KAMANGA

LIVRE BLANC · 28 PAGES · KAMANGA SOFTWARE CRAFTSMAN

Le nouveau développeur 2026 : de codeur à administrateur IA.

L'IA code aussi bien qu'un développeur moyen. Souvent mieux. L'expert Java, PHP ou fullstack disparaît. Mais quand le code généré casse, expose des données ou viole une licence, c'est un humain qui répond. Ce livre blanc dresse l'état des lieux, puis donne le référentiel du nouveau rôle : garantir le process de développement de bout en bout.

Hervé Muludiki · 25 ans : développeur, tech lead, engineering leader, coach craft

kamanga.fr · Édition 2026

KAMANGA

Sommaire

Chaque chapitre peut se lire indépendamment : commencez par les sujets les plus proches de votre situation

COMMENT LIRE CE LIVRE BLANC

Pressé ? L'executive summary (pages 3 et 4) contient la thèse, le référentiel et les 3 décisions à prendre. Le reste du document est la preuve, le détail, et le plan d'exécution.

SYNTHÈSE · 1 / 2

La thèse en cinq points.

01

L'IA a banalisé l'écriture du code

Générer, traduire, déboguer, migrer, tester : sur la majorité des tâches courantes, une IA bien outillée fait le travail d'un développeur moyen, plus vite, et souvent mieux.

02

Ce qui reste rare : la responsabilité

Quand le code généré casse la production, expose des données ou embarque une licence interdite, ce n'est pas l'IA qui répond. C'est votre entreprise, et le développeur qui a validé.

03

Le développeur change de nature

Il n'est plus l'expert d'une technologie. Il devient l'administrateur IA : celui qui détient les droits, qui autorise ou bloque chaque action des agents, et qui garantit le process de bout en bout.

04

Son référentiel : le SDLC complet

Six phases : cadrage, spécification, implémentation, vérification, livraison, amélioration continue. L'IA exécute, l'humain garantit. Chaque phase absente se paie en dette, en incidents ou en risque juridique.

05

La transition est un process, pas une révélation

Cartographier le process réel, encoder les pratiques dans l'outillage, installer les contrôles, boucler avec le métier : 30 jours suffisent pour amorcer le pivot (chapitre 9, page 25).

> *« Jamais autant de code n'a été produit. Jamais il n'y a eu aussi peu de monde pour en répondre. »*

KAMANGA

Executive summary

Le référentiel, la grille de maturité, les 3 décisions

SYNTHÈSE · 2 / 2

Le pipeline que le développeur 2026 administre

Humain décide, arbitre, autorise, répond — IA exécute la production

PHASE 1	PHASE 2	PHASE 3	PHASE 4	PHASE 5	PHASE 6
Cadrage	**Spécification**	**Implémen-tation**	**Vérification**	**Livraison**	**Amélioration**
IA : analyse, explore l'existant	IA : rédige la spec, découpe	IA : code et teste	IA : review multi-angles	IA : prépare PR, changelog	IA : ADRs, doc, nettoyage
Humain : pose le vrai problème	Humain : valide avec le métier	Humain : fixe le cadre, les droits	Humain : impose les gates bloquants	Humain : merge et déploie	Humain : capitalise les décisions

La grille de maturité en un coup d'œil

NIVEAU	VOTRE PROCESS	RISQUE PRINCIPAL
1 · Artisanal	Tout est dans les têtes	Dépendance aux individus
2 · Documenté	Pratiques écrites, application aléatoire	Théâtre de process
3 · Outillé	CI, conventions, gates actifs	Angles morts sur l'IA
4 · Automatisé	L'IA exécute les phases	VITESSE SANS CONTRÔLE
5 · Garanti	IA + contrôles + humain responsable	LA CIBLE

Les 3 décisions à prendre ce trimestre si vous êtes CTO

1. **Cesser d'évaluer les développeurs au volume de code.** Les évaluer sur la tenue du process et de ses contrôles.
2. **Mettre la conformité dans la CI** (licences, sécurité, provenance), pas dans la due diligence.
3. **Nommer des administrateurs IA** : responsables des droits accordés aux agents et de la garantie du process.

INTRODUCTION

2026, le constat.

En 2026, plus personne ne débat sérieusement de la question "l'IA sait-elle coder ?". Elle sait. Les assistants et les agents de développement produisent du code de niveau professionnel sur la majorité des tâches courantes : créer un module, intégrer une API, migrer un framework, écrire les tests, corriger un bug reproduit.

Dans les équipes que j'accompagne, la bascule s'est faite en moins de deux ans. Le développeur qui tape l'intégralité de son code est devenu l'exception. Et le marché suit : les fiches de poste "développeur React" ou "développeur Java" se vident de leur sens, remplacées par des intitulés flous qui cherchent tous la même chose sans savoir la nommer.

> *« Jamais autant de code n'a été produit. Jamais il n'y a eu aussi peu de monde pour en répondre. »*

Le volume de code explose. La capacité à le produire n'est plus rare. Ce qui est devenu rare, c'est la capacité à garantir que ce qui est produit est correct, maintenable, sécurisé et légalement irréprochable.

C'est exactement là que se reconstruit le métier de développeur. Pas dans la nostalgie du code tapé à la main. Pas dans la course aux outils. **Dans la responsabilité.**

CE QUE CE LIVRE BLANC N'EST PAS

- Un tutoriel d'outils IA : vous n'y trouverez aucun comparatif d'assistants.
- Un manifeste anti-IA, ni un discours d'évangéliste.
- Un guide de prompt engineering.

Ce qu'il est : le référentiel du nouveau rôle. Pourquoi l'ancien disparaît (chapitres 1 et 2), ce que le nouveau recouvre exactement (chapitres 3 à 6), comment l'entreprise doit se réorganiser autour de lui (chapitre 7), les pièges de la transition (chapitre 8), et par quoi commencer dès ce mois-ci (chapitre 9).

CHAPITRE 01 · LE DEUIL

La fin du développeur technologue.

1.1 Ce que l'IA a banalisé

Pendant vingt ans, la valeur d'un développeur s'est mesurée à sa maîtrise d'une technologie. On recrutait "un dev Java", "un dev PHP", "un fullstack React/Node". Cette expertise était une rente : longue à acquérir, difficile à vérifier, chère à remplacer. **Cette rente a fondu.** Une IA outillée fait aujourd'hui, en quelques minutes :

- écrire un module complet à partir d'une spécification claire ;
- traduire une base de code d'un langage vers un autre ;
- écrire les tests unitaires et d'intégration d'un code existant ;
- diagnostiquer et corriger un bug reproductible ;
- réaliser une migration de framework, documenter un legacy.

La syntaxe, les idiomes, les API : tout ce qui séparait "je connais Java" de "je ne connais pas Java" est disponible à la demande. **La technologie n'est plus une barrière à l'entrée. C'est un outillage.**

1.2 Ce que l'IA n'assume pas

Relisez la liste ci-dessus : chaque tâche commence par une entrée claire (une spec, un bug reproduit). C'est l'angle mort. L'IA exécute remarquablement, mais elle n'assume rien :

- **Décider quoi construire**, et surtout quoi ne pas construire.
- **Arbitrer les compromis** : dette acceptable, performance, coût, délai.
- **Répondre des conséquences** : le bug en production, la faille exploitée, la licence violée. Une IA ne passe pas en comité des risques et ne porte aucune responsabilité civile ou pénale.

Quand le code généré provoque un incident, la question n'est jamais "quelle IA a écrit ça ?". C'est toujours **"qui a validé ça ?"**.

1.3 Le piège mortel : se vendre (et évaluer) au volume de code

Le danger, pour un développeur comme pour un CTO, c'est de garder les métriques de l'ancien monde : lignes de code, vélocité de production, maîtrise d'un langage. Elles mesurent une capacité que l'IA fournit désormais à coût marginal.

	DÉVELOPPEUR 2020	DÉVELOPPEUR 2026
Compétence centrale	Maîtrise d'une technologie	Maîtrise du process de développement
Unité de valeur	Le code écrit	Le système de production garanti
Livrable principal	Des features	Un flux fiable d'intentions vers la production
Ce qu'on évalue	Volume, vélocité, stack	Tenue du process, contrôles, incidents évités
Rapport à l'IA	Outil d'autocomplétion	Équipe d'agents à administrer

ENCADRÉ · DU TECHNOLOGUE AU TECHNOCRATE

Trois mots, trois époques. Le **technicien** exécute la technique. Le **technologue** est l'expert d'une technologie : c'est lui qui disparaît. Le **technocrate**, au sens littéral, est celui qui gouverne par la maîtrise de la technique, sans l'exécuter lui-même.

On moque le technocrate en politique ; en ingénierie logicielle, c'est exactement ce que devient le bon développeur : il ne tape plus le code, il gouverne la production par sa maîtrise de la technique. Ce livre blanc lui donne un nom plus précis : **l'administrateur IA**.

> *« On n'évalue pas un chef d'orchestre au nombre de notes qu'il joue. »*

CHAPITRE 02 · LES FONDATIONS

Retour aux origines : le Software Craftsmanship.

2.1 Le métier n'a jamais été "taper du code"

Pour comprendre ce qui naît, il faut se rappeler ce que le métier a toujours été. Bien avant l'IA, le mouvement Software Craftsmanship affirmait déjà que produire du logiciel professionnel ne se réduit pas à écrire du code qui marche.

Le manifeste du Software Craftsmanship (2009) tient en quatre déplacements :

1. *"Not only working software, but also well-crafted software."*
   Pas seulement du logiciel qui fonctionne : du logiciel bien construit.
2. *"Not only responding to change, but also steadily adding value."*
   Pas seulement réagir au changement : ajouter de la valeur de façon continue.
3. *"Not only individuals and interactions, but also a community of professionals."*
   Pas seulement des individus : une communauté de professionnels qui se transmettent les pratiques.
4. *"Not only customer collaboration, but also productive partnerships."*
   Pas seulement collaborer avec le client : construire des partenariats productifs.

> **POURQUOI CE DÉTOUR EN 2026**
> Ces quatre piliers n'ont pas vieilli. Ils ont changé d'objet : le standard ne s'applique plus à ce que vous tapez, mais à ce que vos IA produisent. Page suivante : la relecture pilier par pilier.

Chapitre 2 · suite

Les quatre piliers relus, et la bibliothèque du craftsman

2.2 Les quatre piliers, relus en 2026

PILIER 1

Well-crafted software

Le standard de qualité n'est plus ce que vous vous imposez en codant : c'est ce que vous imposez à la production des IA. Le craft devient un référentiel d'exigence appliqué à un flux que vous ne tapez plus.

PILIER 2

Steadily adding value

La valeur ne se mesure plus en code produit mais en flux : des intentions métier qui arrivent en production, régulièrement, sans casse.

PILIER 3

Community of professionals

Transmettre ne suffit plus. Une pratique qui compte est une pratique encodée : template, hook, gate de CI, skill d'agent. La communauté écrit ses standards dans l'outillage.

PILIER 4

Productive partnerships

Le développeur redevient un partenaire du métier, pas un exécutant de tickets. C'est l'objet du chapitre 5.

2.3 L'IA réalise la promesse du craft, à une condition

Le craftsmanship a toujours buté sur le même mur : le temps. Bien faire coûtait cher, et la pression du delivery rognait d'abord sur la qualité. L'IA lève ce mur : tests, documentation, refactoring peuvent désormais être produits systématiquement.

Mais une promesse réalisée par une machine sans standard ne vaut rien. L'IA produit du bien-fait si quelqu'un définit ce que "bien fait" veut dire, le vérifie, et refuse ce qui ne l'est pas. **Ce quelqu'un, c'est le nouveau développeur.**

LA BIBLIOTHÈQUE DU CRAFTSMAN 2026

- *The Software Craftsman* (Sandro Mancuso) : le professionnalisme comme posture.
- *The Developer's Journey* : la trajectoire comme chemin de maîtrise, pas comme suite de stacks.
- *Apprenticeship Patterns* (Hoover, Oshineye) : apprendre le métier comme un artisanat.
- *The Pragmatic Programmer* (Thomas, Hunt) : les invariants du bon jugement technique.
- *Clean Coder* (Robert C. Martin) : dire non, s'engager, répondre de son travail.

Aucun de ces livres ne parle d'un langage. Tous parlent de responsabilité.

CHAPITRE 03 · LE RÉFÉRENTIEL

Le nouveau rôle : l'administrateur IA.

> *Le génie logiciel a industrialisé la production de logiciel il y a longtemps : la norme ISO/IEC/IEEE 12207 décrit les processus du cycle de vie du logiciel (le SDLC, Software Development Life Cycle), de l'expression du besoin à la maintenance. Produire du logiciel, on sait faire, et on sait comment. La question de 2026 n'est plus "comment produire ?". C'est :* ***"qui garantit que c'est bien produit ?"***

3.1 Définition du rôle

Le développeur 2026 est **expert et garant du process de développement**, de l'idée à la production. Sa compétence n'est plus de connaître une technologie : c'est de savoir ce qu'est un développement bien mené, phase par phase, et de le faire respecter, par les humains comme par les IA. Son terrain de jeu n'est plus un langage : c'est le SDLC complet, et tous ses enjeux.

> **ENCADRÉ · L'IA PROPOSE, L'ADMINISTRATEUR IA DISPOSE**
>
> Dans un environnement de développement, l'administrateur est celui qui a tous les pouvoirs : il crée les accès, accorde les droits, les révoque, et rien d'important ne se fait sans son autorisation.
>
> C'est exactement le rôle que le développeur endosse face à l'IA. Il définit ce que les agents ont le droit de faire. Il autorise ou bloque chaque action sensible : merger, déployer, ajouter une dépendance, toucher à la production. Il révoque quand un agent sort du cadre.
>
> **L'IA propose, l'administrateur IA dispose.**

3.2 Un rôle, pas une fiche de poste

Autour de l'IA, des spécialisations émergent, et c'est une bonne chose. **L'AI engineer** est l'expert des moteurs : il connaît les LLM et leurs subtilités, les réglages, les tokens, les fenêtres de contexte, les coûts, l'évaluation des modèles. **Le vibe engineer** est l'expert des process d'agents : il conçoit les workflows que les agents exécutent, met en place l'outillage, et interconnecte les solutions IA entre elles.

L'administrateur IA n'est pas une case de plus dans cet organigramme. **C'est un rôle, pas une fiche de poste** : être l'humain qui garantit les six phases du process (pages suivantes), quel que soit l'intitulé. Un AI engineer, un vibe engineer ou un développeur produit peuvent l'endosser, avec plus ou moins de profondeur technique côté modèles ou côté process. Ce qui ne varie pas : c'est un humain qui autorise, et c'est un humain qui répond.

Chapitre 3 · suite

Les six phases : qui fait quoi, et les signaux d'absence

3.3 Les six phases du process, et qui fait quoi

Pour chaque phase : ce que l'IA exécute, ce que l'administrateur IA garantit, et les signaux qui montrent que la phase est absente de votre projet.

Phase 1 · Cadrage

Reformuler le problème avant toute solution : contexte, contraintes, périmètre, niveau de complexité.

L'IA EXÉCUTE	analyse du code existant, exploration des impacts, premières reformulations du problème.
L'ADMIN IA GARANTIT	que le vrai problème est posé (pas le symptôme), que le périmètre est explicite, que la complexité est évaluée avant d'engager des moyens.
SIGNAUX D'ABSENCE	on passe du ticket au code en une étape ; personne ne sait dire pourquoi une tâche est petite ou grosse ; les "quick wins" durent trois semaines.

Phase 2 · Spécification

Décrire ce qu'on va construire : critères d'acceptation, découpage en tranches livrables, plan d'implémentation.

L'IA EXÉCUTE	rédaction de la spec à partir du cadrage, propositions de découpage, plan détaillé des fichiers et des étapes.
L'ADMIN IA GARANTIT	que les critères d'acceptation sont vérifiables, que le découpage permet de livrer par petites tranches, que la spec est validée par le métier avant l'implémentation.
SIGNAUX D'ABSENCE	le code généré "ne fait pas ce qu'on voulait" ; les allers-retours en review portent sur l'intention, pas sur la qualité ; chaque feature est un gros lot indivisible.

Chapitre 3 · suite

Implémentation et vérification : le cadre et les contrôles

Phase 3 · Implémentation

Produire le code, dans un cadre.

L'IA EXÉCUTE	l'écriture du code et des tests qui l'accompagnent, en suivant les conventions du projet.
L'ADMIN IA GARANTIT	le cadre d'exécution : branche isolée, conventions de commit, périmètre de fichiers autorisé, interdits explicites (pas de réécriture d'historique, pas de secrets en clair, pas de dépendance non validée).
SIGNAUX D'ABSENCE	des agents qui travaillent directement sur la branche principale ; des commits illisibles ; du code généré qui ignore les conventions du projet.

Phase 4 · Vérification

Prouver que c'est correct, sûr et conforme.

L'IA EXÉCUTE	exécution des tests, review multi-angles (correction, sécurité, performance, lisibilité), détection d'anomalies.
L'ADMIN IA GARANTIT	que la review est systématique et séparée de la génération (un autre agent, un autre angle, un humain en dernier ressort), que les quality gates sont bloquants : lint, types, tests, sécurité, licences.
SIGNAUX D'ABSENCE	l'IA qui relit son propre code et se trouve très bien ; des gates "indicatifs" que tout le monde contourne ; des tests qui ne testent rien.

> *« Un générateur qui se vérifie lui-même, c'est un contrôle qui n'existe pas. »*

Chapitre 3 · suite

Livraison, amélioration continue, et les gestes du métier

Phase 5 · Livraison

Amener le changement en production, de façon tracée et réversible.

L'IA EXÉCUTE	préparation de la PR, description, changelog, suivi de la CI.
L'ADMIN IA GARANTIT	la traçabilité complète (ticket, PR, release), une CI verte avant tout merge, un rollback possible et testé. Et surtout : c'est lui qui merge. Personne d'autre, et certainement pas un agent.
SIGNAUX D'ABSENCE	on ne sait pas relier la production à une demande d'origine ; les déploiements sont des événements stressants ; le rollback est une théorie.

Phase 6 · Amélioration continue

Capitaliser : décisions, documentation, nettoyage.

L'IA EXÉCUTE	rédaction des ADRs (décisions d'architecture), synchronisation de la documentation avec le code, nettoyage des branches et de l'outillage.
L'ADMIN IA GARANTIT	que les décisions structurantes sont documentées avec leur pourquoi, que la doc reflète le code réel, que le système s'améliore d'une itération à l'autre.
SIGNAUX D'ABSENCE	personne ne sait pourquoi une techno a été choisie ; la doc décrit un système qui n'existe plus ; chaque projet repart de zéro.

ENCADRÉ · LE PROCESS D'UN SENIOR TIENT EN UNE TRENTAINE DE GESTES

Cadrer. Analyser. Clarifier. Spécifier. Découper. Planifier. Isoler l'espace de travail. Implémenter. Tester. Reviewer. Corriger. Valider les gates. Vérifier les licences. Ouvrir la PR. Suivre la CI. Merger. Livrer. Documenter la décision. Synchroniser la doc. Nettoyer. Mesurer. Améliorer.

Chacun de ces gestes est aujourd'hui automatisable par un agent. **Aucun n'est devenu optionnel.** La différence entre une équipe qui maîtrise l'IA et une équipe qui la subit : la première a encodé ces gestes et contrôle leur exécution. C'est l'objet du chapitre 4.

CHAPITRE 04 · LES DEUX FACES

Automatiser, puis contrôler.

4.1 Face A : encoder les pratiques dans l'outillage

Une règle que j'ai vérifiée dans toutes mes missions :

> *« Une bonne pratique qui n'est pas encodée dans l'outillage ne survit pas à la pression du delivery. »*

La revue de code "quand on a le temps", les tests "si possible", la doc "plus tard" : tout cela disparaît au premier sprint tendu. À l'ère des agents, encoder une pratique veut dire la rendre exécutable et systématique :

- **Le format de commit** n'est pas une recommandation : c'est un hook qui rejette ce qui ne le respecte pas.
- **La spec avant le code** n'est pas une bonne intention : c'est le template d'entrée sans lequel l'agent ne démarre pas.
- **La review multi-angles** n'est pas un luxe : c'est un workflow qui fait relire chaque changement par des agents distincts (correction, sécurité, performance) avant l'humain.
- **Les quality gates** ne sont pas un tableau de bord : ce sont des étapes bloquantes de la CI : lint, types, tests, vulnérabilités, licences.

Le développeur 2026 passe une partie de son temps à construire et affiner cet encodage. C'est le nouveau "framework maison" : non plus une librairie de code, mais une chaîne de production.

4.2 Face B : contrôler que le process est réellement appliqué

L'encodage ne suffit pas : il faut auditer. Sur un projet, le développeur-auditeur vérifie des faits, pas des déclarations.

CHECKLIST D'AUDIT DU PROCESS · 10 POINTS

- ☐ Chaque développement part d'un ticket avec un cadrage explicite
- ☐ Les specs existent comme artefacts versionnés, pas comme souvenirs
- ☐ Le travail se fait sur des branches isolées, jamais sur la principale
- ☐ Les conventions (commits, code, structure) sont imposées par l'outillage
- ☐ Chaque changement passe une review séparée de sa génération
- ☐ Les quality gates sont bloquants, personne ne peut les contourner
- ☐ Le scan licences + vulnérabilités tourne dans la CI
- ☐ La traçabilité idée, ticket, PR, release est reconstituable
- ☐ Le rollback a été testé au moins une fois
- ☐ Les décisions d'architecture sont documentées avec leur pourquoi

Comptez vos cases cochées. **En dessous de 7, votre vitesse de production IA travaille contre vous.**

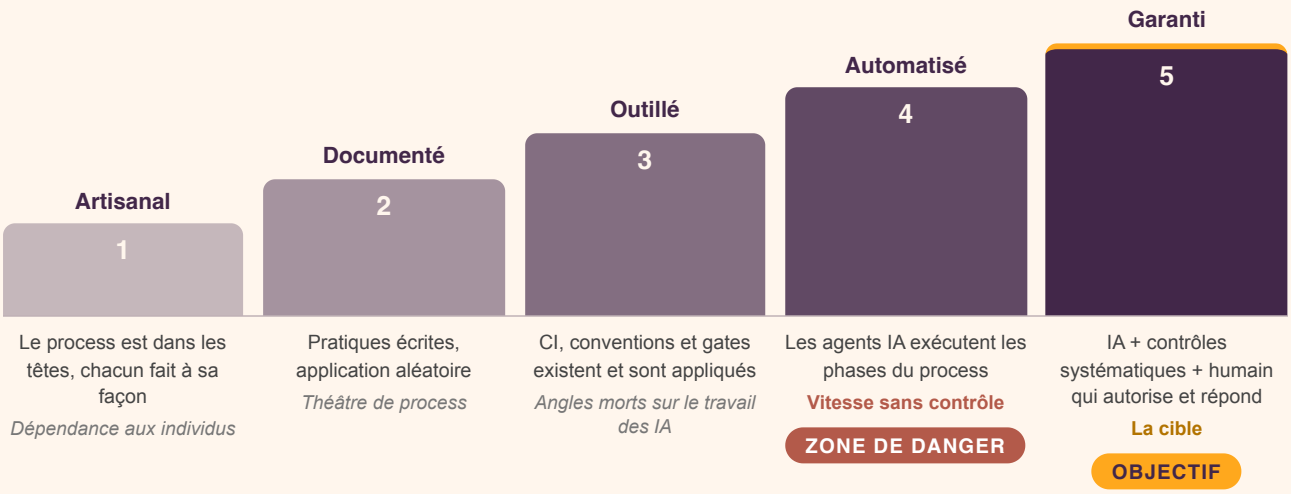
4.3 L'administrateur IA en pratique

Administrer des agents ressemble beaucoup à administrer des accès :

- **Définir les périmètres** : fichiers, dépôts, environnements où un agent a le droit de travailler.
- **Établir les listes d'actions permises** : autorisé sans demander (coder, tester), soumis à approbation (merger, déployer, ajouter une dépendance), interdit (réécrire l'historique, toucher aux secrets).
- **Approuver explicitement les actions sensibles** : l'agent prépare, l'humain autorise.
- **Révoquer immédiatement** quand un agent sort du cadre, puis corriger l'encodage pour rendre la sortie de cadre impossible.

4.4 La grille de maturité du process

Cinq niveaux, du process implicite au process garanti. La progression n'est pas linéaire : le niveau 4 est une zone de danger si les contrôles du niveau 5 ne suivent pas.



LE MESSAGE CENTRAL

Une équipe qui branche des agents sur un process sans contrôles produit de la dette, des failles et des non-conformités **à la vitesse de l'IA**. C'est pour cela que le niveau 4 sans contrôles est plus dangereux que le niveau 2 : au niveau 2, au moins, on produit lentement.

CHAPITRE 05 · LE FLUX

L'interface humaine : de l'idée au logiciel.

Le process ne commence pas au code. Il commence à l'idée, chez un PO, un BA, un client, un dirigeant. Le développeur 2026 est l'architecte du chemin entre cette idée et la production.

5.1 Le workflow unifié

Peu importe votre cadre (Scrum, Kanban, rien du tout) et votre outil (Jira, GitHub Issues, Linear) : ce qui compte, c'est que le chemin soit unifié et tracé.

Pastilles magenta : étapes pilotées par l'humain · Pastilles ambre : étapes exécutées par l'IA sous contrôle humain.

Chaque flèche est un point de passage contrôlé. Aucune étape ne se saute, parce que chaque étape sautée se repaie plus loin, au prix fort.

5.2 Le ticket 2026 n'est plus un pense-bête

Avant, un ticket flou se rattrapait en conversation : le développeur posait des questions, comblait les trous, interprétait. Maintenant, le ticket est l'entrée d'une chaîne de production largement automatisée. **Un ticket flou produit du code IA flou**, en quantité industrielle.

ENCADRÉ · ANATOMIE D'UN TICKET PRÊT POUR L'IA

- **Contexte** : pourquoi cette demande existe, pour qui, avec quel enjeu.
- **Critères d'acceptation** : vérifiables, idéalement exécutables. "L'utilisateur reçoit un email en moins de 2 minutes", pas "améliorer les notifications".
- **Contraintes** : périmètre exclu, technologies imposées ou interdites, exigences de performance et de sécurité.
- **Definition of Done** : tests, doc, review, gates passés, traçabilité.

Le test simple : si un agent IA ne peut pas démarrer sans poser de question de fond, le ticket n'est pas prêt.

5.3 La spec devient le vrai code source

Quand l'implémentation est générée, ce que l'équipe écrit, relit, versionne et fait évoluer, c'est la spécification. C'est elle qui porte l'intention. Deux conséquences pratiques :

- **La spec se relit avec la même exigence que le code autrefois** : en équipe, et avec le métier.
- **La spec se versionne** : elle vit dans le dépôt, à côté du code qu'elle a produit, pas dans un document perdu.

5.4 Ce qui change dans vos rituels

- **L'estimation** : le coût d'une feature n'est plus dans l'écriture du code, il est dans le cadrage et la vérification. Estimer "des jours de dev" n'a plus de sens ; on estime la complexité du cadrage et le risque.
- **La review** : elle remonte d'un cran. On ne débat plus du style (les conventions sont encodées) : on vérifie la conformité à l'intention et la solidité des contrôles.
- **La Definition of Done** : elle intègre désormais la conformité (licences, sécurité, provenance), pas seulement "ça marche et c'est testé".

> *« La spec est le nouveau code source. Le code n'en est plus que la compilation. »*

CHAPITRE 06 · DIFFÉRENCIATEUR

Le garant juridique.

C'est le chapitre que presque personne n'écrit, et c'est pourtant ici que le rôle de garant devient le plus concret. Commençons par la question qui fâche.

6.1 Quand l'IA produit du code, qui en répond ?

Pas l'IA : elle n'a pas de personnalité juridique. Pas son fournisseur : relisez vos conditions d'utilisation, la responsabilité de l'usage du code généré vous revient.

Il reste donc : **votre entreprise, qui livre le logiciel, et la chaîne humaine qui a validé**. Le développeur qui merge du code généré par une IA endosse ce code, exactement comme s'il l'avait écrit. C'est une responsabilité d'éditeur : on publie sous son nom.

6.2 Les familles de licences, en un tableau

FAMILLE	EXEMPLES	LE PRINCIPE	RISQUE SAAS
Permissives	MIT, Apache 2.0, BSD	Usage et modification libres, attribution requise	FAIBLE
Copyleft faible	LGPL, MPL	Réutilisation possible sous conditions ciblées	MODÉRÉ
Copyleft fort	GPL	Distribuer le logiciel impose de fournir le code source	ÉLEVÉ
Copyleft réseau	AGPL	L'usage via le réseau déclenche l'obligation de publication du code	CRITIQUE

> **POURQUOI CE TABLEAU CONCERNE L'IA**
>
> L'IA ne lit pas les licences à votre place. Elle suggère des dépendances pour leur utilité, et peut restituer du code dérivé de n'importe quelle source. La vérification de licence devient un geste du quotidien, pas un sujet d'avocat.

6.3 Le scénario noir : AGPL dans votre SaaS

L'AGPL est la licence qui doit faire réfléchir tout éditeur SaaS. Son principe : si votre service incorpore du code AGPL, le fait de le rendre accessible via le réseau peut vous obliger à publier le code source de ce qui l'incorpore.

Le scénario type : une dépendance pratique ajoutée sans vérification, ou un fragment de code généré par l'IA, dérivé d'un projet AGPL. Personne ne le remarque. Deux ans plus tard, une due diligence (levée de fonds, acquisition) passe votre base de code au scanner. La non-conformité devient un sujet de valorisation, de garantie de passif, parfois de deal cassé.

> *« Le coût de la prévention : un scan automatique dans la CI. Le coût de la découverte tardive : sans commune mesure. »*

6.4 Les trois vecteurs de risque spécifiques à l'IA

1. **Le code restitué** : un modèle peut produire du code très proche d'une source existante sous licence restrictive, sans vous le dire.
2. **Les dépendances suggérées** : l'IA propose des librairies pour leur utilité, pas pour leur licence. Elle ajoutera une dépendance AGPL avec le même enthousiasme qu'une MIT.
3. **Les paquets hallucinés** : l'IA cite parfois des paquets qui n'existent pas. Des attaquants enregistrent ces noms et y placent du code malveillant (le "slopsquatting"). Une dépendance installée sur la foi d'une suggestion est un vecteur d'attaque supply chain.

> **LE RÉFLEXE À INSTALLER**
>
> Aucune dépendance n'entre dans le projet sans vérification de licence et de provenance. Y compris, et surtout, celles suggérées par une IA. Ce réflexe se code : c'est un gate de CI, pas une bonne résolution.

Chapitre 6 · suite

Les gestes du garant, et la signature du merge

6.5 Les gestes du garant

CHECKLIST CONFORMITÉ · 6 POINTS

- ☐ Une politique de licences explicite : familles autorisées, tolérées, interdites
- ☐ Un scan automatique des licences de toutes les dépendances, bloquant dans la CI
- ☐ Un scan de vulnérabilités et de secrets, bloquant dans la CI
- ☐ Un SBOM (inventaire des composants logiciels) généré et tenu à jour
- ☐ Aucune dépendance sans vérification de licence et de provenance, IA comprise
- ☐ Une trace : qui a validé quoi, quand, sur quelle base

6.6 Le merge est une signature

EN CLIQUANT "MERGE", VOUS ENDOSSEZ	LA QUESTION À VOUS POSER
Le fonctionnel	Est-ce que cela fait ce que le ticket demande ?
La qualité	Est-ce maintenable par quelqu'un d'autre dans 6 mois ?
La sécurité	Qu'est-ce que cela expose, et l'ai-je vérifié ?
Le juridique	D'où vient ce code, et sous quelle licence ?

Si une de ces lignes vous est invisible au moment de merger, le problème n'est pas vous : c'est votre process. Retournez au chapitre 4.

Avertissement : ce chapitre décrit des pratiques d'ingénierie, pas un avis juridique. Pour les sujets d'exposition réelle (contrats, contentieux, opérations de M&A), faites intervenir un conseil spécialisé.

CHAPITRE 07 · L'ORGANISATION

Réorganiser l'entreprise autour du développeur 2026.

Les six premiers chapitres décrivent un rôle. Mais un rôle nouveau ne tient pas dans une organisation inchangée. Si le développeur devient l'administrateur IA, garant du process, deux fonctions doivent muter avec lui : celle qui fait *entrer* les gens (les RH), et celle qui décide *où* et *comment* l'IA entre dans l'entreprise (la direction). Ce chapitre les traite à haute altitude ; le détail opérationnel fera l'objet d'un document dédié aux dirigeants et aux RH.

7.1 Les RH : recruter le garant, pas le technologue

Ouvrez n'importe quel site d'emploi en 2026 : la même annonce revient, "développeur senior, 10 ans d'expérience" sur un outil qui en a trois. La formule prête à sourire, mais elle trahit un problème structurel, pas une bourde isolée.

Depuis vingt ans, les RH recrutent sur un raccourci : le mot-clé technologique sert de *proxy* de la compétence. Ce raccourci a cessé de fonctionner :

- Les outils changent désormais plus vite qu'un cycle de recrutement. Le temps de rédiger la fiche de poste, de sourcer et d'intégrer, la stack a déjà bougé.
- La compétence durable n'est plus l'outil, c'est le jugement sur le process (les six phases du chapitre 3). Il se transpose d'un outil à l'autre ; l'inverse n'est pas vrai.

Recruter le développeur 2026, c'est changer l'objet de la mesure : passer de la fiche de poste "stack" au **profil de compétences**. On ne cherche plus quelqu'un qui "connaît" une technologie, mais quelqu'un qui sait mener un développement de bout en bout et faire respecter ce cadre, par les humains comme par les agents.

> **ENCADRÉ · LE DÉPLACEMENT DE LA FICHE DE POSTE**
>
> Hier, une liste de technologies : "Java, Spring, Kafka, AWS, 5 ans minimum". Aujourd'hui, une liste de garanties : sait poser le vrai problème, sait rendre des critères d'acceptation vérifiables, sait séparer génération et vérification, sait ce qu'il endosse en cliquant "merge". La technologie devient une ligne de contexte, plus le critère.

Ce déplacement dépasse le service RH. Les ressources humaines traduisent la **stratégie de l'entreprise en compétences à acquérir** : le profil à recruter n'est pas "un dev de plus", c'est un garant de process, à redéfinir avec le comité de direction avant d'écrire la moindre annonce.

7.2 La direction : diagnostiquer avant de déployer

Côté management, le réflexe le plus répandu est aussi le plus coûteux : distribuer des licences Claude, Copilot ou Cursor à toutes les équipes, et croire qu'on "fait de l'IA". **Distribuer des licences n'est pas une stratégie. C'est une dépense.**

Les faits sont têtus. Chez Uber, le budget IA annuel a été consommé en quatre mois, obligeant l'entreprise à plafonner à 1 500 $ par mois et par employé. Une entreprise a dépensé un demi-milliard de dollars en un seul mois, faute de plafond. Chez une fintech, un cadre a brûlé 81 000 $ de crédits en générant un jeu vidéo "pour voir". Ce ne sont pas des vols isolés : c'est, à l'échelle de l'organisation, le "niveau 4 sans contrôles" du chapitre 4. On a donné la puissance, pas le cadre.

LA PROMESSE DE PRODUCTIVITÉ, EN FACE	SOURCE
Développeurs expérimentés **19 % plus lents** avec l'IA (en se croyant plus rapides)	METR, 2025
95 % des pilotes GenAI sans retour mesurable	MIT, 2025
Entreprises abandonnant la majorité de leurs initiatives IA : de **17 % à 42 %** en un an	S&P Global, 2025

Ces chiffres ne disent pas "l'IA ne marche pas" : d'autres études (Google, GitHub) mesurent des gains réels de 10 à 20 %. Ils disent que **les gains sont réels mais conditionnels, et s'effondrent quand l'IA est déployée sans savoir à quoi elle doit servir.**

D'où le principe : **jamais d'IA sans étude amont.** Où, dans ce que nous produisons, l'IA crée-t-elle de la valeur ? C'est une analyse de chaîne de valeur : l'IA accélère la valeur qui existe déjà, elle ne la crée pas à partir de rien. L'adoption de l'IA n'est pas un événement, c'est la **continuité de la transformation digitale** : diagnostic d'abord, périmètre ensuite, déploiement gouverné à la fin.

> **ENCADRÉ · LES TROIS QUESTIONS DU DIRIGEANT AVANT D'OUTILLER**
>
> - Quelle valeur créons-nous pour nos clients, et par quelles activités ? (la chaîne de valeur)
> - Sur lesquelles l'IA est-elle un accélérateur, et sur lesquelles un coût sans retour ?
> - Une fois branchée, qui gouverne les droits, les périmètres et les budgets des agents ?

> *« Recruter le garant, diagnostiquer avant de déployer : sans ces deux mutations, le meilleur administrateur IA reste seul. »*

CHAPITRE 08 · LES PIÈGES

Les 5 pièges de la transition.

PIÈGE 1

Accélérer le chaos

Brancher des agents IA sur un process inexistant. L'IA amplifie ce qu'on lui donne : un flux sans cadrage ni contrôles produit de la dette et des failles, plus vite que jamais.

ANTIDOTE Le process d'abord (chapitres 3 et 4), la puissance ensuite.

PIÈGE 2

Évaluer (et se vendre) au volume de code

Mesurer les développeurs en lignes de code ou en vélocité de production, à l'heure où ce volume est fourni par les machines.

ANTIDOTE Évaluer la tenue du process : lead time, taux de reprise, incidents, conformité.

PIÈGE 3

Laisser l'IA s'auto-reviewer

Le même agent qui génère et qui valide, c'est un contrôle qui n'existe pas. La complaisance d'un modèle envers sa propre production est documentée et prévisible.

ANTIDOTE Séparer générateur et vérificateur, et garder l'humain comme dernier signataire des actions sensibles.

PIÈGE 4

Découvrir la conformité au premier audit

Licences, vulnérabilités, provenance : tant que rien ne casse, le sujet semble théorique. Il devient très concret le jour d'une due diligence ou d'un incident.

ANTIDOTE La conformité est un gate de CI dès le premier jour, pas un chantier de dernière minute.

PIÈGE 5

Tout automatiser d'un coup

Vouloir encoder les six phases, tous les gates et tous les agents en un sprint. L'équipe rejette la greffe, et on conclut à tort que "l'IA ne marche pas chez nous".

ANTIDOTE Une pratique à la fois : encoder, prouver la valeur, étendre.

CHAPITRE 09 · LE PLAN

30 jours pour pivoter.

9.1 D'abord, mesurez où vous en êtes

Dix questions suffisent pour situer votre process : cadrage, specs, isolation du travail, contrôles, conformité, traçabilité.

AUTO-ÉVALUATION EN LIGNE · 5 MINUTES

Votre process est-il garanti ? →

10 questions, votre score sur la grille de maturité de la page 16, votre niveau (1 à 5), et la phase à travailler en priorité. Résultat immédiat par email.

Faire l'auto-évaluation →

9.2 Les 4 semaines

SEMAINE 1
Cartographier

La carte de vos trous

Prenez les 5 derniers développements livrés. Pour chacun, reconstituez le chemin réel : où était le cadrage ? la spec ? la review ? qui a autorisé quoi ? Les six phases du chapitre 3 sont votre grille de lecture.

SEMAINE 2
Encoder

Trois pratiques, pas dix

Par exemple : un template de ticket (chapitre 5), une convention de commit imposée par hook, un gate de CI bloquant (tests + lint). De quoi prouver la mécanique : encodé = appliqué.

SEMAINE 3
Contrôler

Du niveau 4 potentiel au niveau 5

Le scan licences + vulnérabilités dans la CI (chapitre 6). La review séparée de la génération. La traçabilité ticket, PR, release.

SEMAINE 4
Boucler

Le métier dans la boucle

Présentez le workflow unifié à vos PO et BAs. Mettez le template de ticket entre leurs mains. Instaurez la Process Review mensuelle (page 26). Le pivot est amorcé.

9.3 Template : la Process Review mensuelle

Durée : 45 minutes · **Participants :** équipe de dev + PO

ÉTAPE 1
10 min

Les métriques du mois

- Lead time médian : ___ j (vs mois précédent : ___ j)
- Part des développements avec spec validée en amont : ___ %
- Gates contournés ou désactivés : ___
- Alertes licences / sécurité traitées : ___

ÉTAPE 2
15 min

Les 3 derniers ratés de process

Pour chacun : quelle phase a été sautée ou bâclée ? Qu'est-ce que ça a coûté ?

ÉTAPE 3
5 min

Ce qui a bien tenu

Une pratique encodée qui a prouvé sa valeur ce mois-ci. On la garde, on la cite.

ÉTAPE 4
10 min

Une amélioration, une seule

La prochaine pratique à encoder ou le prochain contrôle à installer. Un owner, une date.

ÉTAPE 5
5 min

La décision d'administration

Un droit à accorder ou à révoquer aux agents : périmètre élargi ici, approbation exigée là. L'administration des IA est un sujet d'équipe, pas un réglage individuel.

POURQUOI UN RITUEL MENSUEL

Le process garanti n'est pas un état, c'est une pratique. 45 minutes par mois suffisent pour que l'équipe reste propriétaire de son système de production, au lieu de le subir.

À propos de l'auteur

KAMANGA Software Craftsman

L'AUTEUR

Hervé Muludiki.

Coach craft et engineering leader international. 25 ans d'expérience et tous les rôles : développeur, tech lead, engineering manager, directeur technique, coach. Intervenu chez BNP Paribas, Crédit Agricole, Canal+, Agirc-Arrco et d'autres grandes structures, sur des transformations où l'enjeu économique du logiciel est réel.

Un double regard, business et technique. Diplômé d'un Executive MBA (Epitech Executive, Paris), j'aborde le logiciel comme un actif stratégique qui se pilote et qui doit produire de la valeur. Et je développe encore aujourd'hui, par passion : je comprends la technique dans le détail, et je sais ce que coûte (et ce que rapporte) un logiciel extrêmement bien fait, pérenne, qui dure dans le temps.

> *« La technologie est un outil au service du métier : la maîtriser, c'est garantir à l'entreprise des revenus solides et réguliers, et à ses clients une satisfaction durable. »*

Comment je peux vous aider

J'interviens sur toutes les étapes décrites dans ce livre blanc, du diagnostic à la réalisation :

- **Conseil et accompagnement** : audit du process de développement (les six phases, les contrôles, la conformité), transformation des pratiques, coaching des équipes et des managers techniques.
- **Réalisation sur mesure** : conception et construction de vos infrastructures techniques d'ingénierie augmentée par l'IA : marketplaces, plugins et outillage ultra-personnalisés à vos process et à vos technologies.

Rester en contact

- LinkedIn : **@kamangacode** · linkedin.com/in/kamangacode
- Site : **kamanga.fr**
- Email : **herve@kamanga.fr**

La prochaine étape.

Le code n'est plus le produit du développeur. Le produit du développeur 2026, c'est un **process garanti** : un système, humains et IA, qui transforme des intentions métier en logiciel fiable, maintenable et légalement irréprochable.

> *« Les développeurs qui prennent ce virage ne sont pas remplacés par l'IA : ils deviennent ceux qui décident de ce que l'IA a le droit de faire. »*

POUR LES CTOS ET LEADERS ENGINEERING

Un regard extérieur sur votre process →

Les six phases, votre outillage IA, vos contrôles, votre conformité : 30 minutes pour situer votre équipe et identifier le premier chantier.

Réserver un Discovery Call →

POUR SITUER VOTRE ÉQUIPE EN 7 MINUTES

AI-Ready Engineering Team Checklist →

25 questions, un score de maturité, un plan d'action 30/60/90 jours adapté à vos 3 gaps critiques. Gratuit.

Démarrer la checklist gratuite →

POUR LES DÉVELOPPEURS

EMA-Dev : l'auto-évaluation du développeur →

Situez votre propre profil face au rôle d'administrateur IA, et identifiez vos prochains gestes de progression.

Faire l'auto-évaluation EMA-Dev →

POUR LES MANAGERS ET LEURS ÉQUIPES · FORMATION

Formez votre équipe au métier de développeur augmenté →

Le programme qui outille vos développeurs pour le rôle décrit dans ce livre blanc : administrer les agents IA, tenir les six phases sans tout coder, et garantir le process de bout en bout. Conçu pour les managers qui pilotent la montée en compétences de leur équipe.

Découvrir la formation →