilote n'est pas une expérimentation ouverte où chacun teste ce qu'il veut. C'est une machine de décision. Il sert à formuler une hypothèse, choisir un périmètre, mesurer une situation de départ, suivre les coûts et les résultats, intégrer la review humaine, puis conclure explicitement : faut-il arrêter, ajuster ou étendre ?

L'objectif n'est donc pas de prouver que l'IA est bonne ou mauvaise. L'objectif est plus opérationnel : comprendre ce qu'elle change dans votre système réel de delivery.

Sommaire

Contexte de lecture

Audience

Cet article s’adresse aux DAF, DSI, CTO, responsables produit, responsables delivery et managers techniques qui veulent préparer un pilote mesurable avant toute généralisation.

Ce que cet article couvre

Il se concentre sur une question concrète : comment construire, en 30 jours, un pilote assez sérieux pour produire une décision exploitable.

Ce que cet article ne couvre pas

Il ne propose pas un comparatif fournisseur, une grille officielle de tarification, une promesse de ROI ou une politique sécurité complète.

Pourquoi un pilote doit produire une décision

Beaucoup d'organisations lancent leurs premiers essais d'IA générative avec de bonnes intentions, mais avec un cadrage insuffisant. On ouvre l'accès à quelques utilisateurs motivés, on observe les usages, on collecte des retours, puis on tente d'en déduire une conclusion. Le problème est que ce type d'expérimentation produit surtout des anecdotes.

On obtient généralement trois types de signaux :

- des témoignages positifs ou négatifs ;
- des chiffres d'usage ;
- une perception générale d'enthousiasme ou de résistance.

Ces signaux ne sont pas inutiles. Ils disent quelque chose de l'adoption, de la friction et de la maturité des équipes. Mais ils ne suffisent pas pour décider d'un investissement, encore moins d'une généralisation.

Un pilote de décision doit produire autre chose. Il doit rendre visibles :

- le périmètre testé ;
- la situation de départ ;
- les workflows concernés ;
- les coûts engagés ;
- les effets observés sur le delivery, la qualité et le risque ;
- la décision finale.

Autrement dit, il doit permettre de répondre à une question simple : l'agent améliore-t-il un workflow réel, dans notre contexte réel, avec un coût et un niveau de risque acceptables ?

Comme le rappelle DORA, l'IA amplifie le système existant : le pilote doit donc mesurer le workflow réel, pas seulement l'adoption de l'outil.[1]

Formuler une hypothèse ROI avant de commencer

La première erreur consiste à lancer le pilote avant d'avoir écrit ce que l'on cherche à vérifier. Un pilote ROI doit commencer par une hypothèse simple, formulée avant l'ouverture des accès.

La structure peut être la suivante :

> **Si nous utilisons un agent de codage IA sur [workflow],**
> **nous devons améliorer [métrique],**
> **sans dégrader [qualité / risque],**
> **pour un coût inférieur à [seuil].**

Cette phrase paraît basique, mais elle oblige à cadrer le débat. Elle force à dire quel workflow est concerné, quelle métrique sera observée, quel niveau de qualité doit être préservé et quel coût reste acceptable.

Les workflows candidats peuvent être variés. Un pilote peut viser :

- la réduction du temps de correction de bugs reproductibles ;

- l'accélération de la préparation de pull requests ;
- l'ajout de tests sur des zones bornées ;
- la réduction du rework sur des tickets récurrents ;
- l'amélioration de la documentation technique liée aux changements.

L'important n'est pas de choisir le cas le plus spectaculaire. L'important est de choisir un cas mesurable.

Une hypothèse claire empêche le pilote de devenir un fourre-tout où l'on additionne des usages trop différents pour être comparés. La valeur se mesure sur le workflow :

- temps gagné ;
- qualité maintenue ;
- review facilitée ;
- rework réduit ;
- dette traitée ;
- capacité libérée.

Définir un seuil économique simple

Un pilote ROI doit avoir un seuil économique, même approximatif, pour éviter une discussion vague.

Par exemple, si le pilote coûte 600 par mois et que le coût horaire complet moyen est de 75, le seuil de break-even est de 8 heures par mois : 600 / 75 = 8 h/mois.

Ce chiffre ne prouve pas que l'usage est rentable ; il donne simplement au pilote une cible minimale à confronter aux workflows observés.

Choisir peu d'utilisateurs, mais avec un fort levier

Pour un premier cycle de 30 jours, il est préférable de commencer petit. Un pilote limité à 3 à 5 utilisateurs est souvent plus utile qu'un test ouvert à une population plus large, parce qu'il permet de mieux comprendre ce qui se passe réellement.

Les participants doivent être choisis pour leur capacité à créer du levier sur les workflows testés, mais aussi pour leur capacité à documenter leurs résultats. Le bon profil n'est pas seulement celui qui utilise beaucoup l'outil. C'est celui qui sait expliquer :

- pourquoi il l'a utilisé ;
- sur quelle tâche ;
- avec quel résultat ;
- quel niveau de review ;
- quelles limites.

Les profils pertinents peuvent inclure :

- des développeurs seniors ;

- des tech leads ;
- des architectes ;
- des profils QA automatisation ;
- des responsables support niveau 3 ;
- des PM techniques travaillant sur des sujets très techniques.

Le titre exact compte moins que la capacité à intervenir sur des workflows importants et vérifiables.

Ce choix restreint a un autre avantage : il réduit le bruit. Sur un pilote large, les écarts de maturité, de contexte et de discipline d'usage peuvent rendre les résultats illisibles. Sur un pilote restreint, il devient possible de relier chaque usage à une tâche, chaque tâche à un résultat, et chaque résultat à une décision.

Il faut aussi choisir des participants capables d'accepter la discipline du pilote. Un utilisateur enthousiaste mais incapable de documenter ses usages produira moins de valeur de décision qu'un utilisateur plus sobre mais plus rigoureux.

Sélectionner deux ou trois workflows vérifiables

Le périmètre du pilote doit être suffisamment étroit pour être instrumenté. Deux ou trois workflows bien choisis valent mieux qu'une longue liste de cas d'usage hétérogènes.

Les bons candidats sont ceux dont le résultat peut être relu, testé ou comparé. Par exemple :

- corriger un bug reproductible ;
- préparer une pull request avec résumé de risque ;
- ajouter des tests sur une zone bornée ;
- produire ou mettre à jour une documentation technique liée à un changement réel ;
- réaliser un refactoring local sous contraintes explicites ;
- préparer une root-cause analysis après incident.

À l'inverse, certains sujets sont de mauvais candidats pour un premier pilote :

- une grande feature mal cadrée ;
- une modification transverse sans critères d'acceptation ;
- un changement critique sans test ;
- une exploration libre sans baseline.

Ces sujets risquent de produire des résultats difficiles à interpréter.

Si le périmètre est trop flou, il devient impossible de savoir si l'outil a réellement aidé ou si l'équipe a simplement déplacé l'effort ailleurs.

Le bon workflow n'est donc pas forcément celui qui donne la meilleure démonstration. C'est celui dont le résultat peut être vérifié sans ambiguïté.

Cette prudence est d'autant plus importante que les effets de l'IA sur la productivité ne sont pas toujours intuitifs. METR a publié en 2025 un essai randomisé sur des développeurs open source expérimentés

travaillant sur leurs propres dépôts. Dans ce contexte précis, les tâches réalisées avec IA ont pris plus de temps, alors même que les participants avaient une perception positive de l'outil.[2] Cette étude ne doit pas être lue comme une conclusion générale contre l'IA. Elle doit plutôt servir d'alerte méthodologique : le ressenti utilisateur ne suffit pas. Il faut mesurer.

Construire une baseline avant l'usage

La baseline est ce qui protège le pilote contre l'illusion de vitesse. Sans situation de départ, tout gain perçu reste fragile. Une équipe peut avoir l'impression d'aller plus vite parce que la production initiale est accélérée, tout en découvrant plus tard que la review, les corrections ou les tests ont absorbé le temps gagné.

La baseline n'a pas besoin d'être parfaite. Elle doit simplement être suffisante pour éviter une décision fondée uniquement sur le ressenti. Dans la plupart des cas, quelques tickets ou pull requests comparables suffisent pour établir un ordre de grandeur.

Métrique	Source possible de baseline
Temps moyen de correction	10 tickets comparables
PR cycle time	10 pull requests comparables
Temps de review	Historique des PR et des relecteurs
Rework	Nombre d'allers-retours ou de reprises
Défauts passés en production	Incidents ou bugs post-livraison
Dette traitée	Tickets de dette ou refactoring
Documentation	Changements documentés ou non

Cette baseline doit être construite avant le pilote, pas après. La reconstituer a posteriori expose à un biais de sélection : on risque de choisir les exemples qui confirment l'impression dominante.

L'objectif n'est pas de produire une étude statistique parfaite. L'objectif est de disposer d'un point de comparaison assez crédible pour arbitrer.

Dérouler le pilote sur 30 jours

Un pilote de 30 jours a besoin d'un rythme. Sinon, il devient une période d'essai molle, sans point de contrôle, sans responsable et sans décision.

J-5 à J0 : préparation

Avant l'ouverture des accès, l'équipe doit :

- écrire l'hypothèse ROI ;
- choisir les utilisateurs pilotes ;
- définir les workflows autorisés ;
- fixer le budget maximum ;

- établir la baseline ;
- rappeler les règles de sécurité ;
- définir les règles de review ;
- écrire les critères arrêter / ajuster / étendre.

Cette préparation n'a pas besoin d'être lourde. Elle doit simplement éviter que les 30 jours servent à découvrir ce qui aurait dû être décidé avant.

Semaine 1 : instrumentation

La première semaine doit valider que les usages sont bien rattachés à des workflows, que les coûts sont visibles et que les participants savent documenter leurs actions.

Le bon objectif n'est pas encore de maximiser le gain. Il est de vérifier que le pilote est observable.

Si les utilisateurs ne savent pas dire sur quel workflow ils utilisent l'agent, le pilote est déjà en difficulté.

Semaine 2 : production contrôlée

La deuxième semaine doit produire les premiers résultats comparables : tickets corrigés, PR préparées, tests ajoutés, documentation mise à jour, refactoring borné.

À ce stade, il faut suivre :

- le temps passé ;
- le temps estimé gagné ;
- le temps de review ;
- les reprises ;
- les irritants.

Le pilote doit aussi noter les cas où l'agent n'a pas aidé. Ces cas sont utiles, car ils permettent de réduire le périmètre ou de modifier les consignes.

Semaine 3 : mid-review

La troisième semaine doit comporter une revue intermédiaire. C'est souvent à ce moment que les signaux deviennent lisibles : certains workflows fonctionnent, d'autres non ; certains prompts ou modes d'usage créent de la valeur, d'autres produisent du bruit ; certains utilisateurs documentent bien, d'autres moins.

La mid-review doit permettre d'ajuster sans attendre J+30. Elle peut :

- réduire un workflow trop large ;
- renforcer une règle de review ;
- plafonner un usage coûteux ;
- demander une meilleure documentation.

L'objectif n'est pas de figer le pilote. L'objectif est de garder une expérimentation contrôlée.

Semaine 4 : consolidation

La quatrième semaine doit consolider les données, comparer à la baseline, documenter les risques et préparer la scorecard de décision.

Ce n'est pas le moment de lancer de nouveaux usages. C'est le moment de comprendre ce que les usages observés permettent de décider.

Un bon pilote de 30 jours se termine par une décision, pas par une liste d'idées supplémentaires.

Suivre les coûts, le delivery, la qualité et la review humaine

Pendant les 30 jours du pilote, il faut suivre plusieurs familles d'indicateurs. Se concentrer uniquement sur le coût ou uniquement sur la vitesse donnerait une vision incomplète.

La première famille concerne le coût. Il faut connaître :

- le budget autorisé ;
- le coût réel ;
- le coût par workflow ;
- les dépassements éventuels ;
- le nombre d'utilisateurs actifs.

Lorsque les fournisseurs proposent des crédits, des limites ou des contrôles de dépenses, ces mécanismes doivent simplement être intégrés au cadrage du pilote.

La deuxième famille concerne le delivery. On peut suivre :

- le temps de correction ;
- le lead time ;
- le PR cycle time ;
- le nombre d'allers-retours ;
- le nombre de tickets terminés.

Ces indicateurs permettent de voir si le workflow avance réellement plus vite, ou si la vitesse apparente est compensée par des reprises ultérieures.

La troisième famille concerne la qualité. Les métriques utiles incluent :

- les défauts passés en production ;
- les tests ajoutés ;
- le rework ;
- la dette traitée ;
- la qualité perçue par les relecteurs.

Une amélioration de vitesse qui dégrade la qualité n'est pas forcément un gain. Elle peut simplement reporter le coût vers la production, le support ou la maintenance.

La quatrième famille concerne la review humaine : son coût doit être mesuré, mais son cadre complet relève surtout de la governance traitée dans l'article 4.

Le point central est donc le suivant : le pilote doit mesurer le workflow complet, pas seulement le moment où l'agent produit du code.

Documenter les risques au lieu de les masquer

Un pilote sérieux doit aussi documenter ce qui ne marche pas. C'est une condition de maturité, pas un signe d'échec.

Les risques à suivre incluent :

- l'usage hors workflow autorisé ;
- un coût variable non expliqué ;
- une baisse de qualité ;
- un rework déplacé vers les relecteurs ;
- l'exposition de données ou de contextes sensibles ;
- une dépendance excessive à un utilisateur expert ;
- un gain déclaré mais non réalloué.

Ces risques ne condamnent pas nécessairement le pilote. Ils indiquent ce qu'il faut corriger avant une extension.

Si le coût est trop variable, il faudra renforcer les contrôles budgétaires.

Si le rework se déplace vers les relecteurs, il faudra revoir :

- les consignes de production ;
- les critères d'acceptation ;
- les règles de review.

Si seuls les profils les plus seniors obtiennent des gains, il faudra éviter de généraliser trop vite à des populations moins préparées.

Un pilote utile n'est donc pas celui qui ne remonte aucun problème. C'est celui qui rend les problèmes visibles assez tôt pour éviter une généralisation coûteuse.

Décider explicitement à J+30

À la fin des 30 jours, la décision doit être prise selon des critères écrits à l'avance. Sans critères explicites, le pilote risque de se prolonger indéfiniment ou de déboucher sur une décision politique plutôt qu'opérationnelle.

La décision peut être structurée en trois options : arrêter, ajuster ou étendre.

Décision	Conditions typiques
Arrêter	Seuil non atteint, qualité dégradée, coût de review trop élevé, risque non contrôlé
Ajuster	Workflows trop larges, mauvais profils, métriques insuffisantes, budget mal calibré
Étendre	Gain mesuré, qualité stable ou meilleure, coût contrôlé, temps réalloué

Un pilote négatif n'est pas un échec. C'est une information qui évite une extension coûteuse. À l'inverse, un pilote positif ne signifie pas nécessairement qu'il faut généraliser immédiatement à toute l'organisation. Il peut justifier une extension progressive, sur des workflows similaires, avec les mêmes règles de mesure et de review.

La bonne question à J+30 n'est donc pas :

> **Les utilisateurs ont-ils aimé l'outil ?**

La bonne question est :

> **Quelle décision ce pilote rend-il possible ?**

La scorecard minimale du pilote

Pour éviter de disperser les résultats, il est utile de résumer le pilote dans une scorecard très courte :

- coût engagé ;
- effet observé sur le workflow ;
- qualité ou risque ;
- décision proposée.

La version complète du dashboard exécutif est détaillée dans l'article Construire un dashboard exécutif ROI ; ici, l'objectif est seulement de préparer une décision à J+30 sans noyer le pilote dans le reporting.

La scorecard doit aussi comporter une ligne narrative courte : "ce que nous avons appris". Cette ligne est importante parce que toutes les décisions ne sont pas purement numériques. Parfois, le pilote montre :

- qu'un workflow n'est pas encore assez standardisé ;
- que la review humaine est le vrai goulot ;
- que l'outil est utile, mais seulement pour des profils seniors.

Ces enseignements doivent être écrits.

Checklist opérationnelle

Avant le lancement :

- [ ] l'hypothèse ROI doit être écrite ;
- [ ] les utilisateurs pilotes identifiés ;
- [ ] les workflows bornés ;
- [ ] la baseline disponible ;
- [ ] le budget plafonné ;
- [ ] les règles de review humaine explicites ;
- [ ] les critères arrêter / ajuster / étendre définis.

Pendant le pilote :

- [ ] chaque usage doit être relié à un workflow ;
- [ ] le coût doit être suivi ;
- [ ] le temps de review mesuré ;
- [ ] le rework documenté ;
- [ ] les défauts passés en production surveillés ;
- [ ] le temps gagné associé à une destination concrète.

Après 30 jours :

- [ ] vérifier si le seuil fixé a été atteint ;
- [ ] vérifier si la qualité est stable, meilleure ou dégradée ;
- [ ] vérifier si le coût est contrôlé ;
- [ ] vérifier si le temps est réellement réalloué ;
- [ ] vérifier si une décision écrite peut être prise.

Cette checklist peut sembler administrative. Elle est pourtant ce qui transforme un test d'outil en décision de management.

Conclusion

Un pilote d'agent de codage IA ne doit pas être traité comme une simple période d'essai. Pour produire de la valeur, il doit être conçu comme un dispositif de décision :

- hypothèse claire ;
- périmètre borné ;
- baseline ;
- coûts suivis ;
- review humaine intégrée ;
- risques documentés ;

- conclusion explicite à J+30.

Cette discipline peut paraître contraignante, mais elle évite deux erreurs symétriques :

- généraliser trop vite sur la base de quelques démonstrations impressionnantes ;
- rejeter trop vite l'outil parce que le premier usage était mal cadré.

Le bon pilote ne cherche pas à confirmer une croyance. Il cherche à réduire l'incertitude. À la fin des 30 jours, la question n'est pas de savoir si l'IA est prometteuse en général. La question est de savoir si, pour vos workflows, vos équipes, vos contraintes de qualité et votre structure de coûts, elle mérite d'être arrêté, ajusté ou étendu.

Articles liés

- Évaluer un agent de codage IA comme une option de capacité.
- Transformer les heures gagnées en résultats business.
- Contrôler risque, budget et review humaine.

Références

1. DORA / Google Cloud, “DORA Research: 2025 — State of AI-assisted Software Development”, consulté le 2026-06-07. https://dora.dev/research/2025/dora-report/

2. METR, “Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity”, publié le 2025-07-10, consulté le 2026-06-07. https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/